User's Manual for elegant

Program Version 2023.4
Advanced Photon Source
Michael Borland

December 13, 2023

Note: another source of help for elegant is the on-line forum. Users are encouraged to join and participate. At minimum, users should subscribe to the "Bugs" topic, since this is where bug notifications are posted.

A set of examples and scripts is available from the software download page that demonstrates many features of elegant. A brief overview of elegant is also available, which introduces the capabilities at a high level.

1 Highlights of What's New in Version 2023.4

Here is a summary of what's changed since release 2023.3. Historical change logs are collected in Section 13.

1.1 Bug Fixes for Elements

- None.

1.2 Bug Fixes for Commands

- None.

1.3 New and Modified Elements

- The CWIGGLER element now supports imposition of uniform horizontal and vertical magnetic fields. These can be used, for example, for trajectory correction.
- The calculation of the Bassetti-Erskine formula, used for obtaining the electric field for gaussian bunches, is now faster by about a factor of two, thanks to use of code from http://ab-initio.mit.edu/Faddeeva. This change by N. Kuklev (APS) affects the IONEFFECTS and BEAMBEAM elements.

1.4 New and Modified Commands

- Added slope-response output and full-names options to the correction_matrix_output command.

- Added store_centers parameter to the floor_coordinate command. This allows storing magnet centers in variables for use in optimization.
- Added profile parameter to the run_setup command, allowing output of (ρ, K_1, K_2, K_3) as a function of s.
- Added fse_units parameter to the chromaticity and correct_tunes command, allowing output of the matrices in possibly more convenient form.
- The load_parameters command now supports wild-cards in the ElementName data. If present, all matching elements are loaded with the same values.

1.5 Other Changes

- None.

1.6 Changes Specific to the MPI Parallel Version

- Fixed a bug when using the find_aperture command in optimization_mode=1, which caused the program to crash or halt.

1.7 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

1.8 Changes to Related Programs and Files

The elegant distribution includes many programs and scripts that perform computations with elegant output data, provide interfaces with other programs, or prepare data for use with elegant. These are listed in Section 8.

Changes to these tools in this release include

- curvedDipoleFringeCalc now computes the lengths of edge drifts needed to keep the vertex point fixed. It also takes most configuration data from the particle output file created by BGGEXP.

1.9 Known Bugs, Problems, and Limitations

- The centroid and sigma output from run_setup is not correct when BRANCH elements are used.
- The REFERENCE_CORRECTION feature of the CSBEND element is ignored while performing calculations related to the moments_output command.
- Setting CHANGE_T=1 on RFCA and RFCW elements can give invalid results when tracking beams with very large time spread compared to the bunch length.

- Twiss output contains entries for the higher-order dispersion, tune shifts with amplitude, higher-order chromaticity, and tune spreads due to chromaticity and amplitude *even when these are not calculated*, which is potentially misleading. The values are zero when the calculation is not requested.

- Computation of closed orbits and Twiss parameters will not always include the effects of synchrotron radiation losses when these are imposed using `SREFFECTS` elements. See the documentation for `SREFFECTS` for details.

- Computation of beam moments does not include synchrotron radiation effects from `UKICKMAP` elements.

- The file created with the parameters field of `run_setup` does not contain any non-numerical parameters of the lattice.

- Computation of radiation integrals does not include the effect of steering magnets.

- There is a bug related to using `ILMATRIX` that will result in a crash if one does not request computation of the twiss parameters. If you encounter this problem, just add the following statement after the `run_setup` command:

```
&twiss_output
        matched = 1
&end
```

- The `OUTPUT_FILE` feature of the `TFBDRIVER` will produce a file with missing data at the end of the buffer if the `OUTPUT_INTERVAL` parameter is not a divisor of the number of passes.

- When the `KQUAD` element was split (with the `divide_elements` command or `element_divisions`, any edge multipoles get evaluated at the interior boundaries. In addition, the `LEFFECTIVE` cannot be used.

2 Credits

Contributors to `elegant` include M. Borland, M. Carla', N. Carmignani, AJ Dick, Z. Duan, M. Ehrlichman, L. Emery, W. Guo, R. Lindberg, V. Sajaev, R. Soliday, Y.-P. Sun, C.-X. Wang, Y. Wang, Y. Wu, and A. Xiao. Contributors to related programs and scripts include M. Borland, R. Dejus, L. Emery, A. Petrenko, H. Shang, Y. Wang, A. Xiao, and B. Yang. R. Soliday is responsible for multi-platform builds and distribution. Of course, we also appreciate the many suggestions, comments, and bug reports from users.

If you use `elegant` in your research, we appreciate a citation. For `elegant`, the citation is

M. Borland, "elegant: A Flexible SDDS-Compliant Code for Accelerator Simulation," Advanced Photon Source LS-287, September 2000.

Additional contributors for the parallel version include Y. Wang and H. Shang. The additional citation for `Pelegant` is

Y. Wang and M. Borland, "Pelegant: A Parallel Accelerator Simulation Code for Electron Generation and Tracking", Proceedings of the 12th Advanced Accelerator Concepts Workshop, AIP Conf. Proc. 877, 241 (2006).

Additional contributors for the GPU version include K. Amyx, J. R. King, and I. V. Pogorelov. The additional citation for the GPU version is

I. V. Pogorelov, J. R. King, K. M. Amyx, M. Borland, and R. Soliday, "Current status of the GPU-accelerated ELEGANT," Proceedings of 2015 International Particle Accelerator Conference, 623 (2015).

3 Introduction

`elegant` stands for "ELEctron Generation ANd Tracking," a somewhat out-of-date description of a fully 6D accelerator program that now does much more than generate particle distributions and track them. `elegant`, written entirely in the C programming language[1], uses a variant of the MAD[2] input format to describe accelerators, which may be either transport lines, circular machines, or a combination thereof. Program execution is driven by commands in a namelist format.

This document describes the features available in `elegant`, listing the commands and their arguments. The differences between `elegant` and MAD formats for describing accelerators are listed. A series of examples of `elegant` input and output are given. Finally, appendices are included describing the post-processing programs.

3.1 Program Philosophy

For all its complexity, `elegant` is not a stand-alone program. For example, most of the output is not human-readable, and `elegant` itself has no graphics capabilities. These tasks are handled by a suite of post-processing programs that serve both `elegant` and other physics programs. These programs, collectively known as the SDDS Toolkit[8, 9], provide sophisticated data analysis and display capabilities. They also serve to prepare input for `elegant`, supporting multi-stage simulation.

Setting up for an `elegant` run thus involves more than creating input files for `elegant` per se. A complicated run will typically involve creation of a post-processing command file that processes `elegant` output and puts it in the most useful form, typically a series of graphs. Users thus have the full power of the SDDS Toolkit, the resident command interpreter (e.g., the UNIX shell), and their favorite scripting language (e.g., Tcl/Tk) at their disposal. The idea is that instead of continually rewriting the physics code to, for example, make another type of graph or squeeze another item into a crowded table, one should allow the user to tailor the output to his specific needs using a set of generic post-processing programs. This approach has been quite successful, and is believed particularly suited to the constantly changing needs of research.

Unlike many other programs, `elegant` allows one to make a single run simulating an arbitrary number of randomizations or variations of an accelerator. By using the SDDS toolkit to postprocess the data, the user's postprocessing time and effort do not depend on how many random seeds or situations are chosen. Hence, instead of doing a few simulations with a few seed numbers or values, the user can simulate hundreds or even thousands of instances of one accelerator to get an accurate representation of the statistics or dependence on parameters, with no more work invested than in doing a few simulations.

In addition, complex simulations such as start-to-end jitter simulations[11] and top-up tracking[12] can be performed involving hundreds or thousands of runs, with input created by scripts depending on the SDDS toolkit. These simulations make use of concurrent computing on about 20 workstation using the Distributed Queueing System[10]. Another example is the `elegantRingAnalysis` script, which allows using many workstations for simulation of storage ring dynamic and momentum aperture, frequency maps, and so on. Clearly, use of automated postprocessing tools greatly increases the scale and sophistication of simulations possible.

In passing, we note another "philosophical" point about `elegant`, namely, the goal of complete backward compatibility. We consider it unacceptable if a new version of the program gives different answers than an old version, unless the old version was wrong. Hence, there are sometimes less-than-ideal default settings in `elegant`, incorrect spelling of parameters, etc., that are never fixed, because doing so would break old input files. It helps to read the manual pages carefully for the more complex features to ensure that the defaults are understood and appropriate.

3.2 Capabilities of `elegant`

`elegant` started as a tracking code, and it is still well-suited to this task. `elegant` tracks in the 6-dimensional phase space $(\mathrm{x}, \mathrm{x}', \mathrm{y}, \mathrm{y}', \mathrm{s}, \mathrm{ffi})$, where x (y) is the horizontal (vertical) transverse coordinate, primed quantities are slopes, s is the *total, equivalent* distance traveled, and δ is the fractional momentum deviation[3]. Note that these quantities are commonly referred to as (x, xp, y, yp, s, dp) in the namelists, accelerator element parameters, and output files. ("dp" is admittedly confusing—it is supposed to remind the user of $\Delta\mathrm{P}/\mathrm{P_o}$. Sometimes this quantity is referred to as "delta.")

In some elements, `elegant` uses canonical coordinates in place of the slopes. These are defined as

$$\begin{aligned} q_x &= \frac{x'(1+\delta)}{\sqrt{1+x'^2+y'^2}} \\ q_y &= \frac{y'(1+\delta)}{\sqrt{1+x'^2+y'^2}} \end{aligned} . \tag{1}$$

The inverse relationship is

$$\begin{aligned} x' &= \frac{q_x}{\sqrt{(1+\delta)^2-q_x^2-q_y^2}} \\ y' &= \frac{q_y}{\sqrt{(1+\delta)^2-q_x^2-q_y^2}} \end{aligned} . \tag{2}$$

Tracking may be performed using matrices (of selectable order), canonical kick elements, numerically integrated elements, or any combination thereof. For most elements, second-order matrices are available; matrix concatenation can be done to any order up to third. Canonical kick elements are available for bending magnets, quadrupoles, sextupoles, and higher-order multipoles; all of these elements also support optional classical synchrotron radiation losses. Among the numerically integrated elements available are extended-fringe-field bending magnets and traveling-wave accelerators. A number of hybrid elements exist that have first-order transport with exact time dependence, e.g., RF cavities. Some of the more unusual elements available are third-order alphamagnets[5, 4], time-dependent kicker magnets, voltage-ramped RF cavities, beam scrapers, and beam-analysis "screens."

Several elements support simulation of collective effects, such as short- and long-range wakefields, resonator impedances, intra-beam scattering, coherent synchrotron radiation, and the longitudinal space charge impedance.

A wide variety of output is available from tracking, including centroid and sigma-matrix output along the accelerator, phase space output at arbitrary locations, turn-by-turn moments at arbitrary locations, histograms of particle coordinates, coordinates of lost particles, and initial coordinates of transmitted particles. In addition to tracking internally generated particle distributions, `elegant` can track distributions stored in external files, which can either be generated by other programs or by previous `elegant` runs. Because `elegant` uses SDDS format for reading in and writing out particle coordinates, it is relatively easy to interface `elegant` to other programs using files that can also be used with SDDS to do post-processing for the programs.

`elegant` allows the addition of random errors to virtually any parameter of any accelerator element. One can correct the orbit (or trajectory), tunes, and chromaticity after adding errors, then compute Twiss parameters, track, or perform a number of other operations. `elegant` makes it easy to evaluate a large number of ensembles ("seeds") in a single run. Alternatively, different ensembles can be readily run of different CPUs and the SDDS output files combined.

In addition to randomly perturbing accelerator elements, `elegant` allows one to systematically vary any number of elements in a multi-dimensional grid. As before, one can track or do other computations for each point on the grid. This is a very useful feature for the simulation of experiments, e.g., emittance measurements involving beam-size measurements during variation of one or more quadrupoles[6].

Like many accelerator codes, `elegant` does accelerator optimization. It will optimize a user defined function of the transfer matrix elements (up to third-order), beta functions, tunes, chromaticities, radiation integrals, natural emittance, floor coordinates, beam moments, etc. It also has the ability to optimize results of tracking using a user-supplied function of the beam parameters at one or more locations. This permits solution of a wide variety of problems, from matching a kicker bump in the presence of nonlinearities to optimizing dynamic aperture by adjusting sextupoles.

`elegant` provides several methods for determining accelerator aperture, whether dynamic or physical. One may do straightforward tracking of an ensemble of particles that occupies at uniform grid in (x, y) space. One may also invoke a search procedure that finds the aperture boundary. A related feature is the ability to determine the frequency map for an accelerator, to help identify aperture-limiting resonances.

In addition to using analytical expressions for the transport matrices, `elegant` supports computation of the first-order matrix and linear optics properties of a circular machine based on tracking. A common application of this is to compute the tune and beta-function variation with momentum offset by single-turn tracking of a series of particles. This is much more efficient than, for example, tracking and performing FFTs (though `elegant` will do this also). This both tests analytical expressions for the chromaticity and allows computations using accelerator elements for which such expressions do not exist (e.g., a numerically integrated bending magnet with extended fringe fields).

A common application of random error simulations is to set tolerances on magnet strength and alignment relative to the correctability of the closed orbit. A more efficient way to do these calculations is to use correct-orbit amplification factors[6]. `elegant` the computes amplification factors and functions for corrected and uncorrected orbits and trajectories pertaining to any element that produces an orbit or trajectory distortion. It simultaneously computes the amplification functions for the steering magnets, in order to determine how strong the steering magnets will need to be.

4 Digression on the Longitudinal Coordinate Definition

A word is in order about the definition of s, which we've described as the *total, equivalent* distance traveled. First, by *total* distance we mean that s is *not* measured relative to the bunch center or a fiducial particle. It is entirely a property of the individual particle and its path through the accelerator.

To explain what we mean by *equivalent* distance, note that the relationship between s and arrival time t at the observation point is, for each particle, $s = \beta ct$, where βc is the instantaneous velocity of the particle. Whenever a particle's velocity changes, `elegant` recomputes s to ensure that this relationship holds. s is thus the "equivalent" distance the particle would have traveled at the present velocity to arrive at the observation point at the given time. This book-keeping is required because `elegant` was originally a matrix-only code using s as the longitudinal coordinate.

Users should keep the meaning of s in mind when viewing statistics for s, for example, in the `sigma` or watch point output files. A quantity like `Ss` is literally the rms spread in s. It is *not* defined as $\sigma_t/(\langle\beta\rangle c)$. A nonrelativistic beam with velocity spread will show no change in `Ss` in a drift space, because the distance traveled is the same for all particles.

5 Fiducialization in `elegant`

In some tracking codes, there is a "fiducial particle" that is tracked along with the beam. This particle follows the ideal trajectory or orbit, with the ideal momentum, and at the ideal phase. There is no fiducial particle in `elegant`. Instead, fiducialization is typically based on statistical properties of the bunch. This can be performed on a bunch-by-bunch basis, or for the first bunch seen in a run. The latter method must be used if one wants to look at the effects of changing phase, voltage, or magnets relative to some nominal configuration.

Internally, `elegant` fiducializes each element in the beamline. Fiducializing an element means determining the reference momentum and arrival time (or phase) for that element. If the reference momentum does not change along a beamline and no time-dependent elements are involved, then fiducialization is irrelevant. All elements are fiducialized at the central momentum defined in `run_setup`.

A number of commands have parameters for controlling fiducialization:

- The `always_change_p0` parameter of `run_setup` causes `elegant` to re-establish the central momentum after each element when fiducializing. This may be more convenient than setting the `CHANGE_P0` parameter on the elements themselves. However, it can have unexpected consequences, such as changing the central momentum to match changes in beam momentum due to synchrotron radiation.

- `run_control` has four parameters that affect fiducialization, which come into play when multi-step runs are made. Typically, these are runs that involve variation of elements, addition of errors, or loading of multiple sets of parameters.

  - `reset_rf_for_each_step` — If nonzero, the rf phases are re-established for each beam tracked. If this is 1 (the default), the time reference is discarded after each bunch is tracked. This means that bunch-to-bunch phasing errors due to time-of-flight differences would be lost.
  - `first_is_fiducial` — The first bunch seen is taken to establish the fiducial phases and momentum profile. If one is simulating, for example, successive beams in a fixed accelerator, this should be set to 1. Otherwise, the momentum reference is discarded after each bunch is tracked. N.B.: as of version 27.0.1, setting `first_is_fiducial=1` does not imply `always_change_p0=1`. You must set this separately, or use the `CHANGE_P0` parameter on various elements (e.g., `RFCA`) to further specify how to set the fiducial momentum profile.
  - `restrict_fiducialization` — If nonzero, then momentum profile fiducialization occurs only after elements that are known to possibily change the momentum. It would not occur, for example, after a scraper that changes the average beam momentum by removing a low-momentum tail. This is a convenience that, essentially, allows modifying the impact of setting `always_change_p0=1`.

  - n_passes_fiducial — If positive, sets the number passes used for fiducial tracking to be different from the n_passes value. For ring fiducialization, should probably always be set to 1.

- The bunched_beam command has a first_is_fiducial parameter that is convenient for use with the first_is_fiducial mode established by run_control. If nonzero, this parameter causes elegant to generate a first bunch with only one particle. This is very useful if one wants to track with many particles but doesn't want to waste time fidicializing with a many-particle bunch.

Here are some examples that may be helpful.

- *Scanning a phase error in a linac with a bunch compressor:* The scan is performed using the vary_element command. For this to work properly, it is necessary to fidcualize the system with zero phase error. Hence, one must use the enumeration feature of vary_element to provide an input file with the phase errors and the file must be sorted so that the row with zero phase error is first. Further, one must set reset_rf_for_each_step = 0 and first_is_fiducial = 1 in run_control, and CHANGE_P0=1 on all rf cavity elements. (See the bunchComp/phaseSweep and bunchComp/dtSweep examples.)

- *Scanning the voltage of a linac to simulate different operating energy choices at the compressor:* In this case, one scans the linac voltage, but wants to fiducialize the system for each voltage. (It's a change in design, not an error or perturbation.) One again uses vary_element, but nothing special needs to be done about the order of the voltage values. One must set reset_rf_for_each_step = 1 and first_is_fiducial = 0 in in run_control, and CHANGE_P0=1 on all rf cavity elements. (See the bunchComp/energySweep example.)

- *Simulation of phase and voltage jitter:* In this case, one uses the error_element command to impart errors to the PHASE and VOLT parameters of rf cavity elements. However, the first beam through the system must not see any errors. This is accomplished by setting no_errors_for_first_step=1 in error_control. One can also (optionally) use a 1-particle beam for fiducialization by setting first_is_fiducial=1 in bunched_beam. In addition, one must set reset_rf_for_each_step = 0 and first_is_fiducial = 1 in run_control, and CHANGE_P0=1 on all rf cavity elements. (See the bunchCompJitter/jitter example.)

6 Preparing beams for bunch-mode simulations

Certain collective-effects elements in elegant can operate under the assumption that the beam is organized into bunches. This includes the FRFMODE, FTRFMODE, LRWAKE, RFMODE, WAKE, TRFMODE, TRWAKE, ZLONGIT, and ZTRANSVERSE elements. At present, this behavior is only available when loading a beam from an external file using the sdds_beam command. A typical sequence is to run elegant once to generate a beam file using bunched_beam, then load that beam into a subsequent run.

This beam file may either contain the entire beam (all the bunches) or it may contain a single bunch. In the latter case, the single bunch must be duplicated using the n_duplicates and duplicate_stagger parameters of sdds_beam. Otherwise, in the beam-generation run, the run_control command must be used to specify both the number of bunches (using n_steps) and the bunch frequency (using bunch_frequency). The beamline for this run would typically consist simply of a zero-length drift space, so that the output file from the run_setup command contains the coordinates for each bunch as generated, with no modifications. Once the beam is generated, it can be used as the input file for `sdds_beam` with `track_pages_separately=0` and `use_bunched_mode=1`.

For those who prepare beams using other programs, it may be helpful to understand how the organization of the beam into bunches is specified. The relevant data from the beam file are the values in the `IDSlotsPerBunch` parameter and `particleID` column. The `particleID` is generally a unique positive integer for each particle. When $S =$`IDSlotsPerBunch` is non-zero, the bunch index is computed as $\lfloor (I-1)/S \rfloor$, where I is the particle ID. For example, with `IDSlotsPerBunch`=1000, particle IDs from 1 to 1000 would be in bunch 0, from 2001-3000 would be bunch 1, and so on. This mechanism allows specifying the bunch structure without adding columns to the beam file, and also handles particle loss automatically.

Note that although in the case of beams generated with `bunched_beam` the individual bunches appear in separate pages of the beam file, this is not necessary.

7 Namelist Command Dictionary

The main input file for an `elegant` run consists of a series of namelists, which function as commands. Most of the namelists direct `elegant` to set up to run in a certain way. A few are "action" commands that begin the actual simulation. FORTRAN programmers should note that, unlike FORTRAN namelists, these namelists need not come in a predefined order; `elegant` is able to detect which namelist is next in the file and react appropriately.

7.1 Commandline Syntax

The commandline syntax for `elegant` is of the form

`elegant {`*`inputfile`*`|-pipe=in} [-rpnDefns=`*`filename`*`] [-configuration=`*`filename`*`]`
`[-macro=`*`tag1`*`=`*`value1`*`[,`*`tag2`*`=`*`value2`*`...]`

inputfile is the name of the command input file, which is a series of namelist commands directing the calculations. Alternatively, one may give the `-pipe=in` option, allowing `elegant` to be fed a stream of commands by another program or script. The `-rpnDefns` option allows providing the name of the RPN definitions file as an alternative to defining the `RPN_DEFNS` environment variable. The `-configuration` option allows providing the name of an input file to be read prior to *inputfile*; this can be used for configuring `elegant` using, e.g., the `global_settings` command; this is an alternative to using the `ELEGANT_CONFIGURATION` environment variable. The `-macro` option allows performing text substitutions in the command stream. Multiple `-macro` options may be given. Usage is described in more detail below.

7.2 General Command Syntax

Each namelist has a number of variables associated with it, which are used to control details of the run. These variables come in three data types: (1) `long`, for the C long integer type. (2) `double`, for the C double-precision floating point type. (3) `STRING`, for a character string enclosed in double quotation marks. All variables have default values, which are listed on the following pages. `STRING` variables often have a default value listed as `NULL`, which means no data; this is quite different from the value "", which is a zero-length character string. `long` variables are often used as logical flags, with a zero value indicating false and a non-zero value indicating true.

On the following pages the reader will find individual descriptions of each of the namelist commands and their variables. Each description contains a sequence of the form

```
&<namelist-name>
    <variable-type> <variable-name> = <default-value>;
    .
    .
    .
&end
```

This summarizes the parameters of the namelist. Note, however, that the namelists are invoked in the form

```
&<namelist-name>
    [<variable-name> = <value> ,]
    [<array-name>[<index>] = <value> [,<value> ...] ,]
        .
        .
        .
&end
```

The square-brackets enclose an optional component. Not all namelists require variables to be given– the defaults may be sufficient. However, if a variable name is given, it must have a value. Values for `STRING` variables must be enclosed in double quotation marks. Values for `double` variables may be in floating-point, exponential, or integer format (exponential format uses the 'e' character to introduce the exponent).

Array variables take a list of values, with the first value being placed in the slot indicated by the subscript. As in C, the first slot of the array has subscript 0, *not* 1. The namelist processor does not check to ensure that one does not put elements into nonexistent slots beyond the end of the array; doing so may cause the processor to hang up or crash.

Wildcards are allowed in a number of places in `elegant` and the SDDS Toolkit. The wildcard format is very similar to that used in UNIX:

- `*` — stands for any number of characters, including none.
- `?` — stands for any single character.
- `[<list-of-characters>]` — stands for any single character from the list. The list may include ranges, such as `a-z`, which includes all characters between and including 'a' and 'z' in the ASCII character table.

The special characters `*`, `?`, `[`, and `]` are entered literally by preceeding the character by a backslash (e.g., `\*`).

In many places where a filename is required in an `elegant` namelist, the user may supply a so-called "incomplete" filename. An incomplete filename has the sequence "%s" imbedded in it, for which is substituted the "rootname." The rootname is by default the filename (less the extension) of the command (i.e., main input) file. The most common use of this feature is to cause `elegant` to create names for all output files that share a common filename but differ in their extensions. Post-processing can be greatly simplified by adopting this naming convention, particularly if one consistently uses the same extension for the same type of output. Recommended filename extensions are given in the lists below.

Note that this substitution feature is not generally available for input files, though there are some exceptions (e.g., load_parameters). Another convenience for input file organization is the search-path feature, which can be set from the run_setup command. By default, elegant assumes input filenames give the full pathname. If the search path is specified, elegant will instead look for files in one of the listed directories.

When elegant reads a namelist command, one of its first actions is to print the namelist back to the standard output. This printout includes all the variables in the namelist and their values. Occasionally, the user may see a variable listed in the printout that is not in this manual. These are often obsolete and are retained only for backward compatibility, or else associated with a feature that is not fully supported. Use of such "undocumented features" is discouraged.

elegant supports substitution of fields in namelists using the commandline macro option. This permits making runs with altered parameters without editing the input file. Macros inside the input file have one of two forms: <tag> or \$tag. To perform substitution, use the syntax

```
elegant inputfile|-pipe=in -macro=tag1=value1[,tag2=value2...]
```

When using this feature, it is important to substitute the value of rootname (in run_setup) so that one can get a new set of output files (assuming use of the suggested "%s" field in all the output file names). One may give the macro option any number of times, or combine all substitutions in one option. The name of the input file is available using the macro INPUTFILENAME.

elegant also allows execution of commands in the shell as part of evaluation of a namelist field. To invoke this, one encloses the commandline string in curly braces. E.g.,

```
betax = "{sdds2stream -parameter=betaxFinal data.twi}"
```

(Note that the quotes are also required.) In this example, betax is assigned the value of the parameter betaxFinal from the file data.twi.

It is also possible to perform calculations using elegant's built-in RPN calculator. (It is identical to the commandline programs rpn and rpnl supplied with the SDDS toolkit.) To do this in the command file, one must use quotation marks and enclose the expression in parentheses, as in

```
betax = "(8 2 / pi /)"
```

(Note that this is different from using such expressions in the lattice file; in that case, one doesn't need the parentheses.) One can not only make such computations, but also use the stack and variables. So, for example, one might use

```
betax = "(8 2 / pi / sto betax0)"
betay = "(betax0)"
```

One can also mix subcommands and RPN expressions, as in

```
betax = "({sdds2stream -parameter=betaxFinal data.twi} 2 /)"
```

would assign to betax half the value of the parameter betaxFinal from the file data.twi.

7.3 Setup and Action Commands

A subject of frequent confusion for elegant users is the distinction between setup and action commands. An "action" command causes elegant to immediately perform a specific computation or set of computations. In contrast, a "setup" command tells elegant how to perform computations when it later encounters a "major" action command (one of analyze_map, find_aperture, frequency_map, momentum_aperture, optimize, or track). (N.B.: After each major action command, the problem space is wiped clear. To peform further computations requires introduction of a new run_setup command.)

Several commands are switchable between action and setup modes. These include the coupled_twiss_output, correction_matrix_output, twiss_output, find_aperture, matrix_output, and sasefel commands. Except for find_aperture, all of the commands that can run in both modes have the output_at_each_step parameter, which is used to switch between the modes. In the case of find_aperture, the switch is accomplished using the optimization_mode parameter. Regardless of which parameter is present, unless the parameter is given a value of 1, the command operates in action mode. Further, if the command is used in setup mode and no relevant action command is present later in the file, then the requested will not be performed.

Typically one wants to use these switchable commands in setup mode whenever one is simulating random errors, performing a parameter scan, or performing optimization. When in setup mode, the indicated computations will be performed repeatedly, e.g., for each set of errors, for each step in the parameter scan, or for use in each evaluation of the optimization penalty function.

7.4 Table of elegant commands and their functions

Command name	Type	Description
alter_elements	action	Change an element parameter from the command file.
amplification_factors	action	Compute orbit amplification functions.
analyze_map	major action	Determine first-order matrix from tracking.
aperture_data	setup	Define aperture using an SDDS file.
bunched_beam	setup	Set up beam generation.
bunched_beam_moments	setup	Set up beam generation.
change_particle	action	Change the type of particle. Default is electron.
chaos_map	action	Compute a map of the degree of chaos in particle motion.
chromaticity	setup	Correct the chromaticity.
closed_orbit	setup	Compute the closed orbit.
correct	setup	Correct the orbit or trajectory.
correction_matrix_output	action/setup	Obtain orbit/trajectory correction matrix in a file.
correct_tunes	setup	Correct the tunes.
coupled_twiss_output	setup/action	Compute and output coupled twiss parameters.
divide_elements	setup	Specify division of elements into pieces.
elastic_scattering	major action	Use tracking to determine local scattering aperture and loss locations due to elastic gas scattering.
error_element	setup	Define errors for a set of elements.
error_control	setup	Set up and control error generation process.
find_aperture	setup/major action	Determine the transverse (e.g., dynamic) aperture.
floor_coordinates	action	Compute and output floor coordinates.
frequency_map	major action	Compute and output frequency map.
global_settings	action	Change global settings.
include_commands	action	Read commands from another file.
ignore_elements	setup	Ignore specified elements during tracking.
inelastic_scattering	major action	Use tracking to determine local scattering aperture and loss locations due to inelastic gas scattering.
insert_elements	action	Insert elements into the lattice at many places.
insert_sceffects	action	Insert space charge kick elements.
linear_chromatic_tracking_setup	setup	Set up for fast tracking with chromatic effects.

link_control	setup	Control linking of element parameters.
link_elements	setup	Define link between parameters of two elements.
load_parameters	setup/action	Load element parameters from SDDS file.
matrix_output	setup/action	Output transfer matrix along beamline.
modulate_elements	setup	Set up time-dependent modulation of elements.
moments_output	setup/action	Compute coupled beam moments, with radiation option.
momentum_aperture	major action	Determine s-dependent momentum aperture.
optimize	major action	Execute an optimization.
optimization_covariable	setup	Define a dependent parameter for optimization.
optimization_setup	setup	Perform initial optimization setup.
optimization_term	setup	Define a term of penalty function.
optimization_variable	setup	Define an optimization variable.
parallel_optimization_setup	setup	Perform initial parallel optimization setup.
print_dictionary	action	Print the element dictionary.
ramp_elements	setup	Set up turn-by-turn ramping of elements.
rf_setup	setup/action	Set up RF cavity elements for storage rings.
rpn_expression	action	Execute an expression in the rpn interpreter.
rpn_load	action	Load values from SDDS file into rpn interpreter.
run_control	setup	Set up simulation steps and passes.
run_setup	setup	Define global simulation parameters and output files.
sasefel	setup/action	Evaluate SASE FEL gain etc.
save_lattice	action	Save new lattice file.
sdds_beam	setup	Define loading of particles from SDDS file.
semaphores	setup	Define file semaphores for start/end of run.
set_reference_particle_output	setup	Define reference particle distribution for optimization
slice_analysis	setup	Perform slice analysis along beamline.
subprocess	action	Execute a command in the shell.
steering_element	setup	Define element parameters as steering correctors.
transmute_elements	setup	Transmute elements from one type to another.
tune_footprint	setup/action	Compute and optimize chromatic and amplitude tune footprints.

twiss_analysis	setup	Define subset of beamline for twiss parameter analysis.
twiss_output	setup/action	Set up twiss parameter and related computation.
track	major action	Execute tracking of particles and other operations.
tune_shift_with_amplitude	setup	Compute tune shifts with amplitude.
vary_element	setup	Vary element parameters in loops.

Table 1: Table of elegant commands and their functions.

7.5 alter_elements

- type: action command.
- function: modify the value of a parameter for one or more elements
- sequence: must follow `run_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&alter_elements
        STRING name = NULL;
        STRING item = NULL;
        STRING type = NULL;
        STRING exclude = NULL;
        double value = 0;
        STRING string_value = NULL;
        long differential = 0;
        long multiplicative = 0;
        long alter_at_each_step = 0;
        long alter_before_load_parameters = 0;
        long verbose = 0;
        long allow_missing_elements = 0;
        long allow_missing_parameters = 0;
        long start_occurence = 0;
        long end_occurence = 0;
        double s_start = -1;
        double s_end = -1;
        STRING before = NULL;
        STRING after = NULL;
&end
```

- `name` — A possibly-wildcard-containing string giving the names of the elements to alter. If not specified, then one must specify `type`.
- `item` — The name of the parameter to alter.
- `type` — A possibly-wildcard-containing string giving the names of element *types* to alter. May be specified with `name` or by itself.
- `exclude` — A possibly-wildcard-containing string giving the names of elements to excluded from alteration.
- `value`, `string_value` — The new value for the parameter. Use `string_value` only if the parameter takes a character string as its value.
- `differential` — If nonzero, the new value is the predefined value of the parameter plus the quantity given with `value`.

- `multiplicative` — If nonozero, the new given value is the predefined value of the parameter times the quantity given with `value`.
- `alter_at_each_step` — If nonzero, the changes requested by the command are performed at each simulation step. Note that if `differential` or `multiplicative` are non-zero, then changes will accumulate. (A more conventional way to perform such variation is with `vary_elements`.)
- `alter_before_load_parameters` — If `alter_at_each_step`, by default the alteration takes place after any `load_parameters` commands are processed. If this control is non-zero, the alteration takes place before any `load_parameters` commands are processed.
- `verbose` — If nonzero, information is printed to the standard output describing what elements are changed.
- `allow_missing_elements` — If nonzero, then it is not an error if an element matching `name` does not exist. Normally, such an occurence is an error and terminates the program.
- `allow_missing_parameters` — If nonzero, then it is not an error if an element does not have the parameter named with `item`. Normally, such an occurence is an error and terminates the program.
- `start_occurence`, `end_occurence` — If nonzero, these give the starting and ending occurence numbers of elements that will be altered. N.B.: if wildcards are used, occurence number counting is for each set of identically-named elements separately, rather than for the sequence of matched elements.
- `s_start`, `s_end` — If non-negative, these give the starting and ending position limits for the end-of-element locations of elements to be altered.
- `after` — The name of an element. If given, the alteration is applied only to elements that follow the named element in the beamline.
- `before` — The name of an element. If given, the alteration is applied only to elements that precede the named element in the beamline.

7.6 amplification_factors

- type: action command.
- function: compute corrected and uncorrected orbit amplification factors and functions.
- sequence: must be the last command in a sequence.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&amplification_factors
    STRING output = NULL;
    STRING uncorrected_orbit_function = NULL;
    STRING corrected_orbit_function = NULL;
    STRING kick_function = NULL;
    STRING name = NULL;
    STRING type = NULL;
    STRING item = NULL;
    STRING plane = NULL;
    double change = 1e-3;
    long number_to_do = -1;
    double maximum_z = 0;
&end
```

- `output` — The (incomplete) name of a file for text output. Recommended value: “%s.af”.
- `uncorrected_orbit_function` — The (incomplete) name of a file for an SDDS-format output of the uncorrected-orbit amplification function. Recommended value: “%s.uof”.
- `corrected_orbit_function` — The (incomplete) name of a file for an SDDS-format output of the corrected-orbit amplification function. Recommended value: “%s.cof”.
- `kick_function` — The (incomplete) name of a file for an SDDS-format output of the kick amplification function. Recommended value: “%s.kaf”.
- `name` — The optionally wildcarded name of the orbit-perturbing elements.
- `type` — The optional type name of the the orbit-perturbing elements.
- `item` — The parameter of the elements producing the orbit.
- `plane` — The plane (“h” or “v”) to examine.
- `change` — The parameter change to use in computing the amplification.
- `number_to_do` — The number of elements to perturb.
- `maximum_z` — The maximum z coordinate of the elements to perturb.

7.7 analyze_map

- type: major action command.
- function: find the transport matrix up to third order based on particle tracking, based on method described in [4]. Also find related quantities, such as chromaticity.
- sequence: must follow `run_control`.
- can use parallel resources (`Pelegant`)
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&analyze_map
    STRING output = NULL;
    long output_order = 1;
    STRING printout = NULL;
    STRING printout_format = "%22.15e ";
    long printout_order = 2;
    double delta_x = 5e-5;
    double delta_xp = 5e-5;
    double delta_y = 5e-5;
    double delta_yp = 5e-5;
    double delta_s  = 5e-5;
    double delta_dp = 5e-5;
    double accuracy_factor = 1e-12;
    long center_on_orbit = 0;
    long verbosity = 0;
    long canonical_variables = 0;
    long periodic = 1;
    double beta_x = 1;
    double alpha_x = 0;
    double eta_x = 0;
    double etap_x = 0;
    double beta_y = 1;
    double alpha_y = 0;
    double eta_y = 0;
    double etap_y = 0;
    long n_points = 9;
    long max_fit_order = 8;
&end
```

- `output` — The (incomplete) name of a file for SDDS output.
  - Recommended value: “%s.ana”.
  - File contents: A series of pages, each consisting of a single data point containing the centroid offsets for a single turn, the single-turn R matrix, the matched Twiss parameters, tunes, and dispersion functions.

- `printout` — The (incomplete) name of a file for text output of the matrix.
- `printout_format` — The C-style formatting statement for the matrix elements. A space, comma, or other separator should appear at the end of the string.
- `delta_X` — The amount by which to change the quantity X in computing the derivatives that give the matrix elements.
- `accuracy_factor` — The fraction of the maximum absolute value of the final coordinate that is considered meaningful. Used to estimate errors and eliminate spurious matrix elements.
- `canonical_variables` — If non-zero, the matrix is expressed in terms of canonical variables $(x, q_x, y, q_y, -s, \delta)$ instead of the default $(x, x', y, y', s, \delta)$.
- `center_on_orbit` — A flag directing the expansion to be made about the closed orbit instead of the design orbit.
- `verbosity` — The larger this value, the more output is printed during computations.
- `printout_order` — Order of the matrix to be printed to the `printout` file.
- `periodic` — If non-zero, system is assumed to be periodic and lattice functions, tunes, chromaticities, etc are computed.
- `beta_x`, `alpha_x`, `eta_x`, `etap_x`, `beta_y`, `alpha_y`, `eta_y`, `etap_y` — If `periodic=0`, these are the starting values for the lattice functions.
- `n_points` — Number of points in each phase-space dimension.
- `max_fit_order` — Maximum order of fits using in determining the matrix elements.

7.8 aperture_data

- type: setup command.
- function: specify a file from which to take x and y aperture data vs s.
- note: this command is also available under the name `aperture_input`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&aperture_data
        STRING input = NULL;
        long periodic = 1;
        long persistent = 0;
        long disable = 0;
&end
```

- `input` — Name of SDDS file supplying the aperture data. The following columns are all required, in double or float type, with units of `m` (meters).
  1. `s` — Distance along the central trajectory.
  2. `xHalfAperture` — Half aperture in the horizontal.
  3. `yHalfAperture` — Half aperture in the vertical.
  4. `xCenter` — Center of the aperture in the horizontal.
  5. `yCenter` — Center of the aperture in the vertical.
- `periodic` — If non-zero, the aperture is a periodic function of `s`, with period equal to the range of the data.
- `persistent` — If non-zero, the aperture data persists across subsequent `run_setup` commands. By default, the aperture data is forgotten when a new `run_setup` command is seen.
- `disable` — If non-zero, the command is ignored.

7.9 bunched_beam

- type: setup command.
- sequence: must follow `run_control`.
- function: set up for tracking of particle coordinates with various distributions.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- Notes:
  - In `Pelegant`, the exact particles generated will change as the number of cores is changed.
  - This command is used when it is convenient to specify the beam dimensions in terms of lattice functions and emittances. The `bunched_beam_moments` command can be used when it is more convenient to specify beam sizes, divergences, etc.

```
&bunched_beam
    STRING bunch = NULL;
    long n_particles_per_bunch = 1;
    long multiply_np_by_cores = 0;
    double time_start = 0;
    STRING matched_to_cell = NULL;
    double emit_x  = 0;
    double emit_nx  = 0;
    double beta_x  = 1.0;
    double alpha_x = 0.0;
    double eta_x   = 0.0;
    double etap_x  = 0.0;
    double emit_y  = 0;
    double emit_ny  = 0;
    double beta_y  = 1.0;
    double alpha_y = 0.0;
    double eta_y   = 0.0;
    double etap_y  = 0.0;
    long use_twiss_command_values = 0;
    long use_moments_output_values = 0;
    double Po = 0.0;
    double sigma_dp = 0.0;
    double sigma_s = 0.0;
    double dp_s_coupling = 0;
    double emit_z = 0;
    double beta_z = 0;
    double alpha_z = 0;
    double momentum_chirp = 0;
    long one_random_bunch = 1;
    long symmetrize = 0;
    long halton_sequence[3] = {0, 0, 0};
```

```
    long halton_radix[6] = {0, 0, 0, 0, 0, 0};
    long optimized_halton = 0;
    long randomize_order[3] = {0, 0, 0};
    long limit_invariants = 0;
    long limit_in_4d = 0;
    long enforce_rms_values[3] = {0, 0, 0};
    double distribution_cutoff[3] = {2, 2, 2};
    STRING distribution_type[3] = {"gaussian","gaussian","gaussian"};
    double centroid[6] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
    long first_is_fiducial = 0;
    long save_initial_coordinates = 1;
&end
```

- `bunch` — The (incomplete) name of an SDDS file to which the phase-space coordinates of the bunches are to be written. Recommended value: "%s.bun".
- `n_particles_per_bunch` — Number of particles in each bunch.
- `multiply_np_by_cores` — If non-zero, the number of particles is multiplied by the number of working cores.
- `time_start` — The central value of the time coordinate for the bunch.
- `matched_to_cell` — The name of a beamline from which the Twiss parameters of the bunch are to be computed.
- `emit_X` — RMS emittance for the X plane.
- `emit_nX` — RMS normalized emittance for the X plane. Ignored if `emit_X` is nonzero.
- `beta_X`, `alpha_X`, `eta_X`, `etap_X` — Twiss parameters for the X plane.
- `use_twiss_command_values` — If nonzero, then the values for β, α, η, and η' are taken from the `twiss_output` command. It is an error if no `twiss_output` command has been given.
- `use_moments_output_values` — If nonzero, then the beam is generated to match the 6D matched, equilibrium beam moments computed by the `moments_output` command. The distribution type must be gaussian. This mode is incompatible with using closed orbit correction with `start_from_centroid=1` (the default value).
- `Po` — Central momentum of the bunch.
- `sigma_dp`, `sigma_s` — Fractional momentum spread, ffi, and bunch length. Note that `sigma_s` is actually the length in $\beta_z * c * t$, so that for $\beta_z << 1$ the length of the bunch in time will be greater than one might expect.
- `dp_s_coupling` — Specifies the coupling between s and ffi, defined as ⟨sffi⟩/(œ$_{s}$œ$_{ffi}$).
- `emit_z`, `beta_z`, `alpha_z` — Provide another way to specify the longitudinal phase space, either separately from or in combination with `sigma_dp`, `sigma_s`, and `dp_s_coupling`.

  Basically, which values `elegant` uses depends on what one sets to nonzero values. If one sets emit_z, then sigma_dp, sigma_s, and dp_s_coupling are ignored. If one doesn't set emit_z, then elegant uses sigma_dp and sigma_s; it additionally uses alpha_z if it is nonzero, otherwise it uses dp_s_coupling. For reference, the relationship between them is $C = \frac{\Sigma_{56}}{\sqrt{\Sigma_{55}\Sigma_{66}}} = -\frac{\alpha}{\sqrt{1+\alpha^2}}$. Note that to impart a chirp that results in compression for $R_{56} < 0$ (e.g., a normal four-dipole chicane), one must have $\alpha_z < 0$ or $C > 0$.

- `momentum_chirp` — Permits imparting an additional momentum chirp to the beam, in units of 1/m. E.g., a value of 1 indicates that a 1mm long bunch has a linear variation in momentum of 0.1% from end-to-end. A positive chirp is needed to provide compression of a bunch with an ordinary $R_{56} < 0$ four-dipole chicane.

- `one_random_bunch` — If non-zero, then only one random particle distribution is generated. Otherwise, a new distribution will be generated for every simulation step.

- `enforce_rms_values[3]` — Flags, one for each plane, indicating whether to force the distribution to have the specified RMS properties.

- `distribution_cutoff[3]` — Distribution cutoff parameters for each plane. For gaussian distributions, this is the number of sigmas to use. For other distributions (except dynamic aperture), this number simply multiplies the sizes. This is potentially confusing and hence it is suggested that the distribution cutoff be set to 1 for nongaussian beams.

  The exception is "dynamic-aperture" distribution type. In this case, the cutoff value is the number of grid points in the dimension in question.

- `distribution_type[3]` — Distribution type for each plane. May be "gaussian", "hard-edge", "uniform-ellipse", "shell", "dynamic-aperture", "line", "halo(gaussian)".

  For the transverse plane, the interpretation of the emittance is different for the different beam types. For gaussian beams, the emittances are rms values. For all other types, $\sqrt{\epsilon * \beta}$ times the distribution cutoff defines the edge of the beam in position space, while $\sqrt{\epsilon * (1 + \alpha^2)/\beta}$ times the distribution cutoff defines the edge of the beam in slope space.

  A hard-edge beam is a uniformly-filled parallelogram in phase space. A uniform-ellipse beam is a uniformly-filled ellipse in phase space. A shell beam is a hollow ellipse in phase space. A dynamic aperture beam has zero slope and uniform spacing in position coordinates. A line beam is a line in phase space. A "halo(gaussian)" beam is the part of the gaussian distribution *beyond* the distribution cutoff.

- `limit_invariants` — If non-zero, the distribution cutoffs are applied to the invariants, rather than to the coordinates. This is useful for gaussian beams when the distribution cutoff is small.

- `limit_in_4d` — If non-zero, then the transverse distribution is taken to be a 4-d gaussian or uniform distribution. One of these must be chosen using the `distribution_type` control. It must be the same for x and y. This is useful, for example, if you want to make a cylindrically symmetric beam.

- `symmetrize` — If non-zero, the distribution is symmetric under changes of sign in the coordinates. Automatically results in a zero centroid for all coordinates.

- `halton_sequence[3]` and `halton_radix[6]` and `optimized_halton` — This provides a "quiet-start" feature by choosing Halton sequences in place of random number generation. There are three new variables that control this feature. `halton_sequence` is an array of three flags that permit turning on Halton sequence generation for the horizontal, vertical, or longitudinal planes. For example, `halton_sequence[0] = 3*1` will turn on Halton sequences for all three planes, while `halton_sequence[2] = 1`, will turn it on for the longitudinal plane only.

`halton_radix` is an array of six integers that permit giving the radix for each sequence (i.e., x, x', y, y', t, p). Each radix must be a prime number. One should never use the same prime for two sequences, unless one randomizes the order of the sequences relative to each other (see the next item). If these are left at zero, then elegant chooses values that eliminate phase-space banding to some extent. The user is cautioned to plot all coordinate combinations for the initial phase space to ensure that no unacceptable banding is present.

A suggested way to use Halton sequences is to set `halton_radix[0] = 2, 3, 2, 3, 2, 3` and to set `randomize_order[0] = 2, 2, 2,`. This avoids banding that may result from choosing larger radix values.

`optimized_halton` uses the improved halton sequence [33]. (Algorithm 659, Collected Algorithm from ACM. Derandom Algorithm is added by Hongmei CHI (CS/FSU)). It avoids the banding problem automatically and the `halton_radix` values are ignored.

- `randomize_order[3]` — Allows randomizing the order of assigned coordinates for the pairs (x, x'), (y, y'), and (t,p). 0 means no randomization; 1 means randomize (x, x', y, y', t, p) values independently, which destroys any x-x', y-y', and t-p correlations; 2 means randomize (x, x'), (y, y'), and (t, p) in pair-wise fashion. This is used with Halton sequences to remove banding. It is suggested that that the user employ `sddsanalyzebeam` to verify that the beam properties when randomization is used.

- `centroid[6]` — Centroid offsets for each of the six coordinates.

- `first_is_fiducial` — Specifies that the first beam generated shall be a single particle beam, which is suitable for fiducialization. See the section on "Fiducialization in `elegant`" for more discussion.

- `save_initial_coordinates` — A flag that, if set, results in saving initial coordinates of tracked particles in memory. This is the default behavior. If unset, the initial coordinates are not saved, but are regenerated each time they are needed. This is more memory efficient and is useful for tracking very large numbers of particles.

7.10 bunched_beam_moments

- type: setup command.
- sequence: must follow run_control.
- function: set up for tracking of particle coordinates with various distributions.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- Notes:
  - In Pelegant, the exact particles generated will change as the number of cores is changed.
  - This command is used when it is convenient to specify the beam dimensions in terms of beam sizes, divergences, and other moments. The bunched_beam command can be used when it is more convenient to specify lattice functions, emittances, etc.

```
&bunched_beam_moments
    STRING bunch = NULL;
    long n_particles_per_bunch = 1;
    long multiply_np_by_cores = 0;
    long use_moments_output_values = 0;
    double S1_beta = 0;
    double S2_beta = 0;
    double S12_beta = 0;
    double S16 = 0;
    double S26 = 0;
    double S3_beta = 0;
    double S4_beta = 0;
    double S34_beta = 0;
    double S36 = 0;
    double S46 = 0;
    double S5 = 0;
    double S6 = 0;
    double S56 = 0;
    double time_start = 0;
    double Po = 0.0;
    long one_random_bunch = 1;
    long save_initial_coordinates = 1;
    long limit_invariants = 0;
    long symmetrize = 0;
    long halton_sequence[3] = {0, 0, 0};
    int32_t halton_radix[6] = {0, 0, 0, 0, 0, 0};
    long optimized_halton = 0;
    long randomize_order[3] = {0, 0, 0};
    long limit_in_4d = 0;
    long enforce_rms_values[3] = {0, 0, 0};
    double distribution_cutoff[3] = {2, 2, 2};
```

```
    STRING distribution_type[3] = {"gaussian","gaussian","gaussian"};
    double centroid[6] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
    long first_is_fiducial = 0;
&end
```

- `bunch` — The (incomplete) name of an SDDS file to which the phase-space coordinates of the bunches are to be written. Recommended value: "%s.bun".
- `n_particles_per_bunch` — Number of particles in each bunch.
- `multiply_np_by_cores` — If non-zero, the number of particles is multiplied by the number of working cores.
- `time_start` — The central value of the time coordinate for the bunch.
- `matched_to_cell` — The name of a beamline from which the Twiss parameters of the bunch are to be computed.
- `use_moments_output_values` — If nonzero, then the beam is generated to match the 6D matched, equilibrium beam moments computed by the `moments_output` command. The distribution type must be gaussian. This mode is incompatible with using closed orbit correction with `start_from_centroid=1` (the default value).
- `Po` — Central momentum of the bunch.
- `S1_beta`, `S2_beta`, `S3_beta`, `S4_beta` — Horizontal beam size and divergence, vertical beam size and divergence, for betatron coordinates. For example, for the x coordinate, we have

  $$x = x_\beta + \delta\eta_x \tag{3}$$

  where x_β is the betatron component, δ is the fractional momentum deviation, and $\eta_x = \Sigma_{16}/\Sigma_{66}$.
- `S5` — Fractional energy spread.
- `S6` — Fractional bunch length.
- `S56` — Σ_{56}.
- S*ij*`_beta` — Element of the beam sigma matrix for betatron coordinates.
- `one_random_bunch` — If non-zero, then only one random particle distribution is generated. Otherwise, a new distribution will be generated for every simulation step.
- `enforce_rms_values[3]` — Flags, one for each plane, indicating whether to force the distribution to have the specified RMS properties.
- `distribution_cutoff[3]` — Distribution cutoff parameters for each plane. For gaussian distributions, this is the number of sigmas to use. For other distributions (except dynamic aperture), this number simply multiplies the sizes. This is potentially confusing and hence it is suggested that the distribution cutoff be set to 1 for nongaussian beams.

  The exception is "dynamic-aperture" distribution type. In this case, the cutoff value is the number of grid points in the dimension in question.

- `distribution_type[3]` — Distribution type for each plane. May be "gaussian", "hard-edge", "uniform-ellipse", "shell", "dynamic-aperture", "line", "halo(gaussian)".

  For the transverse plane, the interpretation of the emittance is different for the different beam types. For gaussian beams, the emittances are rms values. For all other types, $\sqrt{\epsilon * \beta}$ times the distribution cutoff defines the edge of the beam in position space, while $\sqrt{\epsilon * (1+\alpha^2)/\beta}$ times the distribution cutoff defines the edge of the beam in slope space.

  A hard-edge beam is a uniformly-filled parallelogram in phase space. A uniform-ellipse beam is a uniformly-filled ellipse in phase space. A shell beam is a hollow ellipse in phase space. A dynamic aperture beam has zero slope and uniform spacing in position coordinates. A line beam is a line in phase space. A "halo(gaussian)" beam is the part of the gaussian distribution *beyond* the distribution cutoff.

- `limit_invariants` — If non-zero, the distribution cutoffs are applied to the invariants, rather than to the coordinates. This is useful for gaussian beams when the distribution cutoff is small.

- `limit_in_4d` — If non-zero, then the transverse distribution is taken to be a 4-d gaussian or uniform distribution. One of these must be chosen using the `distribution_type` control. It must be the same for x and y. This is useful, for example, if you want to make a cylindrically symmetric beam.

- `symmetrize` — If non-zero, the distribution is symmetric under changes of sign in the coordinates. Automatically results in a zero centroid for all coordinates.

- `halton_sequence[3]` and `halton_radix[6]` and `optimized_halton` — This provides a "quiet-start" feature by choosing Halton sequences in place of random number generation. There are three new variables that control this feature. `halton_sequence` is an array of three flags that permit turning on Halton sequence generation for the horizontal, vertical, or longitudinal planes. For example, `halton_sequence[0] = 3*1` will turn on Halton sequences for all three planes, while `halton_sequence[2] = 1`, will turn it on for the longitudinal plane only.

  `halton_radix` is an array of six integers that permit giving the radix for each sequence (i.e., x, x', y, y', t, p). Each radix must be a prime number. One should never use the same prime for two sequences, unless one randomizes the order of the sequences relative to each other (see the next item). If these are left at zero, then elegant chooses values that eliminate phase-space banding to some extent. The user is cautioned to plot all coordinate combinations for the initial phase space to ensure that no unacceptable banding is present.

  A suggested way to use Halton sequences is to set `halton_radix[0] = 2, 3, 2, 3, 2, 3` and to set `randomize_order[0] = 2, 2, 2,`. This avoids banding that may result from choosing larger radix values.

  `optimized_halton` uses the improved halton sequence [33]. (Algorithm 659, Collected Algorithm from ACM. Derandom Algorithm is added by Hongmei CHI (CS/FSU)). It avoids the banding problem automatically and the `halton_radix` values are ignored.

- `randomize_order[3]` — Allows randomizing the order of assigned coordinates for the pairs (x, x'), (y, y'), and (t,p). 0 means no randomization; 1 means randomize (x, x', y, y', t, p) values independently, which destroys any x-x', y-y', and t-p correlations; 2 means randomize (x, x'), (y, y'), and (t, p) in pair-wise fashion. This is used with Halton sequences to remove banding. It is suggested that that the user employ `sddsanalyzebeam` to verify that the beam properties when randomization is used.

- `centroid[6]` — Centroid offsets for each of the six coordinates.
- `first_is_fiducial` — Specifies that the first beam generated shall be a single particle beam, which is suitable for fiducialization. See the section on "Fiducialization in `elegant`" for more discussion.
- `save_initial_coordinates` — A flag that, if set, results in saving initial coordinates of tracked particles in memory. This is the default behavior. If unset, the initial coordinates are not saved, but are regenerated each time they are needed. This is more memory efficient and is useful for tracking very large numbers of particles.

7.11 change_end

- type: action command.
- function: change the ending point in a lattice
- sequence: must precede `run_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&change_start
        STRING element_name = NULL;
        long element_occurence = -1;
        long delta_position = 0;
&end
```

- `element_name` — Name of the element where the lattice will end, which implies removing all elements downstream of the named element. If the element occurs more than once, the first instance is used, unless `element_occurence` is given. The named element will be the last element in the lattice.
- `element_occurence` — Occurrence number of the element to use. By default, uses the last occurrence.
- `delta_position` — The number of elements before (if negative) or after (if positive) by which to offset the end position.

7.12 change_particle

- type: action command.
- function: change the particle type from the default value of "electron."
- sequence: must precede `run_setup`.
- N.B.: this feature has had limited testing, mostly to verify that electron tracking is not impacted by the implementation. Please use with caution and be alert for suspicious results.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&change_particle
    STRING name = "electron";
    double mass_ratio = 0;
    double charge_ratio = 0;
&end
```

- `name` — The name of the particle to use. Possible values are `electron`, `positron`, `proton`, `muon`, and `custom`.
- `mass_ratio`, `charge_ratio` — If the particle name is "custom," these parameters specify the mass and charge of the particle relative to the electron. E.g., for an anti-proton, one would use a mass ratio of 1836.18 and a charge ratio of 1.

7.13 change_start

- type: action command.
- function: change the starting point in a lattice
- sequence: must precede `run_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&change_start
        STRING element_name = NULL;
        long element_occurence = 1;
        long ring_mode = 0;
        long delta_position = 0;
&end
```

- `element_name` — Name of the element where the lattice will start, which implies removing all elements upstream of the named element. If the element occurs more than once, the first instance is used. The named element will be the first element in the lattice.
- `element_occurence` — Occurrence number of the element to use.
- `ring_mode` — If nonzero, the ring structure of the lattice is preserved by moving the elements upstream of the named element to the end of the lattice.
- `delta_position` — The number of elements before (if negative) or after (if positive) by which to offset the end position.

7.14 chaos_map

- type: major action command.
- function: compute chaos map from tracking. Note that the number of turns tracked is set by the `run_control` command.
- can use parallel resources (`Pelegant`)
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- NB: this feature is new in 2019.4 and somewhat experimental. Please report problems on the forum.

```
&chaos_map
    STRING output = NULL;
    double xmin =  -0.1;
    double xmax =  0.1;
    double ymin =  1e-6;
    double ymax =  0.1;
    double delta_min = 0;
    double delta_max = 0;
    long nx = 20;
    long ny = 21;
    long ndelta = 1;
    long forward_backward = 0;
    double change_x = 1e-6;
    double change_y = 1e-6;
    long verbosity = 1;
&end
```

- `output` — The (incomplete) name of an SDDS file to send output to. Recommended value: "%s.cmap". For the parallel version, particles will be listed in essentially random order. If needed, `sddssort` can be used to sort particles by initial coordinates.
- `xmin`, `xmax` — Limits of grid of initial x coordinates for tracking.
- `ymin`, `ymax` — Limits of grid of initial y coordinates for tracking. `ymin` should be a small, positive value so that there is some betatron oscillation from which to get the tune.
- `delta_min`, `delta_max` — Limits of grid of initial δ coordinates for tracking. Note that particles are not centered around the dispersive closed orbit. Hence, the tracking is appropriate to simulation of dynamics from a touschek scattering event.
- `nx` — Number of values of x coordinate in the grid.
- `ny` — Number of values of y coordinate in the grid.
- `ndelta` — Number of values of δ coordinate in the grid.

- `forward_backward` — If non-zero, uses the forward/backward integration technique of Y. Li *et al.* [56]. The number of passes tracked is still controlled by the `n_passes` parameter of `run_control`. In addition, the number of iterations of forward and backward tracking is given by the value of `forward_backward`. If zero, a less interesting technique is used that computes the change in J_x and J_y from tracking with small changes in initial conditions.

- `change_x`, `change_y` — If `forward_backward` is zero, gives the perturbation to initial x and y used to assess chaotic motion from divergence of trajectories.

- `verbosity` — If nonzero, prints possibly useful information while running.

7.15 chromaticity

- type: setup command.
- function: set up for chromaticity correction.
- sequence: should follow `twiss_output`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&chromaticity
    STRING sextupoles = NULL;
    STRING lower_limits = NULL;
    STRING upper_limits = NULL;
    STRING exclude = NULL;
    double dnux_dp = 0;
    double dnuy_dp = 0;
    double sextupole_tweek = 1e-3;
    double correction_fraction = 0.9;
    long n_iterations = 5;
    double tolerance = 0;
    STRING strength_log = NULL;
    long change_defined_values = 0;
    double strength_limit = 0;
    long use_perturbed_matrix = 0;
    long exit_on_failure = 0;
    long update_orbit = 0;
    long verbosity = 1;
    double dK2_weight = 1;
    STRING response_matrix_output = NULL;
    STRING correction_matrix_output = NULL;
    long fse_units = 0;
&end
```

- `sextupoles` — List of names of elements to use to correct the chromaticities. Several names may be given and names may include wildcards. If so, then sextupoles in each group are changed by the same amount for each iteration. This would typically be used when the sextupoles are nominally identical (though perhaps differing in strength because of introduced errors). If that's not the case, the iteration may fail to converge.
- `lower_limits`, `upper_limits` — Lists of lower and upper limits for each family. The input style is unusual for `elegant`, given that lists of numbers are to be provided in a string. For example

  ```
  &chromaticity
   sextupoles = "S1 S2 S3",
   lower_limits = "0 -1.5 0",
   upper_limits = "1.75 0 1.25"
  ```

```
  ...
&end
```

- `strength_limit` — Deprecated. Limit on the absolute value of sextupole strength (K_2).
- `exclude` — List of names of elements to exclude. This may be used to exclude some sextupoles that are matched by wildcards in the `sextupole` list.
- `dK2_weight` — Weighting factor that is used to minimize the mean-square changes in K_2 values in the event that there are more than two families.
- `dnux_dp`, `dnuy_dp` — Desired chromaticity values.
- `sextupole_tweek` — Amount by which to tweak the sextupoles to compute derivatives of chromaticities with respect to sextupole strength. [The word "tweak" is misspelled "tweek" in the code.]
- `correction_fraction` — Fraction of the correction to apply at each iteration. In some cases, correction is unstable at this number should be reduced.
- `n_iterations` — Number of iterations of the correction to perform.
- `tolerance` — Stop iterating when chromaticities are within this value of the desired values.
- `strength_log` — The (incomplete) name of an SDDS file to which the sextupole strengths will be written. Recommended value: "%s.ssl". May be used with `load_parameters`.
- `change_defined_values` — Changes the defined values of the sextupole strengths. This means that when the lattice is saved (using `save_lattice`), the sextupoles will have the corrected values. This would be used for correcting the chromaticity of a design lattice, for example, but not for correcting chromaticity of a perturbed lattice.
- `use_perturbed_matrix` — If nonzero, requests use of the perturbed correction matrix in performing correction. For difficult lattices with large errors, this may be necessary to obtain correction. In general, it is not necessary and only slows the simulation.
- `exit_on_failure` — If nonzero, then failure to reach the desired chromaticities within the tolerance results in the program exiting.
- `update_orbit` — If non-zero, the orbit calculation is updated after each n^{th} adjustment of the sextupoles. If this is needed, it may also help to use reduce the `n_iterations` parameter and set the `correction_iterations` parameter of `run_setup` to a larger value; this will cause `elegant` to cycle through orbit, tune, and chromaticity correction multiple times, which will help converge to a fully-corrected result.
- `verbosity` — Increasing positive values result in increasing amounts of information printed during execution.
- `response_matrix_output`, `correction_matrix_output` — The (incomplete) names of SDDS files to which the response and correction matrices will be written.
- `fse_units` — If non-zero, then the data in the `response_matrix_output` and `correction_matrix_output` files is in units of fractional strength error rather than K_2L.

7.16 closed_orbit

- type: setup/action command.
- function: set up for computation of the closed orbit.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&closed_orbit
    STRING output = NULL;
    long output_monitors_only = 0;
    long start_from_centroid = 1;
    long start_from_dp_centroid = 0;
    double closed_orbit_accuracy = 1e-12;
    double closed_orbit_accuracy_requirement = 1e-7;
    long closed_orbit_iterations = 40;
    long fixed_length = 0;
    long start_from_recirc = 0;
    long verbosity = 0;
    double iteration_fraction = 0.9;
    double fraction_multiplier = 1.05;
    double multiplier_interval = 5;
    long output_monitors_only = 0;
    long tracking_turns = 0;
    long disable = 0;
    long immediate = 0;
&end
```

- `output` — The (incomplete) name of an SDDS file to which the closed orbits will be written. Recommended value: "%s.clo".
- `output_monitors_only` — If non-zero, indicates that the closed orbit output should include only the data at the locations of the beam-position monitors.
- `start_from_centroid` — A flag indicating whether to force the computation to start from the centroids of the beam distribution.
- `start_from_dp_centroid` — A flag indicating whether to force the computation to use the momentum centroid of the beam for the closed orbit. This can allow computing the closed orbit for an off-momentum beam, then starting the beam on that orbit using the `offset_by_orbit` or `center_on_orbit` parameters of the `track` command. In contrast to the `start_from_centroid`, this command doesn't force the algorithm to start from the beam transverse centroids.
- `closed_orbit_accuracy` — The desired accuracy of the closed orbit, in terms of the difference between the start and end coordinates, in meters. Iteration will terminate when this value is achieved.
- `closed_orbit_accuracy_requirement` — The required accuracy of the closed orbit. If not achieved, the closed orbit calculation is considered to have failed.

- closed_orbit_iterations — The number of iterations to take in finding the closed orbit.
- iteration_fraction — Fraction of computed change that is used each iteration. For lattices that are very nonlinear or close to unstable, a number less than 1 can be helpful. Otherwise, it only slows the simulation.
- fixed_length — A flag indicating whether to find a closed orbit with the same length as the design orbit by changing the momentum offset.
- start_from_recirc — A flag indicating whether to compute the closed orbit from the recirculation (recirc) element in the beamline. In general, if one has a recirculation element, one should give this flag.
- verbosity — A larger value results in more printouts during the computations.
- iteration_fraction — Controls the fraction of the update to apply when iterating toward a closed orbit. Smaller numbers give less chance of instability at the price of slower convergence.e
- fraction_multiplier — Multiplier to apply to the iteration fraction if iteration is converging.
- multiplier_interval — Interval in number of iterations at which to adjust the iteration fraction.
- output_monitors_only — If non-zero, output file contains data only at beam position monitors, i.e., at MONI, HMON, and VMON elements.
- tracking_turns — If non-zero, the number of turns to track for detemination of the closed orbit by averaging. This may be useful if the regular closed orbit algorithm complains about convergence issues.
- disable — If non-zero, disables the command.
- immediate — If non-zero, computations are done immediately, so the command acts as an action command.

7.17 correct

- type: setup command.
- sequence: must follow `run_setup` and precede beam definition (`bunched_beam` or `sdds_beam`).
- function: set up for correction of the trajectory or closed orbit.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&correct
    STRING mode = "trajectory";
    STRING method = "global";
    STRING trajectory_output = NULL;
    STRING corrector_output = NULL;
    STRING statistics = NULL;
    STRING bpm_output = NULL;
    double corrector_tweek[2] = {1e-6, 1e-6};
    double corrector_limit[2] = {0, 0};
    double correction_fraction[2] = {1, 1};
    double correction_accuracy[2] = {1e-6, 1e-6};
    long do_correction[2] = {1, 1};
    long remove_smallest_SVs[2] = {0, 0};
    long keep_largest_SVs[2] = {0, 0};
    double minimum_SV_ratio[2] = {0, 0};
    long auto_limit_SVs[2] = {1, 1};
    double Tikhonov_relative_alpha[2] = {0, 0};
    long Tikhonov_n[2] = {-1, -1};
    long removed_pegged[2] = {0, 0};
    long threading_divisor[2] = {100, 100};
    long threading_correctors[2] = {-1, -1};
    double bpm_noise[2] = {0, 0};
    double bpm_noise_cutoff[2] = {1.0, 1.0};
    STRING bpm_noise_distribution[2] = {"uniform", "uniform"};
    long verbose = 1;
    long fixed_length = 0;
    long fixed_length_matrix = 0;
    long n_xy_cycles = 1;
    long minimum_cycles = 1;
    long force_alternation = 0;
    long n_iterations = 1;
    long prezero_correctors = 1;
    long track_before_and_after = 0;
    long start_from_centroid = 1;
    long use_actual_beam = 0;
    double closed_orbit_accuracy = 1e-12;
    double closed_orbit_accuracy_requirement = 1e-7;
```

```
    long closed_orbit_iterations = 40;
    double closed_orbit_iteration_fraction = 0.9;
    double closed_orbit_fraction_multiplier = 1.05;
    double closed_orbit_multiplier_interval = 5;
    double closed_orbit_tracking_turns = 0;
    long use_perturbed_matrix = 0;
    long disable = 0;
    long use_response_from_computed_orbits = 0;
&end
```

In the case of array variables with dimension 2, the first entry is for the horizontal plane and the second is for the vertical plane.

- `mode` — Either "trajectory" or "orbit", indicating correction of a trajectory or a closed orbit.
- `method` — For trajectories, may be "one-to-one", "one-to-best", "one-to-next", "thread", "global", or "coupled". "One-to-one" and "one-to-next" are the same: steering is performed by pairing one corrector with the next downstream BPM. "One-to-best" attempts to find a BPM with a large response to each corrector. "Thread" does corrector sweeps to work the beam through a beamline with apertures; it is quite slow. "Global" simply uses the global response matrix; it is the best choice if the trajectory is not lost on an aperture. "Coupled" is like global, but should be used for strongly-coupled transport lines; in this case, only `HMON` and `VMON` elements are permitted for monitors and only `EHKICK`, `EVKICK`, `HKICK`, and `VKICK` elements are permitted for correctors. For closed orbit, must be "global".
- `trajectory_output` — The (incomplete) name of an SDDS file to which the trajectories or orbits will be written. Recommended value: "%s.traj" or "%s.orb".
- `corrector_output` — The (incomplete) name of an SDDS file to which information about the final corrector strengths will be written. Recommended value: "%s.cor". N.B.: although this file looks as if it can be used with the `load_parameters` command, care must be exercised because the data for the horizontal and vertical planes is on separate pages. Typically, one will need to use `sddscombine -merge=Step ...` in order to place the data from both planes on the same page. Also, be aware that if all correctors have the same name, using `change_defined_values=1` on `load_parameters` will not produce the expected results. See the documentation for `load_parameters` for more details.
- `statistics` — The (incomplete) name of an SDDS file to which statistical information about the trajectories (or orbits) and corrector strengths will be written. Recommended value: "%s.scor".
- `bpm_output` — The (incomplete) name of an SDDS file to which post-correction BPM errors will be written. The errors are the residual after correction, and include the effects of offsets (`DX` and `DY`), setpoints (`XSETPOINT`, `YSETPOINT`, and `SETPOINT`), and tilts (`TILT`). Recommended value: "%s.bpm".
- `corrector_tweek[2]` — The amount by which to change the correctors in order to compute correction coefficients for transport lines. [The word "tweak" is misspelled "tweek" in the code.] The default value, 1 mrad, may be too large for systems with small apertures. If you get an error message about "tracking failed for test particle," try decreasing this value.

- corrector_limit[2] — The maximum strength allowed for a corrector.
- correction_fraction[2] — The fraction of the computed correction strength to actually use for any one iteration.
- correction_accuracy[2] — The desired accuracy of the correction in terms of the RMS BPM values.
- do_correction[2] — Flags to allow disabling correction in one or both planes (if set to zero).
- remove_smallest_SVs, keep_largest_SVs, minimum_SV_ratio, auto_limit_SVs — These parameters control the elimination of singular vectors from the inverse response matrix, which can help deal with degeneracy in the correctors and reduce corrector strength. By default, the number of singular vectors is limited to the number of BPMs, which is a basic condition for stability; this can be defeated by setting auto_limit_SVs to 0 for the desired planes. Set remove_smallest_SVs to require removal of a given number of vectors with the smallest singular values; this is ignored if auto_limit_SVs is also requested and would remove more SVs. Set keep_largest_SVs to require keeping at most a given number of the largest SVs. Set minimum_SV_ratio to require removal of any vectors with singular values less than a given factor of the largest singular value.
- Tikhonov_relative_alpha[2] , Tikhonov_n[2] — Used for invoking Tikhonov regularization of the singular value spectrum prior to creating the inverse matrices. If Tikhonov_relative_alpha is positive, the Tikhonov α parameter is set to the given value times the largest singular value for the plane in question. If Tikhonov_relative_alpha is zero or negative and Tikhonov_n is greater than 0, the Tikhonov α parameter is set to the singular value of the indicated vector; e.g., using 10 means α is equal to the 11^{th} largest singular value (indexing starts at zero). Can be used together with singular-value removal controls.
- remove_pegged[2] — If nonzero, then for the plane in question pegged correctors will be removed from the correction matrix. This results in recomputation of the matrix, following which correction continues with the reduced set of correctors. The pegged corrector is left at its last value.
- threading_divisor — In threading mode trajectory correction, each corrector is varied between 0 and $\pm\theta_{\rm max}$, where $\theta_{\rm max}$ is the strength limit. This parameter sets the number of steps to divide the corrector range into on the positive and negative sides. A smaller value results in faster execution but is less reliable.
- threading_correctors — In threading mode trajectory correction, gives the number of correctors upstream of the loss point to use for threading the beam further through the system.
- bpm_noise[2] — The BPM noise level.
- bpm_noise_cutoff[2] — Cutoff values for the random distributions of BPM noise.
- bpm_noise_distribution[2] — May be either "gaussian", "uniform", or "plus_or_minus".
- verbose — If non-zero, information about the correction is printed during computations.

- `fixed_length` — Indicates that the closed orbit length should be kept the same as the design orbit length by changing the momentum offset of the beam.
- `fixed_length_matrix` — Indicates that for fixed-length orbit correction, the fixed-length matrix should be computed and used. This will improve convergence but isn't always needed.
- `n_xy_cycles` — Number of times to alternate between correcting the x and y planes.
- `force_alternation` — Forces alternation between x and y correction even if one plane appears to have converged.
- `minimum_cycles` — The minimum number of x-y cycles to perform, even if the correction does not improve.
- `n_iterations` — Number of iterations of the correction in each plane for each x/y cycle.
- `prezero_correctors` — Flag indicating whether to set the correctors to zero before starting.
- `track_before_and_after` — Flag indicating whether tracking should be done both before and after correction.
- `start_from_centroid` — Flag indicating that correction should start from the beam centroid. For orbit correction, only the beam momentum centroid is relevant.
- `use_actual_beam` — Flag indicating that correction should employ tracking of the beam distribution rather than a single particle. This is valid for trajectory correction only.
- `closed_orbit_accuracy` — Target accuracy of closed orbit computation.
- `closed_orbit_accuracy_requirement` — Required accuracy of closed orbit computation.
- `closed_orbit_iterations` — Number of iterations of closed orbit computation.
- `closed_orbit_iteration_fraction` — Fraction of change in closed orbit to use at each iteration.
- `closed_orbit_fraction_multiplier` — Multiplier to apply to the iteration fraction if iteration is converging.
- `closed_orbit_multiplier_interval` — Interval in number of iterations at which to adjust the iteration fraction.
- `closed_orbit_tracking_turns` — If non-zero, the absolute value gives the number of turns to track for detemination of the closed orbit by averaging. This may be useful if the regular closed orbit algorithm complains about convergence issues. If less than zero, then *only* this method is used. If greater than zero, then regular orbit determination is tried first, and tracking is used as a fallback.
- `use_perturbed_matrix` — If nonzero, specifies that prior to each correction `elegant` shall recompute the response matrix. This is useful if the lattice is changing significantly between corrections.
- `disable` — If nonzero, the command is ignored.

- `use_response_from_computed_orbits` — If nonzero, in-plane response matrices are computed using differences of closed orbits, which is slower but may be more accurate. For cross-plane matrices, this is always the case.

7.18 correction_matrix_output

- type: setup/action command.
- function: provide output of the orbit/trajectory correction matrix.
- sequence: must follow `run_setup` and definition of steering elements (if wanted, with `steering_element`).
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&correction_matrix_output
    STRING response[4] = NULL, NULL, NULL, NULL;
    STRING inverse[2] = NULL, NULL;
    STRING slope_response[2] = NULL, NULL;
    long full_names = 0;
    long KnL_units = 0;
    long BnL_units = 0;
    long output_at_each_step = 0;
    long output_before_tune_correction = 0;
    long fixed_length = 0;
    long coupled = 0;
    long use_response_from_computed_orbits = 0;
&end
```

- `response` — Array of (incomplete) filenames for SDDS output of the x and y response matrices, plus the cross-plane response matrices. Recommended values, in order: "%s.hrm" (horizontal response to horizontal correctors), "%s.vrm" (vertical response to vertical correctors), "%s.vhrm" (vertical response to horizontal correctors), and "%s.hvrm" (horizontal response to vertical correctors).
- `inverse` — Array of (incomplete) filenames for SDDS output of the x and y inverse response matrices. Recommended values: "%s.hirm" and "%s.virm".
- `slope_response` — Array of (incomplete) filenames for SDDS output of the x' and y' response matrices.
- `full_names` — Flag that, if set, indicates that corrector columns should be named with the element name and item name, instead of just the element name.
- `KnL_units` — Flag that, if set, indicates use of "units" of m/K0L rather than m/rad. This results in a sign change for the horizontal data.
- `BnL_units` — Flag that, if set, indicates use of "units" of m/(T*m) rather than m/rad. This is useful for linac work in that the responses are automatically scaled with beam momentum.
- `output_at_each_step` — Flag that, if set, specifies output of the data at each simulation step. By default, the data is output immediately for the defined lattice.
- `output_before_tune_correction` — Flag that, if set, specifies that when `output_at_each_step` is set, that output shall occur prior to correcting the tunes.

- fixed_length — Flag that, if set, specifies output of the fixed-path-length matrix.
- coupled — If nonzero, the cross-plane response matrices are computed.
- use_response_from_computed_orbits — If nonzero, in-plane response matrices are computed using differences of closed orbits, which is slower but may be more accurate. For cross-plane matrices, this is always the case.

7.19 correct_tunes

- type: setup command.
- function: set up for correction of the tunes.
- sequence: should follow `twiss_output`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&correct_tunes
    STRING quadrupoles = NULL;
    STRING lower_limits = NULL;
    STRING upper_limits = NULL;
    STRING exclude = NULL;
    double tune_x = 0;
    double tune_y = 0;
    long n_iterations = 5;
    double correction_fraction = 0.9;
    double tolerance = 0;
    long step_up_interval = 0;
    double max_correction_fraction = 0.9;
    double delta_correction_fraction = 0.1;
    long update_orbit = 0;
    STRING strength_log = NULL;
    long change_defined_values = 0;
    long use_perturbed_matrix = 0;
    double dK1_weight = 1;
    STRING response_matrix_output = NULL;
    STRING correction_matrix_output = NULL;
    long fse_units = 0;
&end
```

- `quadrupoles` — List of names of quadrupoles to be used. Several names may be given and the names may include wildcards. If so, then quadrupoles in each group are changed by the same amount for each iteration. This would typically be used when the quadrupoles are nominally identical (though perhaps differing in strength because of introduced errors). If that's not the case, the iteration may fail to converge
- `lower_limits`, `upper_limits` — Lists of lower and upper limits for each family. The input style is unusual for `elegant`, given that lists of numbers are to be provided in a string. For example

  ```
  &correct_tunes
   quadrupoles = "Q1 Q2 Q3",
   lower_limits = "0 -1.5 0",
   upper_limits = "1.75 0 1.25"
  ```

```
  ...
&end
```

- `exclude` — List of names of elements to exclude. This may be used to exclude some quadrupoles that are matched by wildcards in the `quadrupoles` list.
- `dK1_weight` — Weighting factor that is used to minimize the mean-square changes in K_1 values in the event that there are more than two families.
- `tune_x`, `tune_y` — Desired x and y tune values. If not given, the desired values are assumed to be the unperturbed tunes.
- `n_iterations` — The number of iterations of the correction to perform.
- `correction_fraction` — The fraction of the correction to apply at each iteration.
- `tolerance` — When both tunes are within this value of the desired tunes, the iteration is stopped.
- `step_up_interval` — Interval between increases in the correction fraction.
- `max_correction_fraction` — Maximum correction fraction to allow.
- `delta_correction_fraction` — Change in correction fraction after each `step_up_interval` steps.
- `update_orbit` — If non-zero, the orbit calculation is updated after each n^{th} adjustment of the quadupoles.
- `strength_log` — The (incomplete) name of a SDDS file to which the quadrupole strengths will be written as correction proceeds. Recommended value: "%s.qst". May be used with `load_parameters`.
- `change_defined_values` — Changes the defined values of the quadrupole strengths. This means that when the lattice is saved (using `save_lattice`), the quadrupoles will have the corrected values. This would be used for correcting the tunes of a design lattice, for example, but not for correcting tunes of a perturbed lattice.
- `use_perturbed_matrix` — If nonzero, requests use of the perturbed correction matrix in performing correction. For difficult lattices with large errors, this may be necessary to obtain correction. In general, it is not necessary and only slows the simulation.
- `response_matrix_output`, `correction_matrix_output` — The (incomplete) names of SDDS files to which the response and correction matrices will be written.
- `fse_units` — If non-zero, then the data in the `response_matrix_output` and `correction_matrix_output` files is in units of fractional strength error rather than $K_1 L$.

7.20 coupled_twiss_output

- type: setup/action command.
- function: set up or execute computation of coupled twiss parameters and beam sizes
- sequence: must follow `run_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&coupled_twiss_output
    STRING filename = NULL;
    long output_at_each_step = 0;
    long emittances_from_twiss_command = 1;
    double emit_x = 0;
    double emittance_ratio = 0.01;
    double sigma_dp = 0;
    long calculate_3d_coupling = 1;
    long verbosity = 0;
    long concat_order = 2;
&end
```

- `filename` — The (incomplete) name of the SDDS file to which coupled twiss parameters and beam sizes will be written. Suggested value: "%s.ctwi".
- `output_at_each_step` — If nonzero, then this is a setup command and results in computations occurring for each simulation step (e.g., for each perturbed machine if errors are included). If zero, then this is an action command and computations are done immediately (e.g., for the unperturbed machine). If you wish to compute Twiss parameters on a closed orbit or after other calculations, be sure to set this control to a nonzero value.
- `emittances_from_twiss_command` — If nonzero, then the values of the horizontal emittance and the momentum spread are taken from the uncoupled computation done with the `twiss_output` command. In this case, the user must issue a `twiss_output` command prior to the `coupled_twiss_output`. If zero, then the values of the horizontal emittance and the momentum spread are taken from the parameters `emit_x` and `sigma_dp`, respectively.
- `emit_x` — Gives the horizontal emittance, if `emittances_from_twiss_command=0`.
- `emittance_ratio` — Gives the ratio of the x and y emittances. Used to determine the vertical emittance from the horizontal emittance. Note that the computation is not self-consistent. I.e., the user is free to enter any emittance ratio desired, whether it is consistent with the machine optics or now.
- `sigma_dp` — Gives the momentum spread, if `emittances_from_twiss_command=0`.

This feature was added to `elegant` using code supplied by V. Sajaev, based on Ripkin's method. The code computes the coupled lattice functions, then uses the supplied emittance, emittance ratio, and momentum spread to compute the beam sizes, bunch length (if rf is included), and beam tilt.

7.21 divide_elements

- type: setup command.
- function: define how to subdivide certain beamline elements.
- sequence: must precede `run_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- notes:
  - Any number of these commands may be given.
  - Not effective unless given prior to `run_setup`.
  - The `element_divisions` field in `run_setup` provides a simpler, but less flexible, method of performing element division. At present, these element types may be divided: `CSBEND`, `CSRDRIFT`, `DRIFT`, `EDRIFT`, `KOCT`, `KQUAD`, `KQUSE`, `KSEXT`, `OCTU`, `QUAD`, `RBEND`, `RFCA`, `SBEND`, `SEXT`, and `SOLE`.
  - Only effective if given prior to the `run_setup` command.
- warnings:
  - Using `save_lattice` and element divisions together will produce an incorrect lattice file.
  - Element subdivision may produce unexpected results when used with `load_parameters` or parameters saved via the `parameter` entry of the `run_setup` command. If you wish to load parameters while doing element divisions or if you wish to load parameters from a run that had element divisions in effect, you should not load length data for any elements that are (or were) split. The name and item pattern features of `load_parameters` are helpful in restricting what is loaded.

```
&divide_elements
    STRING name = NULL;
    STRING type = NULL;
    STRING exclude = NULL;
    long divisions = 0;
    double maximum_length = 0;
    long clear = 0;
&end
```

- `name` — A possibly wildcard-containing string specifying the elements to which this specification applies.
- `type` — A possibly wildcard-containing string specifying the element types to which this specification applies.
- `exclude` — A possibily wildcard-containing string specifying elements to be excluded from the specification.

- `divisions` — The number of times to subdivide the specified elements. If zero, then `maximum_length` should be nonzero.
- `maximum_length` — The maximum length of a slice. This is usually preferrable to specifying the number of divisions, particularly when the elements divided may be of different lengths. If zero, then `divisions` should be nonzero.
- `clear` — If nonzero, all prior division specifications are deleted.

7.22 elastic_scattering

- type: major action command
- function: perform simulation of elastic scattering at multiple s locations, for use in computing elastic gas scattering lifetime and loss distribution
- sequence: must follow `run_control`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- notes:
  - Only available in `Pelegant`.
  - Data may be postprocessed with the command `elasticScatteringAnalysis`.

```
&elastic_scattering
    STRING losses = NULL;
    STRING output = NULL;
    STRING log_file = NULL;
    double theta_min = 0.001;
    double theta_max = 0.010;
    long n_theta = 11;
    long n_phi = 37;
    long twiss_scaling = 0;
    double s_start = 0;
    double s_end = DBL_MAX;
    STRING include_name_pattern = NULL;
    STRING include_type_pattern = NULL;
    long verbosity = 1;
&end
```

- `losses` — The (incomplete) name of an SDDS file to which the record of initial scattering location, initial scattering angle, and loss coordinates will be written.
- `output` — The (incomplete) name of an SDDS file to which the final coordinates of all surviving particles will be written.
- `log_file` — The (incomplete) name of an SDDS file to which statistical data will be written as the simulations run. Users should check the `MinParticles` and `MaxParticles` columns as the simulation runs to ensure reasonable load balance (e.g., within 10-20%). If balance is poor, consider changing the values of `n_phi` and `n_theta` slightly. The product of these values should not evenly divide the number of working cores (which is one less than the total number of cores).
- `theta_min` — Minimum polar scattering angle in radians. Should be small enough that no particle scattered by this angle are lost, regardless of the scattering location. See also `twiss_scaling`.

- theta_max — Maximum polar scattering angle in radians. Should be large enough that no particle scattered by this angle survives, regardless of scattering location.
- n_theta — Number of polar scattering angle values on the range theta_min to theta_max.
- n_phi — Number of azimuthal scattering angles on the range $[0, \pi]$.
- twiss_scaling — If nonzero, then theta_min is scaled by $\min(\sqrt{\beta_x(0)/\beta_x(s)}, \sqrt{\beta_y(0)/\beta_y(s)})$, where s is the location of the scattering location and $s = 0$ is the start of the lattice.
- s_start, s_end — Range of s location for simulated scattering sites.
- include_name_pattern — Wildcard-containing string to match to element names in selecting scattering sites.
- include_type_pattern — Wildcard-containing string to match to element types in selecting scattering sites.
- verbosity — Higher values may result in more verbose informational output.

7.23 error_element

- type: setup command.
- sequence: must follow `run_control`.
- function: assert a random error defintion for the accelerator.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&error_element
    STRING name = NULL;
    STRING element_type = NULL;
    STRING item = NULL;
    STRING type = "gaussian";
    double amplitude = 0.0;
    double cutoff = 3.0;
    long bind = 1;
    long bind_number = 0;
    longn bind_across_names = 0;
    long post_correction = 0;
    long fractional = 0;
    long additive = 1;
    long allow_missing_elements = 0;
    STRING after = NULL;
    STRING before = NULL;
    STRING sample_file = NULL;
    STRING sample_file_column = NULL;
    STRING sample_mode = NULL;
&end
```

- `name` — The possibly wildcarded name of the elements for which errors are being specified.
- `element_type` — An optional, possibly wildcarded string giving the type of elements to which the errors should be applied. E.g., `element_type=*MON*` would match all beam position monitors. If this item is given, then `name` may be left blank.
- `item` — The parameter of the elements to which the error pertains.
- `type` — The type of random distribution to use. May be one of "uniform", "gaussian", "plus_or_minus", or "sampled". A "plus_or_minus" error is equal in magnitude to the amplitude given, with the sign randomly chosen. A "sampled" error is drawn from a set of user-supplied values, as described below.
- `amplitude` — The amplitude of the errors.
- `cutoff` — The cutoff for the gaussian random distribution in units of the amplitude. Ignored for other distribution types.

- `bind`, `bind_number`, `bind_across_names` — These parameters control "binding" of errors among elements, which means assigning the same error contribution to several elements. This occurs if `bind` is nonzero, **which it is by default**! If `bind` is negative, then the sign of the error will alternate between successive elements. `bind_number` can be used to limit the number of elements bound together. In particular, if `bind_number` is positive, then a positive value of `bind` indicates that `bind_number` successive elements having the same name will have the same error value. Finally, by default, `elegant` only binds the errors of objects having the same name, even if they are assigned errors by the same `error_element` command (i.e., through a wildcard `name`). If `bind_across_names` is nonzero, then binding is done even for elements with different names.
- `post_correction` — A flag indicating whether the errors should be added after orbit, tune, and chromaticity correction.
- `fractional` — A flag indicating whether the errors are fractional, in which case the amplitude refers to the amplitude of the fractional error.
- `additive` — A flag indicating that the errors should be added to the prior value of the parameter. If zero, then the errors replace the prior value of the parameter.
- `allow_missing_elements` — A flag indicating that execution may continue even if no matching elements are found.
- `after` — The name of an element. If given, the error is applied only to elements that follow the named element in the beamline.
- `before` — The name of an element. If given, the error is applied only to elements that precede the named element in the beamline.
- `sample_file`, `sample_file_column`, `sample_mode`— If the error type is "sampled", then `sample_file` must contain the name of an SDDS containing the numerical column named by `sample_file_column`. The values in this column from the first page of the file are used for assigning error values. `sample_mode` may be one of "random", "shuffle", or "sequentual", with the following meanings:
  - `random` — Values are drawn randomly from the list as needed, without regard to reuse of a given value.
  - `shuffle` — Values are drawn from the list in a random order until all are used, then a new random order is created. This ensures that all values are used with equal probability.
  - `sequential` — Values are used in the order given until all are used, repeatedly as needed starting from the beginning of the list.

  This feature can be used to assign errors based on a set of measured values or, using `sddssampledist`, an arbitrary external distribution.

7.24 error_control

- type: setup command
- sequence: must follow `run_control`.
- function: overall control of random errors.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&error_control
    long clear_error_settings = 1;
    long summarize_error_settings = 0;
    long no_errors_for_first_step = 0;
    STRING error_log = NULL;
    double error_factor = 1;
&end
```

- `clear_error_settings` — Clear all previous error settings.
- `summarize_error_settings` — Summarize current error settings. *If non-zero, then the command has no other function except showing a summary of the current error settings.*
- `no_errors_for_first_step` — If non-zero, then there will be no errors for the first step. This can be useful for fiducialization of phase and momentum profiles.
- `error_log` — The (incomplete) name of a SDDS file to which error values will be written. Recommended value: "%s.erl".
- `error_factor` — A value by which to multiply the error amplitudes in all `error` commands.

The proper use of this command can be confusing. A typical sequence will be as follows:

```
&error_control
 clear_error_settings = 1,
 error_log = %s.erl
&end

&error_element ... &end
&error_element ... &end
.
.
.
&error_element ... &end

&error_control
 summarize_error_settings = 1
&end
```

7.25 find_aperture

- type: setup/major action command.
- function: find the aperture in (x, y) space for an accelerator.
- N.B.: can use parallel resources (`Pelegant`). Recommend using n-line mode with `nx`*`n_splits` greater than the number of cores (e.g., a factor of 10).
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&find_aperture
    STRING output = NULL;
    STRING search_output = NULL;
    STRING boundary = NULL;
    STRING mode = "many-particle";
    double xmin = -0.1;
    double xmax =  0.1;
    double xpmin = 0.0;
    double xpmax = 0.0;
    double ymin =  0.0;
    double ymax =  0.1;
    double ypmin = 0.0;
    double ypmax = 0.0;
    long nx  = 21;
    long ny  = 11;
    long n_splits = 0;
    double split_fraction = 0.5;
    double desired_resolution = 0.01;
    long assume_nonincreasing = 0;
    long verbosity = 0;
    long offset_by_orbit = 0;
    long n_lines = 11;
    long optimization_mode = 0;
    long full_plane = 0;
&end
```

- `output` — The (incomplete) name of an SDDS file to send output to. Recommended value: "%s.aper".
- `mode` — May be "many-particle", "single-particle", "one-line, "three-lines", or "n-lines". Many-particle searching is much faster than single-particle, but does not allow interval splitting to search for the aperture boundary. Both "many-particle" and "single-particle" modes involve searching from the outside inward, which improves speed but may result in including islands.

  The line modes avoid this by searching form the origin outward. Of these, the one-line and three-line modes are special: one-line mode searches the line from the origin to (x_{max}, y_{max}). three-line mode searches this line, plus the lines from the origin to $(x_{max}, 0)$ and $(0, y_{max})$.

For n-line mode, the number of lines is set with the `n_lines` parameter. With $n > 3$, n lines are explored from $(0, 0)$ to $(x_{max} * sin(\theta), y_{max} * cos(\theta))$, where θ takes values from $-pi/2$ to $\pi/2$. In these modes, the output file contains a parameter called "Area," which gives the area of the dynamic aperture.

Also still recognized are other modes, namely, "five-line", "seven-line", "nine-line", and "eleven-line".

- `search_output` — The (incomplete) name of an SDDS file for output of detailed information on each tracked particle (single-particle mode only). Recommended value: "%s.apso".
- `boundary` — The (incomplete) name of an SDDS file for the boundary points of the aperture search. Recommended value: "%s.bnd". Valid for many- and single-particle modes.
- `xmin`, `xmax`, `ymin`, `ymax` — Region of the aperture search, in spatial coordinates. The minimum values are relevant only for many- and single-particle modes.
- `xpmin`, `xpmax`, `ypmin`, `ypmax` — Region of the aperture search, in slope coordinates. The minimum values are relevant only for many- and single-particle modes. Ignored unless `xmin=xmax` and `ymin=ymax`.
- `nx` — For many- and single-particle modes, the number of x values to take in initial search. For line modes, this determines the initial x and y step sizes via $\Delta x = x_{max}/n_x$ and $\Delta y = y_{max}/n_x$.
- `ny` — For many- and single-particle modes, the number of y values to take in search. Ignored for line modes.
- `n_splits` — If positive, the number of times to do interval splitting. Interval splitting refers to searching between the original grid points in order to refine the results. This is done only for single-particle and line modes.
- `split_fraction` — If interval splitting is done, how the interval is split.
- `desired_resolution` — If interval splitting is done, fraction of `xmax-xmin` to which to resolve the aperture. Ignored for all but single-particle mode.
- `assume_nonincreasing` — If this variable is non-zero, the search assumes that the aperture at $\mathrm{y} + \mathrm{sign(y)} * \Delta\mathrm{y}$ is no larger than that at y. This results in tracking of fewer particles but may give a pessimistic result. Used only for single- and multi-particle modes.
- `offset_by_orbit` — A flag indicating whether to offset the transverse beam coordinates by the closed orbit before tracking. The default value is zero for backward compatibility, but the recommended value is 1.
- `verbosity` — A larger value results in more printouts during computations.
- `n_lines` — In "n-lines" mode, the number of lines to search.
- `optimization_mode` — If non-zero, then `find_aperture` is a setup command and can be used with `elegant`'s internal optimizer. The quantity `DaArea` is defined, giving the area of the dynamic aperture for use in the penalty function. This is available only for the line search modes.

- full_plane — If non-zero, then the search covers both positive and negative y values. Only available in line-search modes.

7.26 floor_coordinates

- type: action command.
- function: compute floor coordinates for an accelerator.
- sequence: must follow `run_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&floor_coordinates
    STRING filename = NULL;
    double X0 = 0.0;
    double Z0 = 0.0;
    double theta0 = 0.0;
    long include_vertices = 0;
    long vertices_only = 0;
    long magnet_centers = 0;
    long store_vertices = 0;
&end
```

- `filename` — The (incomplete) name of an SDDS file to send output to. Recommended value: "%s.flr".
- `X0`, `Z0`, `theta0` — Initial X, Z, and angle coordinate of the beamline.
- `include_vertices` — Flag that, if set, specifies including in the output the coordinates of the vertices of bending magnets.
- `vertices_only` — Flag that, if set, specifies that output will contain only the coordinates of the vertices of bending magnets.
- `magnet_centers` — Flag that, if set, specifies that output will contain the coordinates of the centers of all magnets, where the center is defined as the average of the entrance and exit points. By default, the coordinates of the downstream end are given.
- `store_vertices` — Flag that, if set, results in storing the floor coordinates for dipole magnet vertex points. The coordinates are stored in variables with names of the form *magnetName*`#`*occurrenceNumber*`-VP.`*property*, where *property* is `X`, `Y`, `Z`, `theta`, `phi`, and `psi`.

The "vertex point" for a dipole or string of dipoles is defined as the intersection of the straight lines from the ideal entrance and exit trajectories. The `s` quantity for the vertex is defined as the sum of the actual distance traveled to the start of the dipole or string of dipoles plus the straight-line distance from the entrace to the vertex. Hence, one cannot subtract the `s` values for two successive vertices and expect to get the distance between the vertices.

7.27 frequency_map

- type: major action command.
- function: compute frequency map from tracking Note that the number of turns tracked is set by the `run_control` command.
- can use parallel resources (`Pelegant`)
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&frequency_map
    STRING output = NULL;
    double xmin = -0.1;
    double xmax = 0.1;
    double ymin = 1e-6;
    double ymax = 0.1;
    double delta_min = 0;
    double delta_max = 0;
    long nx = 21;
    long ny = 21;
    long ndelta = 1;
    long verbosity = 1;
    long include_changes = 0;
    long quadratic_spacing = 0;
    long full_grid_output = 0;
&end
```

- `output` — The (incomplete) name of an SDDS file to send output to. Recommended value: "%s.fma". For the parallel version, particles will be listed in essentially random order. If needed, `sddssort` can be used to sort particles by initial coordinates.
- `xmin`, `xmax` — Limits of grid of initial x coordinates for tracking.
- `ymin`, `ymax` — Limits of grid of initial y coordinates for tracking. `ymin` should be a small, positive value so that there is some betatron oscillation from which to get the tune.
- `delta_min`, `delta_max` — Limits of grid of initial δ coordinates for tracking. Note that particles are not centered around the dispersive closed orbit. Hence, the tracking is appropriate to simulation of dynamics from a touschek scattering event.
- `nx` — Number of values of x coordinate in the grid.
- `ny` — Number of values of y coordinate in the grid.
- `ndelta` — Number of values of δ coordinate in the grid.
- `verbosity` — If nonzero, prints possibly useful information while running.

- include_changes — If nonzero, then computes not only the tunes, but also the changes in the tunes. This is expressed in terms of the diffusion, which is defined as

$$d = \log_{10}\left(\Delta\nu_x^2 + \Delta\nu_y^2\right) \tag{4}$$

where $\Delta\nu_x$ and $\Delta\nu_y$ are respectively the differences in x and y tunes from the first and second half of the tracking (the total number of turns is equal to the value set in run_setup). The diffusion rate,

$$d_r = \log_{10}\left(\frac{\sqrt{\Delta\nu_x^2 + \Delta\nu_y^2}}{N}\right), \tag{5}$$

is also computed. d_r is the more conventional quantity, computed by programs such as TRACY and MAD [53].

- quadratic_spacing — If non-zero, the spacing of points is quadratic rather than linear, thus emphasizing the higher amplitude regions.

- full_grid_output — If non-zero, all grid points are represented in the output file, even if tracking or tune determination failed. This makes it possible to plot with programs (e.g., sddscontour) that require a strictly uniform grid.

7.28 global_settings

- type: action command.
- sequence: should precede `run_setup`.
- function: change global settings.
- Command syntax, including use of equations and subcommands, is discussed in 7.2. One way to use the `global_settings` command is in a configuration file, which can be supplied via the `-configuration` option or `ELEGANT_CONFIGURATION` environment variable; an example of using this would be to set MPI I/O options an a per-system basis. N.B.: unlike other commands, values given for parameters in `global_settings` commands become the new default for subsequent invocations of the command during the same run. Hence multiple commands can be used to set individual values without overriding previously-given settings.

```
&global_settings
    long inhibit_fsync = 0;
    long allow_overwriting = 1;
    long echo_namelists = 1;
    long mpi_randomization_mode = 3;
    long exact_normalized_emittance = 0;
    double SR_gaussian_limit = 3.0;
    long inhibit_seed_permutation = 0;
    STRING log_file = NULL;
    STRING error_log_file = NULL;
    long mpi_io_force_file_sync = 0;
    long usleep_mpi_io_kludge = 0;
    long mpi_io_read_buffer_size = 0;
    long mpi_io_write_buffer_size = 0;
    long parallel_tracking_based_matrices = 1;
    long share_tracking_based_matrices = 1;
    long tracking_based_matrices_store_limit = 5000;
    double tracking_matrix_step_factor = 1;
    double tracking_matrix_points = 9;
    double tracking_matrix_step_size[6] = {5e-5, 5e-5, 5e-5, 5e-5, 5e-5, 5e-5};
    long warning_limit = 10;
    short malign_method = 0;
    double slope_limit = 1.0;
    double coord_limit = 10.0;
&end
```

- `inhibit_fsync` — By default, `elegant` forces file synchronization across a network file system to ensure that users see up-to-date files as soon as possible. In cases where a great deal of output is generated, this can degrade performance. Setting this parameter to 1 will turn off synchronization until the end of the run.
- `allow_overwriting` — By default, `elegant` will overwrite existing output files. If this parameter is set to 0, it will instead exit if an output file already exists.

- echo_namelists — By default, elegant echoes all namelist input to the terminal. If this parameter is set to 0, this output will be inhibited.
- SR_gaussian_limit — By default, elegant uses a 3-σ cutoff for the gaussian random numbers used in simulation of synchrotron radiation from CSBEND, CSRCSBEND, KQUAD, KSEXT, and SREFFECTS. This parameter allows changing the cutoff.
- inhibit_seed_permutation — If nonzero, randomization of the user-supplied random number seed is *not* performed. This feature is useful in that it provides a higher degree of apparent randomness, in that small changes in the seed result in very different random sequences.
- log_file — By default, elegant writes status information to the terminal. If a filename is supplied for this parameter, the output will instead go to the file. On Linux and Unix, using /dev/null will result in the output being discarded.
- error_log_file — By default, elegant writes error messages to the terminal. If a filename is supplied for this parameter, the output will instead go to the file. On Linux and Unix, using /dev/null will result in the output being discarded.
- share_tracking_based_matrices — If non-zero, then the matrices determined by tracking for various elements (e.g., BRAT, BGGEXP, CCBEND) are computed only once for a set of identical elements, then shared. This can save considerable computation time.
- mpi_randomization_mode — Controls how the random numbers are seeded on multiple processors
  - 1 — This is the original default, which showed issues in some simulations. The seed on the i^{th} processor is $s_0 + 2 * i$.
  - 2 — The seed on the i^{th} processor is $s_0 + 2 * i^2$.
  - 3 — This is the new default. The seed on the i^{th} processor is $s_0 + i * (i + 1)$.
  - 4 — The seed on the i^{th} processor is $s_0 + R_i$, where R_i is the i^{th} random integer returned by the system rand() function.
- exact_normalized_emittance — By default, elegant uses an approximate computation for the normalized emittance, namely, $\epsilon_n = \epsilon\langle\beta\gamma\rangle$, where ϵ is the geometric emittance computed from the trace-space coordinates. If this variable is set to a non-zero value, elegant instead uses a slower but more accurate method, namely, using the momentum coordinates. [43]. The results will show up in the sigma and final output files, if these are requested in the run_setup command.
- mpi_io_force_file_sync — If non-zero, Pelegant will perform a file synchronization after writing each row of an SDDS file. This can *significantly* degrade performance, but can solve problems on some filesystems that result in corrupted files or files in which zeros appear in place of the expected data.
- mpi_io_read_buffer_size — If non-zero, Pelegant will change the read buffer size to the given value. May allow improving read performance, but should be used with care.
- mpi_io_write_buffer_size — If non-zero, Pelegant will change the write buffer size to the given value. May allow improving write performance, but should be used with care.

- `usleep_mpi_io_kludge` — If non-zero, `Pelegant` will sleep for the given number of microseconds after writing each row of an SDDS file. This can degrade performance, but can solve problems on some filesystems that result in corrupted files or files in which zeros appear in place of the expected data. It may give better performance than setting `mpi_io_force_file_sync=1`. A value of 100 is suggested as a starting point, but this will be highly system-dependent.

- `parallel_tracking_based_matrices` — If non-zero, then the matrices determined by tracking for various elements (e.g., `BRAT`, `BGGEXP`, `CCBEND`) are computed using parallel resources in `Pelegant`. This can save considerable wall clock time. N.B.: This is set to zero when using `parallel_optimization_setup`.

- `share_tracking_based_matrices` — If non-zero, then the matrices determined by tracking for various elements (e.g., `BRAT`, `BGGEXP`, `CCBEND`) are computed only once for a set of identical elements, then shared.

- `tracking_based_matrices_store_limit` — The maximum number of matrices that will be stored for possible sharing.

- `tracking_matrix_step_factor` — The default step size for tracking-based matrices is 5×10^{-5} (in the appropriate units for each corodinate). This can be increased or decreased by supplying a value for `tracking_matrix_step_factor`.

- `tracking_matrix_points` — By default, five grid points are used in each dimension for tracking-based matrix determination. This can be increased by setting `tracking_matrix_points` to a larger, odd value, at the expense of longer running time. (The run time scales approximately as the sixth power of this value.)

- `tracking_matrix_step_size` — Sets the step sizes, in each of the six coordinates, used for tracking-based matrix determination

- `warning_limit` — Allows setting a limit on the number of similar warnings that are printed in detail. Set to 0 to see no detailed warnings, only the summary; this is not recommended. Set to -1 to see all detailed warnings; this is suggested for debugging only, since it may create large log files and reduce performance.

- `malign_method` — Allows globally setting the default misalignment method *for those elements that lack the* `MALIGN_METHOD` *parameter.*. Five values are recognized:
  - 0 — Indicates the original misalignment method that is entrance-centered and includes three position offsets and roll (`TILT`).
  - 1 — Indicates the new method [58] in entrance-centered mode in the linear approximation. This includes the possibility of pitch and yaw errors on elements that support it.
  - 2 — Indicates the new method in body-centered mode in the linear approximation. This includes the possibility of pitch and yaw errors on elements that support it.
  - 3 — Indicates the new method [58] in entrance-centered mode using an exact form. This includes the possibility of pitch and yaw errors on elements that support it.
  - 4 — Indicates the new method in body-centered mode using an exact form. This includes the possibility of pitch and yaw errors on elements that support it.

- `slope_limit`, `coord_limit` — By default, `elegant` limits the maximum slopes (x' and y') and coordinates (x and y) in order to avoid potential overflows and unreasonable values. These parameters allow users to change those limits.

7.29 ignore_elements

- type: setup command.
- function: causes specified elements to be ignored during tracking.
- Must precede `run_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- notes:
  - Any number of these commands may be given.
  - This command can provide improved *parallel* performance in cases where large numbers of non-transforming elements (e.g., `MARK` or `MONI` elements) exist in a beamline. (The presence of such elements can reduce performance because `elegant` checks particles against aperture limits after *every* element.) Using the `show_element_timing` flag in `run_setup` can help determine if this will help.
  - This command cannot be used if `centroid` or `sigma` output is requested in `run_setup`.

```
&ignore_elements
STRING name = NULL,
STRING type = NULL,
STRING exclude = NULL,
        long disable = 0;
        long clear = 0;
        long completely = 0;
&end
```

- `name` — Possibily wild-card containing string specifying the elements to which the operation is to be applied.
- `type` — Possibily wild-card containing string specifying the element types to which the operation is to be applied.
- `exclude` — Possibily wild-card containing string specifying elements to be excluded from the operation. Does not affect elements included by other specifications.
- `disable` — If nonzero, the command is ignored.
- `clear` — If nonzero, all prior specifications are deleted.
- `completely` — If nonzero, the element is ignore not only for tracking, but for all purposes. (This allows, for example, requesting `sigma` and `centroid` output from `run_setup`.)

7.30 include_commands

- type: setup command.
- function: reads commands from another file.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- notes:
  - Any number of these commands may be given.
  - This command may be used from within other included command files without any limit.

```
&include_commands
        STRING filename = NULL,
        long disable = 0;
&end
```

- `filename` — Name of file from which to read commands.
- `disable` — If nonzero, the command is ignored.

7.31 inelastic_scattering

- type: major action command
- function: perform simulation of inelastic scattering at multiple s locations, for use in computing inelastic gas scattering lifetime and loss distribution
- sequence: must follow `run_control`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- notes:
  - Only available in `Pelegant`.
  - Data may be postprocessed with the command `inelasticScatteringAnalysis`.

```
&inelastic_scattering
    STRING losses = NULL;
    STRING output = NULL;
    STRING log_file = NULL;
    double k_min = 0.001;
    STRING momentum_aperture = NULL;
    double momentum_aperture_scale = 0.90;
    double momentum_aperture_periodicity = 0;
    long n_k = 101;
    double s_start = 0;
    double s_end = DBL_MAX;
    STRING include_name_pattern = NULL;
    STRING include_type_pattern = NULL;
    long verbosity = 1;
    long soft_failure = 0;
    long allow_watch_file_output = 0;
&end
```

- `losses` — The (incomplete) name of an SDDS file to which the record of initial scattering location, initial scattering δ, and loss coordinates will be written.
- `output` — The (incomplete) name of an SDDS file to which the final coordinates of all surviving particles will be written.
- `log_file` — The (incomplete) name of an SDDS file to which statistical data will be written as the simulations run. Users should check the `MinParticles` and `MaxParticles` columns as the simulation runs to ensure reasonable load balance (e.g., within 10-20%). If balance is poor, consider changing the value of `n_delta` slightly.
- `k_min` — Minimum energy k of the brehmsstrahlung photon as a fraction of the beam energy. The electron has $\delta = -k$ after scattering. `k_min` should be small enough that no electron scattered by `-k_min` is lost, regardless of the scattering location.
- `n_k` — Number of scattering values on the range `k_min` to 1.

- momentum_aperture, momentum_aperture_scale — If given, names a file giving the momentum aperture vs s, which is interpolated at the scattering locations to obtain the local momentum aperture. Such a file may be obtained from running the momentum_aperture command. The absolute values of the values in the deltaNegative column will be used in place of k_min. The k_min values thus obtained are multiplied by momentum_aperture_scale, so there is some assurance that the minimally-scattered particles will survive. This ensures that the results are valid for computation of loss rates, for example.
- momentum_aperture_periodicity — If nonzero, the momentum aperture data from momentum_aperture is periodic with the given periodicity.
- s_start, s_end — Range of s location for simulated scattering sites.
- include_name_pattern — Wildcard-containing string to match to element names in selecting scattering sites.
- include_type_pattern — Wildcard-containing string to match to element types in selecting scattering sites.
- verbosity — Higher values may result in more verbose informational output.
- soft_failure — If nonzero, failure to kind a loss does not result in aborting the run.
- allow_watch_file_output — If nonzero, WATCH elements provide output during tracking.

7.32 insert_elements

- type: action command.
- function: Insert elements into a beamline at specified locations. This is a convenient way to add elements to a beamline without modifying the lattice file.
- sequence: must follow `run_setup`.
- notes: The modified beamline can be saved through `save_lattice` command. Be sure to use "output_seq = 1" option in that command.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&insert_elements
        STRING name = NULL;
        STRING type = NULL;
        STRING exclude = NULL;
        double s_start = -1;
        double s_end = -1;
        STRING start_at_element = NULL;
        STRING end_at_element = NULL;
        long skip = 1;
        long disable = 0;
        long insert_before = 0;
        long add_at_end = 0;
        long add_at_start = 0;
        STRING element_def = NULL;
        long total_occurrences = 0;
        long occurrence[100]={0};
        long allow_no_insertions = 0;
        long allow_no_matches = 0;
&end
```

- `name` — Possibly wild-card containing string specifying the names of the elements after which the new element is inserted. A list of comma- or space-separated names may be given.
- `type` — Possibly wild-card containing string specifying the type of the elements after which the new element is inserted.
- `exclude` — Possibly wild-card containing string specifying the names of elements to be excluded from the specification.
- `skip` — New elements are inserted at every n^{th} specified location.
- `s_start`, `s_end` — If positive, these give the starting and ending s locations for insertion of new elements. Note that the s locations are not updated as elements are inserted, but only after completion of all insertions covered by a single command.

- start_at_element, end_at_element — If given, these are used to define (and override) values for s_start and s_end based on the locations of named elements. s_start is taken as the end of the first element with the name given by start_at_element, while s_end is taken as the end of the last element with the name given by end_at_element. No wildcards are accepted.
- disable — If nonzero, the command is ignored.
- insert_before — If nonzero, the insertions are before the selected elements. By default, insertion is after the selected elements.
- add_at_end — If nonzero, the element is also inserted to the end of the beamline.
- add_at_start — If nonzero, the element is also inserted to the start of the beamline, ahead of all other elements.
- element_def — The definition of the new element should be just as it would be entered in the lattice file. To insert an existing element, simply give the name of the element and define it to be a MARK element; elegant will recognize that the element already exists and use the previous definition.
- total_occurrences, occurrence — These parameters are used to insert the new elements after specified occurrences of the element name. total_occurrences specifies how many new elements to add, up to a maximum of 100, while the entries in the array occurrence specify the occurrences after which to add the new elements. If total_occurrences is non-zero, then **skip** must be set to zero and the **name** must be the exact name (no wild-card matching).
- allow_no_matches — If nothing is inserted because there are no matches found to the name and element type, it is normally considered an error. If this parameter is non-zero, a warning is issued but the run continues.
- allow_no_insertions — If nothing is inserted, it is normally considered an error. If this parameter is non-zero, a warning is issued but the run continues.
- insertion_count_variable — If given, the name of an rpn variable into which the number of inserted elements will be written. Can be used, for example, with the field_final_value parameter of optimization_term.

7.33 insert_sceffects

- type: setup command.
- function: set up for transverse space charge calculation.
- sequence: must precede `run_setup`.
- NB: this command is intended only for simulation of space-charge kicks in rings. Please read the manual page for `SCMULT` for details on the algorithm.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&insert_sceffects
        STRING name = NULL;
        STRING type = NULL;
        STRING exclude = NULL;
        long disable = 0;
        long clear = 0;
        STRING element_prefix = "MYSC";
        long skip = 0;
        long vertical = 0;
        long horizontal = 0;
        long nonlinear = 0;
long uniform_distribution = 0;
        long verbosity = 0;
        double averaging_factor = 1;
&end
```

- `name` — Possibily wild-card containing string specifying the name of the elements after which to insert the space charge kick element.
- `type` — Possibily wild-card containing string specifying the type of the elements after which to insert the space charge kick element.
- `exclude` — Possibily wild-card containing string specifying the name of elements to be excluded from the insertion of the space charge kick element.
- `disable` — If nonzero, the command is ignored.
- `clear` — If nonzero, all prior space charge insertions are deleted.
- `element_prefix` — Name under which the space charge kick will appear in the beamline.
- `skip` — If nonzero, the given number of insertion locations are skipped. If zero, only one space charge kick is inserted at the end of beamline.
- `vertical`, `horizontal`, `nonlinear` — If non-zero, then space charge is included in the plane in question.

- `uniform_distribution` — Used for bi-Gaussian distributed beam (coasting beam), i.e., beam that is uniform in z but gaussian in x and y.
- `verbosity` — Larger non-zero values request greater amounts of detail in printouts.
- `averaging_factor` — For nonlinear space charge mode only, this parameter allows applying an infinite-impulse-response (IIR) filter to the turn-by-turn beam size data in order to reduce the effects of noise. A value of 1 means that only data from the present turn is used, while values approaching 0 will tend to use the initial beam sizes only. In more detail, the effective rms beam size $\hat{\sigma}$ used in the calculation of the kicks for the i^{th} turn is

$$\hat{\sigma}_i = f\sigma_i + (1-f)\hat{\sigma}_i, \tag{6}$$

  where σ_i is the actual rms beam size. N.B.: strictly speaking, simulations performed with $f \neq 1$ are invalid, as the effect of strong space charge could be understated. However, judicious use of this parameter may allow valid simulations with fewer particles. The user should vary the parameter to ensure that results are insenstive to the value.

Important notes:

- By default `skip=0`, which results in only one `SCMULT` element at the end of the beamline, regardless of whether values are given for the `name` or `type` fields.
- This element is not designed for space charge calculations in guns or linacs. It is only intended for simulating space charge in rings.
- This command can not work with concatenation-based matrix tracking.
- Some users use `matched_to_cell` in the `bunched_beam` command. This will erase SCMULT assignments along the beamline. In this case, issue another `twiss_output` command just before tracking.

7.34 ion_effects

- type: setup command.
- function: set up for modeling of residual gas ions.
- sequence: must follow `run_setup`.
- Notes:
  1. This feature is considered experimental and should be used with caution. Feedback is welcome. The fitting-based methods, i.e., `bigaussian`, `bilorentzian`, `trigaussian`, and `trilorentzian`, typically show instability when it is not expected and may well have noise challenges that have not been resolved.
  2. One or more `IONEFFECTS` elements must be inserted in the lattice. This can be done manually, or using the `insert_elements` command.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&ion_effects
        STRING pressure_profile = NULL;
        double pressure_factor = 1.0;
        STRING ion_properties = NULL;
        STRING beam_output = NULL;
        long beam_output_all_locations = 0;
        STRING ion_density_output = NULL;
        long ion_output_all_locations = 1;
        long ion_species_output = 0;
        long ion_output_interval = 1;
        STRING field_calculation_method = NULL;
        double gaussian_ion_range = 3;
        double distribution_fit_target = 0.03;
        double distribution_fit_tolerance = 1e-5;
        long distribution_fit_evaluations = 300;
        long distribution_fit_passes = 3;
        long distribution_fit_restarts = 10;
        long hybrid_simplex_comparison_interval = -1;
        STRING fit_residual_type = NULL;
        long macro_ions = 0;
        long symmetrize = 0;
        long generation_interval = 1;
        long multiple_ionization_interval = 100;
        double multiple_ionization_energy_peak = 20;
        double multiple_ionization_energy_rms = 10;
        double ion_span[2] = {0, 0};
        double ion_bin_divisor[2] = {10.0, 10.0};
        double ion_range_multiplier[2] = {2.0, 2.0};
        double ion_sigma_limit_multiplier[2] = {0, 0};
```

```
        long ion_histogram_max_bins = 1000;
        long ion_histogram_min_per_bin = 5;
        STRING ion_histogram_output = NULL;
        double ion_histogram_output_s_start = -1;
        double ion_histogram_output_s_end = -1;
        long ion_histogram_output_interval = 1000;
        long ion_histogram_min_output_bins = 200;
        long ion_poisson_bins[2] = {0, 0};
        long disable_until_pass = 0;
        long freeze_ions_until_pass = 0;
        long freeze_electrons_until_pass = 0;
        long verbosity = 0;
&end
```

- `pressure_profile` — Name of an SDDS file giving the s-dependent gas pressure for various gas species. Column names will be matched to the entries in the `SourceName` column of the `ion_properties` file.
- `pressure_factor` — Factor by which to multiply the pressures given in the `pressure_profile`.
- `ion_properties` — Name of an SDDS file giving properties of ions. Column names are
  - `IonName` — String column giving the name of the ion.
  - `Mass` — Floating-point column giving the ion mass, in AMU.
  - `ChargeState` — Integer column giving the ion charge state (a positive integer).
  - `SourceName` — String column giving the name of the source gas for this ion. Alternately, for a multiply ionized molecule (e.g. CO++) one can give a source ion (e.g. CO+). The source ion must also be defined in the `ion_properties` file.
  - `CrossSection` — Floating-point column giving the cross section for producing the ion from the source, in Mb.
- `beam_output` — Possibly incomplete name of an SDDS file to which beam data will be written. Asking for this output can significantly reduce performance, so it should generally be used for testing only.
- `beam_output_all_locations` — If nonzero, `beam_output` includes data at the location of all `IONEFFECTS` elements. By default, only the first element is included.
- `ion_density_output` — Possibly incomplete name of an SDDS file to which ion density data will be written.
- `ion_output_all_locations` — If nonzero, `ion_density_output` includes data at the location of all `IONEFFECTS` elements. By default, only the first element is included.
- `ion_species_output` — If nonzero, `ion_density_output` includes data for each ion species.
- `ion_output_interval` — The interval in bunches between output of ion data.
- `field_calculation_method` — By default, the fields are computed on the assumption that the beam and ion distributions are gaussian. This is a good assumption for the beam, but not highly accurate for the ions.

One can make use of a Poisson solver for the ion fields by setting `field_calculation_method` to `"poisson"`; *this is presently the recommended method.* The Poisson solver uses the FFTW library and is both more stable and faster than the other methods.

Older alternatives that, though not as good as the Poisson solver, are still more accurate than the gaussian approximation for ion fields are sums of two or three gaussians, or sums of two or three lorentzians, which can be invoked by setting `field_calculation_method` to `"gaussianfit"`, `"bigaussian"`, `"trigaussian"`, `"bilorentzian"`, or `"trilorentzian"`; these are collectively referred to as "histogram fitting methods" below. In the gaussian-fit case, the charge distribution is of the form

$$\rho(x, y) = G(x, h_x, \sigma_x, c_x) * G(y, h_y, \sigma_y, c_y), \tag{7}$$

where $G(q, h, \sigma, c) = h \exp -(q - c)^2/(2\sigma^2)$. In the bigaussian case, the charge distribution is of the form

$$\rho(x, y) = (G(x, h_{x,1}, \sigma_{x,1}, c_{x,1}) + G(x, h_{x,2}, \sigma_{x,2}, c_{x,2})) * (G(y, h_{y,1}, \sigma_{y,1}, c_{y,1}) + G(y, h_{y,2}, \sigma_{y,2}, c_{y,2})). \tag{8}$$

The charge distribution for the bilorentzian is

$$\rho(x, y) = (L(x, h_{x,1}, a_{x,1}, c_{x,1}) + L(x, h_{x,2}, a_{x,2}, c_{x,2})) * (L(y, h_{y,1}, a_{y,1}, c_{y,1}) + L(y, h_{y,2}, a_{y,2}, c_{y,2})), \tag{9}$$

where $L(q, h, a, c) = h/(1 + (q - c)^2/a^2)$.

- `gaussian_ion_range` — If the default field calculation method is used, gives the range (in beam sigma) over which ions are counted, for calculating the ion-beam kicks.
- `distribution_fit_target` — If the distribution field calculation method is selected, gives the target for the fractional deviation of the fit. Smaller numbers will result in long run times.
- `distribution_fit_tolerance` — If the distribution field calculation method is selected, gives the tolerance for the fractional deviation of the fit. Smaller numbers will result in long run times but higher likelihood of reaching the target.
- `distribution_fit_evaluations`, `distribution_fit_passes`, `distribution_fit_restarts` — Parameters for the simplex optimizer that performs the distribution fit. Note that in `Pelegant`, a hybrid simplex method is used, which appears to converge quickly if the default parameters are used.
- `fit_residual_type` — Residual type for distribution fitting. The default is `max-ad-plus-ad-charge`, which indicates using the sum of the maximum absolute deviation and the normalized absolute deviation of the total charge, where the latter is computed from difference of the actual total ion charge and the analytical integral of the charge in the summed distributions; this tends to ensure that there are no hidden spikes in the distribution due to overfitting. Other options are `sum-ad` (sum of normalized absolute deviation), `rms-dev` (sum of normalized rms deviation), `max-ad` (maximum normalized absolute deviation), `max-ad-plus-rms-dev` (sum of maximum normalized absolute deviation and normalized rms deviation), `sum-ad-plus-rms-dev`, `rms-dev-plus-ad-sum`, `sum-ad-plus-ad-sum`, `rms-dev-plus-centroid`, and `rms-dev-plus-ad-charge`.
- `macro_ions` — The number of macro ions to generate per bunch on each turn for which generation is done. The macro ion charge is adjusted according to the cross section and bunch charge. May be overriden by the `MACRO_IONS` parameter on individual `IONEFFECTS`

elements. If this value is too small, the ion distribution will be noisy, which may result in unreliable results. When using the parallel version, setting macro_ions to 1,000 or higher is not unreasonable.

- symmetrize — If nonzero, ions are emitted in symmetric pairs to ensure that the centroids don't deviate from the electron beam centroids because of noise. Doubles the number of macro ions that are emitted. Intended primarily for testing purposes.

- generation_interval — The number of bunches between generation of ions. The macro ion charge is adjusted to account for this, so the effective ion charge after many turns is the same. May be overridden with the GENERATION_INTERVAL parameter on individual IONEFFECTS elements. The actual condition for generation of ions is such that the generating bunches vary on each turn. This can be used to effectively reduce macro_ions below 1, to prevent generation of too many macro ions. This will result in noisy histograms and should be used with caution.

- multiple_ionization_interval — The number of bunches between multiple ionization calculations. The macro ion charge is adjusted to account for this, so the effective ion charge after many turns is the same.

- multiple_ionization_energy_peak, multiple_ionization_energy_rms — Specifies the distribution of the energy of multiply-ionized ions in terms of the peak (or centroid) of the distribution and its rms width, in eV.

- ion_span — The transverse half-extent, in meters, of the region within which ions are modeled. Ions moving outside this region are considered lost. May be overriden by the X_SPAN and Y_SPAN parameters on individual IONEFFECTS elements.

- ion_bin_divisor — For histogram fitting methods, the number of ion bins per rms parameter of the electron beam.

- ion_range_multiplier — For histogram fitting methods, used to determine the full span of the ion binning region bins in units of the rms parameter of the ion distribution. The sign of the value determines which algorithm is used. For $m < 0$, the binning range is $|m|\,\sigma_{ion}$. For $m = 0$, the full span of the ion distribution is included; this may result in a very large number of bins being used to cover a few outlying ions, and is not recommended. For $m > 0$, the code first finds the approximate range containing the central 80% of the ions, then multiplies by m to get the range used.

- ion_sigma_limit_multiplier — For histogram fitting methods, the minimum value for either of the ion sigmas (for bigaussian) or size parameters (for bilorentizan) in units of the bin size. Use to prevent one of the gaussians or lorentzians from being too delta-function-like.

- ion_histogram_max_bins — Maximum number of ion bins for fitting methods. If this limit is reached, the *span* of the histograms will be reduced to ensure that the central portion is resolved. If the value is too large, the histograms may be noisy, which will make the fits unreliable. Also, a large value will result in reduced parallel efficiency, as processors must pass around more data.

- ion_histogram_min_per_bin — Minimum number of ions per bin (on average).

- `ion_histogram_output`, `ion_histogram_output_s_start`, `ion_histogram_output_s_end`, `ion_histogram_output_interval`, `ion_histogram_max_bins` — Controls for the output of ion histograms when using histogram fitting methods. `ion_histogram_output` gives the (incomplete) filename. `ion_histogram_output_s_start` and `ion_histogram_output_s_end` give limits on the s coordinate of the `IONEFFECTS` element. `ion_histogram_output_interval` gives the interval in bunches between output.
- `verbosity` — Larger values result in more output during running. Used for debugging only.
- `ion_poisson_bins` — For the Poisson solver option, the number of bins in x and y. For optimal speed, these should by the multiple of small primes.

The user is strongly advised to study the ion histograms by using the `ion_histogram_output` parameter to request this data. The histograms should not be excessively noisy. The data also includes the fits, which should be close to the data. (For "gaussian" mode, this is generally not possible.) A sample command to examine the histograms and fits for the y plane (generally the most difficult) is

```
sddsplot -column=Position,Charge* run.ionHist -split=page
        -groupby=page -separate=page -graph=line,vary
```

The Poisson solver option makes use of the FFTW library (http://www.fftw.org/). This library must be installed for the Poisson solver to work. The Poisson equation is solved over the full `ion_span`.

7.35 linear_chromatic_tracking_setup

- type: setup command.
- function: define chromatic variation of beta functions, tunes, etc. for using in fast linear-chromatic tracking
- sequence: must follow `run_setup`.
- N.B.: This command is deprecated and no longer maintained. Use a beamline containing one or more `ILMATRIX` elements instead. This provides much more functionality.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&linear_chromatic_tracking_setup
    double nux[4] = {-1, 0, 0, 0};
    double betax[2] = {1.0, 0.0};
    double alphax[2] = {0.0, 0.0};
    double etax[2] = {0.0, 0.0};
    double etapx[2] = {0.0, 0.0};
    double nuy[4] = {-1, 0, 0, 0};
    double betay[2] = {1.0, 0.0};
    double alphay[2] = {0.0, 0.0};
    double etay[2] = {0.0, 0.0};
    double etapy[2] = {0.0, 0.0};
    double alphac[2] = {0.0, 0.0};
&end
```

- `nux` — Provide the horizontal tune plus its first three chromatic derivatives, i.e., $\partial\nu_x/\partial\delta$, $\partial^2\nu_x/\partial\delta^2$, and $\partial^3\nu_x/\partial\delta^3$.
- `betax` — Provide the horizontal beta function plus its chromatic derivative.
- `alphax` — Provide the horizontal alpha function plus its chromatic derivative.
- `etax` — Provide the first- and second-order horizontal dispersion: $\eta_x = \eta_x[0] + \eta_x[1]\,\delta$.
- `etapx` — Provide the first- and second-order horizontal dispersion slope.
- `alphac` — Provide the first and second-order momentum compaction. N.B: if you are tracking with an rf cavity, be sure that your lattice length equal to the actual circumference. See the example below.

7.36 link_control

- type: setup command.
- function: overall control of element parameter links.
- sequence: must follow `run_control`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&link_control
    long clear_links = 1;
    long summarize_links = 0;
    long verbosity = 0;
&end
```

- `clear_links` — Clear all previously set links.
- `summarize_links` — Summarize all current set links.
- `verbosity` — A larger value results in more output during computations.

7.37 link_elements

- type: setup command.
- function: assert a link between parameters of accelerator elements.
- sequence: must follow `run_control` and `link_control`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&link_elements
    STRING target = NULL;
    STRING exclude = NULL;
    STRING item = NULL;
    STRING source = NULL;
    STRING source_from_target_edit = NULL;
    STRING source_position = "before";
    STRING mode = "dynamic";
    STRING equation = NULL;
    double minimium = -DBL_MAX;
    double maximum = DBL_MAX;
    long exclude_self = 1;
&end
```

- `target` — The name of the elements to be modified by the link. May contain wild-cards.
- `exclude` — Wildcard sequence to match to element names. If a match is found, the element is excluded from the link.
- `item` — The parameter that will be modified.
- `source` — The name of the elements to be linked to.
- `source_from_target_edit` — If given and if `source` is not given, an editing command to create the name of the elements to be linked to from the name of the target. Uses the syntax of the `editstring` program.
- `source_position` — May be one of "first", "before", "after", "adjacent", "nearest", or "same-occurrence".
- `mode` — May be either "dynamic" or "static". A dynamic link is asserted whenever the source is changed (during correction, for example). A static link is asserted only when an error or variation is imparted to the source, and at the end of correction.
- `equation` — An `rpn` equation for the new item value in terms of the item values for the source. The prior value of the item is on the top of the stack. To refer to the source parameter values, use the name of the parameters. To refer to the initial source parameter values, append "0" to the parameter name. These names must appear in capital letters.
- `minimum`, `maximum` — Minimum and maximum values that will be assigned to the target parameter.

- `exclude_self` — If nonzero, self-links are blocked. It is not recommended to change this.

7.38 load_parameters

- type: setup command.
- function: load parameters for elements from an SDDS file.
- sequence: must follow `run_setup` and precede `run_control` and `error_control` (if present).
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&load_parameters
        STRING filename = NULL;
        STRING filename_list = NULL;
        STRING include_name_pattern = NULL;
        STRING exclude_name_pattern = NULL;
        STRING include_item_pattern = NULL;
        STRING exclude_item_pattern = NULL;
        STRING include_type_pattern = NULL;
        STRING exclude_type_pattern = NULL;
        STRING edit_name_command = NULL;
        long change_defined_values = 0;
        long repeat_at_each_step = 0;
        long clear_settings = 0;
        long allow_missing_elements = 0;
        long allow_missing_parameters = 0;
        long allow_missing_files = 0;
        long force_occurence_data = 0;
        long verbose = 0;
        long skip_pages = 0;
        long use_first = 0;
&end
```

- `filename` — Name (possibly containing the "%s" field) of SDDS file from which to take data. The file must contain some of the following columns:
  - ElementName — Required string column. The name of the element to change. May contain wildcards.
  - ElementParameter — Required string column. The name of the parameter of the element to change.
  - ParameterValue — Optional double column. If given, gives value of the parameter named in ElementParameter for element named in ElementName.
  - ParameterValueString — Optional string column. If ParameterValue is not present, then this column must be present. The string data will be scanned, if necessary, to obtain a value for the parameter.
  - ParameterMode — Optional string column. If given, for each row the value must be one of "absolute", "differential", "ignore", or "fractional". The meaning of these modes is as follows: absolute mode means the given value is used as the new value for the

parameter; differential mode means the given value is added to the existing value for the parameter; ignore mode means the value is ignored; fractional mode means the existing value is increased by the product of the given value and the existing value (i.e., the given value is a fractional change).

Unless `change_defined_values` is set, successive pages of the file are used for successive steps of the simulation. Several `elegant` commands generate output that may be used (on a subsequent run) with `load_parameters`; among these are the tune and chromaticity correction commands and the `run_setup` command (parameters output).

- `filename_list` — A list of filenames, which may be given in place of `filename`. If used, each file in the list is treated as if it was separately supplied with an individual `load_parameters` command.
- `include_name_pattern`, `exclude_name_pattern` — A comma- or space-separated list of wildcard patterns to be used in selecting, respectively, which elements to include and which to exclude from loading. To be used, data must match at least one inclusion pattern and no exclusion patterns.
- `include_item_pattern`, `exclude_item_pattern` — A comma- or space-separated list of wildcard patterns to be used in selecting, respectively, which items (i.e., which element parameters) to include and which to exclude from loading. To be used, data must match at least one inclusion pattern and no exclusion patterns.
- `include_type_pattern`, `exclude_type_pattern` — Wildcard patterns to be used in selecting, respectively, which element types (e.g., QUAD, DRIFT) to include and which to exclude from loading. To be used, data must match at least one inclusion pattern and no exclusion patterns.
- `edit_name_command` — A command using the syntax of the `editstring` program, allowing the strings in the `ElementName` column to be modified before values are assigned.
- `change_defined_values` — Changes the defined values of the parameters. This means that when the lattice is saved (using `save_lattice`), the parameters will have the altered values. Also, if one wants to alter the values for all steps of the simulation, one must set this flag.

  Note that the `ElementOccurence` data is normally ignored if `change_defined_values` is nonzero. This is because there is only one definition of each element, even if it is used multiple times. This behavior can be altered with the next control.
- `repeat_first_page_at_each_step` — Normally each simulation step results in loading a new page from the file. If this parameter is non-zero, then the first page is instead loaded at each step. This can be used, for example, to reassert a starting configuration prior to other changes, or to load a common set of changes into a series of configurations.
- `force_occurence_data` — If set, then occurence data is used even in `change_defined_values` mode. When loading data for a highly repetitive system, where many elements have identical names, this can greatly speed completion of the operation.
- `use_first` — It is possible that the input file will contain multiple lines for any given parameter. In this case, `elegant` will by default process all lines. For example, if the lines give differential values, then all would be included. However, if the lines give absolute values,

then the last one will overwrite the previous values; this flag allows overriding the behavior in this case to force `elegant` to use the first value. This can have speed advantages for cases where there are many identical occurences of the same element with identical values for the parameters.

- `clear_settings` — If set, clear all settings and files being used for loading parameters.
- `allow_missing_elements` — If set, allow elements in the file that are not in the lattice. In this case, the nonapplicable data is simply ignored.
- `allow_missing_parameters` — If set, it is not an error if any element in the lattice lacks a parameter that exists in the file.
- `allow_missing_files` — If set, it is not an error if any listed file is missing.
- `verbose` — If set, provide informational printouts about changes to parameters.
- `skip_pages` — Specify the number of pages of input to skip.

7.39 matrix_output

- type: setup/action command.
- function: generate matrix output, or set up to do so later.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&matrix_output
    STRING printout = NULL;
    long printout_order = 1;
    STRING printout_format = "%22.15e ";
    long full_matrix_only = 0;
    long print_element_data = 1;
    long mathematica_full_matrix = 0;
    STRING mathematica_matrix_name = "MFull";
    STRING mathematica_matrix_file = NULL;
    STRING SDDS_output = NULL;
    long SDDS_output_order = 1;
    long individual_matrices = 0;
    STRING SDDS_output_match = NULL;
    long output_at_each_step = 0;
    STRING start_from = NULL;
    long start_from_occurence = 1;
&end
```

- `printout` — The (incomplete) name of a file to which the matrix output will be printed (as text). Recommended value: "%s.mpr".
- `printout_order` — The order to which the matrix is printed.
- `printout_format` — The C-style formatting statement for the matrix elements. A space, comma, or other separator should appear at the end of the string.
- `full_matrix_only` — A flag indicating that only the matrix of the entire accelerator is to be output.
- `print_element_data` — A flag indicating whether the element data should be printed out.
- `mathematica_full_matrix` — If non-zero, print the full linear matrix in a format that can read by Mathematica.
  - `mathematica_matrix_name` — The name of the Mathematica variable to which the linear matrix will be assigned.
  - `mathematica_matrix_file` — The name a file to which the Mathematica-format matrix will be written.
- `SDDS_output` — The (incomplete) name of an SDDS file to which the matrix will be written. Recommended value: "%s.mat".

- `SDDS_output_order` — The order to which the matrix is output in SDDS format.
- `individual_matrices` — If non-zero, the matrices in the SDDS file are the individual *on-trajectory* matrices of the elements, rather than the concatenated matrix of the beamline.
- `SDDS_output_match` — A wildcard string which element names must match in order for data to appear in the SDDS output file.
- `output_at_each_step` — A flag indicating whether matrix output is desired at every simulation step.
- `start_from` — The optional name of the accelerator element from which to begin concatenation and output.
- `start_from_occurence` — If `start_from` is not NULL, the number of the occurrence of the named element from which to start.

7.40 modulate_elements

- type: setup command.
- function: define parameters for time-dependent modulation of elements
- sequence: must follow `run_setup`.
- N.B.: if the ramped element is modeled with a matrix, a significant performance hit may be seen. It is best to use symplectic variants of the elements, since these don't invoke the matrix calculation.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&modulate_elements
    STRING name = NULL;
    STRING item = NULL;
    STRING type = NULL;
    STRING expression = NULL;
    STRING filename = NULL;
    STRING time_column = NULL;
    long convert_pass_to_time;
    STRING amplitude_column = NULL;
    long refresh_matrix = 0;
    long differential = 1;
    long multiplicative = 0;
    long start_occurence = 0;
    long end_occurence = 0;
    double s_start = -1;
    double s_end = -1;
    STRING before = NULL;
    STRING after = NULL;
    long verbose = 0;
    double verbose_threshold = 0;
    STRING record = NULL;
    long flush_record = 1;
&end
```

N.B.: This command will produce unpredictable results when used with `error_element`, `alter_elements`, and `load_parameters` (except when `change_defined_values=1`). It should work properly with `link_elements` in turn-by-turn mode when the source element is modulated, but not when the target element is modulated.

- `name` — A possibly-wildcard-containing string giving the names of the elements to modulate. If not specified, then one must specify `type`.
- `item` — The name of the parameter to modulate.

- `type` — A possibly-wildcard-containing string giving the names of element *types* to modulate. May be specified with `name` or by itself.
- `expression` — RPN expression for the modulation amplitude A. The value of the time is on top of the stack.
- `filename` — Name of SDDS file from which to read modulation data, if `expression` is not used.
- `time_column` — Name of column in `filename` giving time data for the modulation table.
- `convert_pass_to_time` — By default, the mean arrival time of the beam is used to compute the time value for computing the modulation amplitude. If the arrival time vales are offset by `CHANGE_T=1` on `RFCA` elements, this won't work as desired. In that case, one can compute the time from the pass number and the position of the element within the lattice.
- `amplitude_column` — Name of column in `filename` giving amplitude data for the modulation. Together, `time_column` and `amplitude_column` define a function $A(t)$.
- `refresh_matrix` — Frequently there is a matrix associated with an element even if tracking through the element does not use the matrix. In this case, `elegant` doesn't normally update the matrix for the element as it modulates the element, since that may involve a significant time penalty. If this parameter is set to a non-zero value, the matrix will be updated. For elements that use a matrix for tracking, the matrix is always updated.
- `differential`, `multiplicative` — Determine how the amplitude function $A(t)$ is used to obtain the new value of the parameter. There are four cases
  - `differential=1`, `multiplicative=0`: $v(t) = v_0 + A(t)$ (default).
  - `differential=0`, `multiplicative=0`: $v(t) = A(t)$.
  - `differential=1`, `multiplicative=1`: $v(t) = v_0 + v_0 A(t)$.
  - `differential=0`, `multiplicative=1`: $v(t) = v_0 A(t)$.
- `start_occurence`, `end_occurence` — If nonzero, these give the starting and ending occurrence numbers of elements that will be modulated. N.B.: if wildcards are used, occurrence number counting is for each set of identically-named elements separately, rather than for the sequence of matched elements.
- `s_start`, `s_end` — If non-negative, these give the gaving and ending position limits for the end-of-element locations of elements to be modulated.
- `after` — The name of an element. If given, the modulation is applied only to elements that follow the named element in the beamline.
- `before` — The name of an element. If given, the modulation is applied only to elements that precede the named element in the beamline.
- `verbose` — If nonzero, information is printed to the standard output as changes are made. Use for debugging only, since otherwise it may slow the simulation.
- `verbose_threshold` — If nonzero, verbose information is printed only when the fractional change exceeds the given value.

- `record` — Gives a possibly incomplete filename to which will be written a record of the values of the modulation.

- `flush_record` — Gives the interval in steps at which to flush the record file. Higher values result in less frequent updates to the record, but may improve performance.

7.41 moments_output

- type: action/setup command.
- function: compute periodic or propagate non-periodic beam moments without tracking, optionally including radiation.
- sequence: must follow `run_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&moments_output
    STRING filename = NULL;
    long output_at_each_step = 0;
    long output_before_tune_correction = 0;
    long final_values_only = 0;
    long verbosity = 0;
    long matched = 1;
    long equilibrium = 1;
    long radiation = 1;
    long n_slices = 10;
    long tracking_based_diffusion_matrix_particles = 1000;
    long slice_etilted = 1;
    double emit_x = 0;
    double beta_x = 0;
    double alpha_x = 0;
    double eta_x = 0;
    double etap_x = 0;
    double emit_y = 0;
    double beta_y = 0;
    double alpha_y = 0;
    double eta_y = 0;
    double etap_y = 0;
    double emit_z = 0;
    double beta_z = 0;
    double alpha_z = 0;
&end
```

- `filename` — The (incomplete) name of a file to which the moments results will be written. Recommended value: "%s.mom".
- `output_at_each_step` — A flag indicating, if set, that computations and/or output is desired at each step of the simulation. If you wish to compute Twiss parameters on a closed orbit or after other calculations, be sure to set this control to a nonzero value.
- `output_before_tune_correction` — A flag indicating, if set, that output is desired both before and after tune correction.

- final_values_only — A flag indicating, if set, that only the final values of the Twiss parameters should be output, and not the parameters as a function of s.

- verbosity — Larger numbers result in an increasing amount of informational output to the standard output stream.

- matched — A flag indicating, if set, that the periodic or matched moments should be found.

- equilibrium — A flag indicating, if set, that the equilibrium moments should be found. If matched=1 and equilibrium=0, then the initial twiss parameters are computed from the periodic solution for the beamline.

- radiation — A flag indicating, if set, that synchrotron radiation effects should be included. N.B.: this flag is all that needs to be set if the lattice contains no kick elements. However, if the lattice contains CSBEND, CSRCSBEND, KQUAD, or KQUAD elements (or other elements with SYNCH_RAD and ISR parameters), then the SYNCH_RAD andISR must be set to 1 as well.

- n_slices — The number of slices into which to cut individual dipoles, quadrupoles, and sextuoples for computations. 10 has been found to work for all rings tested, but users are advised to ensure it is sufficient for their cases.

- tracking_based_diffusion_matrix_particles — For most elements, the diffusion matrix determined by moments_output is computed used matrix concatenation. For some elements, this doesn't work well because of possible internal coordinate transformations. For these elements, the diffusion matrix is determined approximately by tracking an ensemble of tracking_based_diffusion_matrix_particles particles. Setting this parameter to 0 will disable this feature.

- emit_x, beta_x, alpha_x, eta_x, etap_x, and related quantities for y and z — If matched=0, then these specify the starting beam ellipses in all three planes.

This command performs several functions. In the most basic form, it propagates beam moments, i.e., the 6x6 sigma matrix, from the beginning to the end of a transport line, including coupling from rotated elements or offset sextupoles. This can be performed with or without synchrotron radiation effects in dipoles, quadrupoles, and sextupoles. These computations include the evolution of the trajectory due to errors and (if included) synchrotron radiation.

If desired, the command will instead compute the periodic beam moments. In this case, the user must include an appropriate rf cavity in the lattice in order to get valid results. (By "appropriate rf cavity" we mean that it must have the right voltage, frequency, and phase to support stored beam.) It is also suggested that the user compute the closed orbit using closed_orbit so that the computations are performed on the closed orbit.

The results of moments computation may be subjected to optimization using values at marker elements. See the documentation for MARK for more details.

Notes:

- When using CSBEND, KQUAD, and KSEXT elements, one may find that the calculations of moments_output do not make sense. This is because, by default, synchrotron radiation is disabled on these elements. To resolve the issue, set ISR=1 and SYNCH_RAD=1 on CSBEND at a miminum. If a closed orbit is present, making the same setting on the KQUAD and KSEXT is also suggested. It is essential to do this if there is an rf frequency offset.

- When bending magnets are tilted, `elegant` has problems computing the moments and closed orbit self-consistently when the bending radius is small. To address this, the `n_slices` parameter is set to 1 for tilted bending magnets when `slice_etilted=0`. This reduces the accuracy of the calculations. **Users are strongly advised to check that this is acceptable.**
- The program `sddsmatchmoments` is available to transform a particle distribution so that its 6x6 beam moments match those given in a `moments_output` output filename. In addition, the `bunched_beam` command provides a similar capability for generating a distribution from computed moments.

7.42 momentum_aperture

- type: major action command.
- function: determine momentum aperture as a function of position in the lattice by tracking
- can use parallel resources (Pelegant)
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&momentum_aperture
    STRING output = NULL;
    double x_initial = 0;
    double y_initial = 0;
    double delta_negative_start = 0.0;
    double delta_positive_start = 0.0;
    double delta_negative_limit = -0.10;
    double delta_positive_limit = 0.10;
    double delta_step_size = 0.01;
    long steps_back = 1;
    long splits = 2;
    long split_step_divisor = 10;
    long skip_elements = 0;
    long process_elements = 2147483647;
    double s_start = 0;
    double s_end = DBL_MAX;
    STRING include_name_pattern = NULL;
    STRING include_type_pattern = NULL;
    long fiducialize = 0;
    long verbosity = 1;
    long soft_failure = 0;
    long output_mode = 0;
    long forbid_resonance_crossing = 0;
&end
```

- `output` — The (incomplete) name of a file to which the momentum aperture results will be written. Recommended value: "%s.mmap". The data are related to the momentum aperture at the exit of the named elements.
- `x_initial`, `y_initial` — The initial x and y coordinate values for tracking. It is essential that `y_initial` be nonzero if one wants to see losses due to vertical resonances.
- `delta_negative_start`, `delta_positive_start` — Starting values of scans in the negative and positive directions.
- `delta_negative_limit`, `delta_positive_limit` — Limiting values of scans in the negative and positive directions.
- `delta_step_size` — Initial size of steps in δ. This should be fairly large in order to save time.

- steps_back — Number of steps to back up after a particle is lost, relative to the last surviving δ, before continuing with a smaller step size. If this is set to zero, there is a risk of finding a too-large momentum aperture (a stable island).
- splits — Number of times to split the step size in order to refine the location of the maximum surviving momentum offsets. When a particle is lost, the algorithm steps back to a momentum offset where a particle survived, subdivides the step size, and continues searching.
- split_step_divisor — Factor by which to subdivide the step size for each split.
- skip_elements — Number of elements to skip before starting to compute momentum apertures.
- process_elements — Number of elements for which to compute momentum aperture.
- s_start, s_end — Limiting s coordinates of the elements from which tracking will start. The default values will exclude no elements.
- include_name_pattern — If given, tracking will start only at the entrance to elements that match the given wildcard pattern.
- include_type_pattern — If given, tracking will start only at the entrance to elements whose type matches the given wildcard pattern.
- fiducialize — If given, an initially on-energy particle is tracked before the momentum aperture search begins, in order to fiducialize the reference momentum. This is useful if there are synchrotron radiation losses or energy gain due to cavities in the system.
- verbosity — Larger values result in more detailed printouts as calculations proceed. Mostly for debugging.
- soft_failure — Normally, if elegant fails to find the momentum aperture, it aborts. If soft_failure is non-zero, it instead assigns a momentum aperture equal to the search limit.
- output_mode — Normally, elegant puts the values for positive and negative momentum aperture in different columns. Each element thus has a single row of data in the output file. If output_mode=1, elegant instead puts the values for positive and negative apertures in successive rows, with a reduced number of columns. This is mostly advantageous for the parallel version, since it allows using twice as many simultaneous processors. If output_mode=2, elegant tracks many more probe particles simultaneously, which is better for massively parallel systems. The number of particles tracked is the number of elements selected times the number of probe points between delta_negative_limit and delta_positive_limit.
- forbid_resonance_crossing — Normally, elegant allows the momentum aperture search to cross integer and half-integer resonances if no unstable particles are found. If this is undesirable, this flag can be set to 1.

The idea for this command is from M. Belgroune *et al.*, "Refined Tracking Procedure for the SOLEIL Energy Acceptance Calculation," Proceedings of PAC 2003, p 896, as implemented for TRACYII. In particular, the energy aperture as a function of position around the ring is determined by tracking. Starting at the beginning of the lattice and working downstream, particles are tracked starting from the exit of each selected element. The betatron coordinates are initially zero (or very small), while the momentum deviation is gradually increased until loss of the particle is observed. This defines the momentum aperture at that location.

In `elegant` version 19.0 and later, the algorithm is as follows. For simplicity in wording, we'll assume the momentum deviations are positive values, although the method is applied separately for negative values as well:

1. Start with $\delta = 0$, i.e., zero momentum offset.
2. Track a particle to see if it gets lost. If so, proceed to step 4.
3. Increase δ by step size $\Delta\delta$ and return to step 2.
4. If no splitting steps remain, proceed to the next step. Otherwise:
   (a) Change δ to $\delta_s - s_b\Delta\delta$., where δ_s is the largest δ for which the particle survived, and s_b is the `steps_back` parameter.
   (b) Divide the step size by `split_step_divisor` to get a new step size $\Delta\delta$.
   (c) Set $\delta = \delta + \Delta\delta$.
   (d) Decrement the "splits remaining" counter by 1.
   (e) Continue from step 2.
5. Stop. The momentum aperture is δ_s

This command can be used for both rings and transport lines. For rings it is most appropriate to have an rf cavity (i.e., an `RFCA` element) in the lattice. One should also include radiation loss using either of two methods:

1. `SREFFECTS` element, with `QEXCITATION=0`. To set up this element more easily, one can include a `twiss_output` command with `radiation_integrals=1`.
2. Use `CSBEND` and `KQUAD` elements with `SYNCH_RAD=1` and `ISR=0`.

When including radiation loss, one must be certain to set the frequency and phase of the rf cavity correctly. The `rf_setup` command can be used for this purpose. It is also a good idea to track for several synchrotron oscillation periods.

Note for Pelegant: Unlike for `elegant`, the data in the output file will not be sorted by `s`. To sort the data, simply use `sddssort` from the commandline, e.g.,

```
sddssort -column=s output.mmap
```

Also, if it is desirable for the output from `Pelegant` to have exactly the same form as that from `elegant`, then the script `reorganizeMmap` should be used. This script is provided with `elegant` and `Pelegant` distributions.

7.43 obstruction_data

- type: setup command
- function: define obstructions in the global coordinate system (Z, X)
- sequence: must follow `floor_coodinates` and preceed `track`
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&obstruction_data
        STRING input = NULL;
        long periods = 1;
        long disable = 0;
        double y_spacing = 0;
        double y_limit[2] = {-10, 10};
&end
```

- `input` — Name of SDDS file containing obstruction data. The file must contain two columns, `Z` and `X`, giving the global coordinates of points on the obstruction contour in the $Y = 0$ plane, in units of meters. The file may contain more than one page, with each page giving a closed contour for a separate obstruction. By default, obstructions are considered to extend over $Y : [-\infty, \infty]$; however, different contours can be defined for different vertical planes, as described below. The file must also contain three parameters:
  - `Superperiodicty` —- integer parameter giving the number of repetitions of the defined obstructions in the full ring. For example, for the 40-sector APS ring, if the obstruction data covered a single sector, then the value would be 40.
  - `XCenter`, `ZCenter` — floating point parameters giving the position of center of the ring in meters. The obstructions are rotated about this center if periodic.

  The file may optionally contain two other parameters:

  - `CanGo` — This integer parameter can be used to distinguish between "no-go" and "can-go" regions. By default, all contours enclose "no-go" regions, so that particles inside are assumed to be lost. If this parameter is specified and the value is non-zero, the contour instead defines a region inside which particles *can* propagate. This is typically used to override part of a no-go region.
  - `Y` — This floating-point parameter can be used to specify the vertical plane occupied by the contour. The values must be equispaced with spacing giving by the `y_spacing` parameter, as discussed below.
- `periods` — Obstructions represented by the provided data are to be repeated in a periodic fashion the number of times given. It is assumed that the system is a storage ring.
- `disable` — If nonzero, then the command is ignored.

- `y_spacing` — If nonzero, then the input file is expected to have data for multiple vertical planes, instead of the default $y = 0$ midplane. The plane for each page is identified by the parameter `Y` in the input file. The data must be sorted in increasing order of `Y`, which can be accomplished using `sddssort`, e.g.,

  ```
  sddssort input.sdds -parameter=Y
  ```

- `y_limit` — Allows specifying maximum limits on the vertical coordinate, beyond which particles are lost.

7.44 optimize

- type: major action command.
- function: perform optimization.
- sequence: must follow `optimization_setup` and beam definition (`bunched_beam` or `sdds_beam`).
- can use parallel resources (`Pelegant`) for tracking-based optimization.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- note: on UNIX systems, the user may press Control-C to force `elegant` to terminate optimization and proceed as if optimization had converged. (To genuinely terminate the run during optimization press Control-C twice.) This is very useful if one wants to get a look at the partially optimized result. If one uses parameter saving (`run_setup`) or `save_lattice` one can make a new run that starts from the optimized result.

```
&optimize
    long summarize_setup = 0;
&end
```

- `summarize_setup` — A flag indicating, if set, that a summary of the optimization parameters should be printed.

7.45 optimization_constraint

- type: setup command.
- function: define a constraint for optimization.
- sequence: must follow `optimization_setup` and precede beam definition (`bunched_beam` or `sdds_beam`).
- N.B.: This command is *disparaged*. It is *far* better to put constraints into the optimization equation (via the `equation` parameter of `optimization_setup` or via `optimization_term`). The reason is that the hard constraints imposed by `optimization_constraint` may make it more difficult for the optimizer to converge. See the discussion of the `selt` and `segt` macros in the manual entry to `optimization_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&optimization_constraint
    STRING quantity = NULL;
    double lower = 0;
    double upper = 0;
&end
```

- `quantity` — The quantity to be constrained, given as the name of a quantity from among the optimization variables, optimization covariables, and the "final" parameters (see the entry for `run_setup` for the last of these). The optimization (co)variables are referred to as `<element-name>.<parameter-name>`, in all capital letters. Other quantities, such as Twiss parameters or anything else but what is listed just above, are not recognized. Expressions involving multiple quantities are not supported.
- `lower`, `upper` — The lower and upper limits allowed for the expression.

7.46 optimization_covariable

- type: setup command.
- function: define an element parameter to be varied as a function of optimization parameters.
- sequence: must follow `optimization_setup` and precede beam definition (`bunched_beam` or `sdds_beam`).
- N.B.: It is not possible to optimize an element if the element name starts with one of the following characters: `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `.`, `+`, or `-`. The reason is that `elegant` will attempt to make an SDDS parameter name containing the element name, and these characters are disallowed at the beginning of such a name.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&optimization_covariable
    STRING name = NULL;
    STRING item = NULL;
    STRING equation = NULL;
    long disable = 0;
&end
```

- `name` — The name of the element.
- `item` — The parameter of the element to be changed.
- `equation` — An `rpn` equation for the value of the parameter in terms of the values of any parameters of any optimization variable. These latter appear in the equation in the form `<element-name>.<parameter-name>`, in all capital letters. The original values of all variables and covariable may be accessed via names like `<element-name>.<parameter-name>0`.
- `disable` — If nonzero, the covariable is ignored.

7.47 optimization_setup

- type: setup command.
- function: define overall optimization parameters and methods.
- sequence: must precede beam definition (`bunched_beam` or `sdds_beam`)
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&optimization_setup
    STRING equation = NULL;
    STRING mode = "minimize";
    STRING method = "simplex";
    double tolerance = -0.01;
    double target = 0;
    long center_on_orbit = 0;
    long center_momentum_also = 1;
    long soft_failure = 1;
    long n_passes = 2;
    long n_evaluations = 500;
    long n_restarts = 0;
    double restart_reset_threshold = 0;
    double restart_worst_term_factor = 1;
    long restart_worst_terms = 1;
    long matrix_order = 1;
    STRING log_file = NULL;
    STRING term_log_file = NULL;
    long output_sparsing_factor = 0;
    long balance_terms = 0;
    long verbose = 1;
    long balance_terms = 0;
    double simplex_divisor = 3;
    double simplex_pass_range_factor = 1;
    double rcds_step_factor = 0.1;
    long include_simplex_1d_scans = 1;
    long start_from_simplex_vertex1 = 0;
    long restart_random_numbers = 0;
    STRING interrupt_file = "%s.interrupt";
    double interrupt_file_check_interval = 10;
&end
```

- `equation` — An `rpn` equation for the optimization function, expressed in terms of any parameters of any optimization variables, the "final" parameters of the beam (as recorded in the `final` output file available in the `run_setup` namelist), and selected quantities from Twiss parameter, tune shift with amplitude, closed orbit, beam moments, driving terms, and other computations. The optimization variables or covariables may appear in the equation in the form `<element-name>.<parameter-name>`, all in capital letters. In addition, initial values of variables and covariables are available in the form `<element-name>.<parameter-name>0`.

Data from MARK elements with `FITPOINT=1` and from beam position monitors with `CO_FITPOINT=1` may be used via symbols of the form *`elementName#occurenceNum.parameterName`*. See the documentation for the `MARK`, `MONI`, `HMON`, and `VMON` elements for detailed discussion and listing.

If response matrix calculation is requested, response matrix values are available in variables with names *Plane*`R_`*bpmName*`#`*occurence*`_`*corrName*`#`*occurence*`.`*corrParam*, where *Plane* is `H` (horizontal) or `V` (vertical) and *corrParam* is the parameter of the corrector used for changing the orbit (e.g., `HKICK` or `VKICK` for a `KICKER` element).

If cross-plane response matrix calculation is requested, response matrix values are available in variables with names *BpmPlaneCorrPlane*`R_`*bpmName*`#`*occurence*`_`*corrName*`#`*occurence*`.`*corrParam*, where *BpmPlane* and *CorrPlane* are `H` (horizontal) or `V` (vertical) and *corrParam* is the parameter of the corrector used for changing the orbit (e.g., `HKICK` or `VKICK` for a `KICKER` element).

Many quantities are made available for optimization if `twiss_output` command is given with `output_at_each_step=1`:

- Final Twiss parameters, e.g., `betax`, `alphax`, `etax`. The names are the same as the column names in the twiss output file.
- Linear acceptances `Ax` and `Ay` for the horizontal and vertical planes, respectively.
- Statistics of Twiss parameters in the form `<statistic>.<parameter-name>`, where `<statistic>` is `min`, `max`, `ave`, `p99`, `p98`, or `p96`. `p99` is the 99^{th} pencentile value, a similarly for `p98` and `p96`.
- Tunes and chromaticities via symbols `nux`, `dnux/dp`, (and corresponding symbols for y).
- Chromatic derivatives of beta and alpha functions, via symbols `dbetax/dp`, `dbetay/dp`, `dalphax/dp`, and `dalphay/dp`.
- First-, second-, and third-order momentum compaction factors via symbols `alphac`, `alphac2`, and `alphac3`, respectively.
- If radiation integral computation is requested, one may use `ex0` and `Sdelta0` for the equilibrium emittance and momentum spread, plus `J<plane>` and `tau<plane>` for the damping partition and damping time, where `<plane>` is `x`, `y`, or `delta`. One may also use `I1` through `I5` for the individual radiation integrals.
- If `compute_driving_terms=1`, then the quantities `h11001`, `h00111`, `h20001`, `h00201`, `h10002`, `h21000`, `h30000`, `h10110`, `h10020`, `h10200`, `h22000`, `h11110`, `h00220`, `h31000`, `h40000`, `h20110`, `h11200`, `h20020`, `h20200`, `h00310`, `h00400`, `dnux/dJx`, `dnux/dJy`, and `dnuy/dJy` may be used. Table 2 explains the meaning of the terms.
- The coupling integral and emittance ratio due to x-y coupling may be accessed using the symbols `couplingIntegral` and `emittanceRatio`. See section 3.1.4.4 of [19].
- If higher-order chromaticity is requested, then one may use the symbols `dnux/dp2`, `dnux/dp3`, `dnuy/dp2`, `dnuy/dp3`, `etax2` , `etax3`, `etay2` , `etay3`, `nuxChromLower`, `nuxChromUpper`, `nuyChromLower`, and `nuyChromUpper`.
- If the `tune_shift_with_amplitude` command was also given and one may use the symbols `dnux/dAx`, `dnux/dAy`, `dnuy/dAx`, `dnuy/dAy`, `dnux/dAx2`, `dnux/dAy2`, `dnux/dAxAy`, `dnuy/dAx2`, `dnuy/dAy2`, `dnuy/dAxAy`, `nuxTswaLower`, `nuxTswaUpper`, `nuyTswaLower`, and `nuyTswaUpper`.

– If HMON, VMON, or MONI elements are in the beamline, the symbol sMaxTransmittedMonitor records the s position of the last monitor that sees at least one particle.

If the floor_coordinates command was given, one may use X, Z, and theta to refer to the final values of the floor coordinates.

If the sasefel command was given, one may use variables of the form SASE.<property>, where <property> is one of gainLength, saturationLength, saturationPower, or lightWavelength.

Finally, one may use any of the names from the "final" output file (see run_setup), e.g., Sx (x beamsize) or eny (y normalized emittance). These refer to tracked properties of the beam.

The equation may be left blank, in which case the user must give one or more optimization_term commands. These use the same symbols, of course.

There are several rpn functions that are useful in constructing a good optimization equation. These are "soft-edge" greater-than, less-than, and not-equal functions, which have the names segt, selt, and sene, respectively. The usage is as follows:

– *V1 V2 T* segt. Returns a nonzero value if and only if value *V1* is greater than *V2*. The returned value is $((V_1 - V_2)/T)^2$. Typically used to constraint a quantity from above. E.g., to limit the maximum horizontal beta function to 10m with a tolerance of $T = 0.1m$, one would use max.betax 10 .1 segt.
– *V1 V2 T* selt. Returns a nonzero value if and only if value *V1* is less than value *V2*. The returned value is $((V_1 - V_2)/T)^2$. Typically used to constrain a value from below. E.g., to limit a beta function to greater than 3 m with a tolerance of 0.1 m, one would use betax 3 .1 selt.
– *V1 V2 T* sene. Returns a nonzero value if and only if *V1* and *V2* differ by more than *tol*. If $V_1 > V_2$, returns $((V_1 - (V_2 + T))/T)^2$. If $V_2 > V_1$, returns $((V_2 - (V_1 + T))/T)^2$.

- mode — May be either "minimize" or "maximize".
- method — May be one of "simplex", "grid", "powell", "randomwalk", "randomsample", "1dscans", "rcds", and "sample". Recommended methods are "simplex", "rcds" [62], and "randomwalk". The latter is very useful when the lattice is unstable or nearly so.
- tolerance — The convergence criterion for the optimization, with a negative value indicating a fractional criterion.
- target — The value which, if reached, results in immediate termination of the optimization, whether it has converged or not.
- center_on_orbit — A flag indicating whether to center the beam transverse coordinates on the closed orbit before tracking.
- center_momentum_also — A flag indicating whether to center the momentum coordinate also.
- soft_failure — A flag indicating, if set, that failure of an optimization pass should not result in termination of the optimization.
- n_evaluations — The number of allowed evaluations of the optimization function. If simplex optimization is used, this is the number of allowed evaluations per pass.

- `n_passes` — The number of optimization passes made to achieve convergence ("simplex" only). A pass ends (roughly) when the number of evaluations is completed or the function doesn't change within the tolerance. A new pass involves starting the optimization again using step sizes determined from the range of the simplex and the factor `simplex_pass_factor`.

- `n_restarts` — The number of complete restarts of the optimization (simplex only). This is an additional loop around the `n_passes` loop. The difference is that a restart involves using the optimized result but the original step sizes. It is highly recommended that this feature be used if convergence problems are seen.

- `restart_reset_threshold` — If positive, then before a restart, the change in each optimization variable relative to the starting value is compared to the difference between the upper and lower limits of the variable. If that ratio is less than the given threshold, the variable's value is set to the starting value. This can help prune small, spurious changes that have little effect on the outcome.

- `restart_worst_term_factor`, `restart_worst_terms` — Often when there are convergence problems, it is because a few terms are causing difficulty. Convergence can often be obtained by *increasing* the weighting of these terms. If `restart_worst_term_factor` is positive, then `elegant` will multiply the weight of the `restart_worst_terms` largest terms by this factor at the beginning of a restart.

- `matrix_order` — Specifies the highest order of matrix elements that should be available for fitting. Elements up to third order are available for the terminal point of the beamline, and up to secod order for interior fit points. Names for first-, second-, and third-order elements are of the form R*ij*, T*ijk*, and U*ijkl*.

- `log_file` — A file to which progress reports will be written as optimization proceeds. For SDDS data, use the `final` output file from the `run_setup` namelist.

- `term_log_file` — This names a file to which the values of the optimization terms are written at the completion of optimization, which can be convenient when large numbers of terms are used. For example, by using `sddssort` one could find which terms are contributing most to the penalty value.

- `output_sparsing_factor` — If set to a value larger than 0, results in sparsing of output to the "final" file (see `run_setup`). This can make a significant difference in the optimization speed.

- `balance_terms` — If nonzero, then all terms of the optimization expression have their weights adjusted so they make equal contributions to the penalty function. This can help prevent optimization of a single term at the expense of others. It is performed only for the initial value of the optimization function.

- `simplex_divisor` — The factor by which simplex step sizes are changed as the optimization algorithm searches for a valid initial simplex.

- `simplex_pass_range_factor` — When starting a new pass, the simplex optimizer takes the range over the previous simplex of each variable times this factor as the starting step size for that variable. This can be useful if the optimization brings the system close to an instability. In such a case, the simplex routine may have trouble constructing an initial simplex if the range of the variables is large. Setting this control to a value less than 1 may help.

- `include_simplex_1d_scans` — If nonzero, optimizer performs single-variable scans prior to starting simplex optimization. This is usually a good idea, but in some cases it will cause problems. For example, if your design is on the edge of being unstable, you may get some many errors from the initial steps that the single-variable optimizer can't continue. Disabling the single-variable scans will sometimes solve this.
- `start_from_simplex_vertex1` — If nonzero, optimizer uses the initial simplex vertex as the starting point for each new 1d scan. Otherwise, it uses the result of the previous scan.
- `rcds_step_factor` — Gives the step sizes as a fraction of the range of each variable. If non-zero, overrides the step sizes given in the `optimization_variable` commands.
- `restart_random_numbers` — If nonzero, the random number generators used by `elegant` are reset for each evaluation of the optimization function. This is valuable if one is optimizing tracking results that involve random processes (e.g., ISR or scattering).
- `interrupt_file` — Gives the name of a file that will be monitored by the program as it runs. If the file is created or modified while optimization is running, the optimizer will complete the present step and cleanly terminate, allowing subsequent commands, if any, to proceed.
- `interrupt_file_check_interval` — The interval in seconds between checking the interrupt file. N.B.: Depending on the responsiveness of the file system and the time required for a function evaluation, setting this to a small value could have a significant adverse impact on the run time.

7.48 parallel_optimization_setup

- type: setup command (for Pelegant only).
- function: define overall parallel optimization parameters and methods.
- N.B.: In addition to the optimization parameters used in the optimization_setup command, several new parameters are added for parallel optimization. User should replace optimization_setup with parallel_optimization_setup and append necessary parameters.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&parallel_optimization_setup
    STRING method = "simplex";
    double hybrid_simplex_tolerance = -0.01;
    double hybrid_simplex_tolerance_count = 2;
    long hybrid_simplex_comparison_interval = 0;
    double random_factor = 1
    long n_iterations = 10000;
    long max_no_change = 10000;
    long population_size = 100;
    STRING population_log = NULL;
    long print_all_individuals = 0;
    long output_sparsing_factor = 1;
    STRING crossover = "twopoint";
    STRING simplex_log = NULL;
    long simplex_log_interval = 1;
&end
```

- `method` — May be one of "genetic", "hybridsimplex" or "swarm". If the default "simplex" method is chosen, all the processors will do the same optimization as the serial version if there is only one particle for optimization tracking, or do optimization tracking in parallel if the number of particles is larger than the number of CPUs. All algorithms can be used for global optimization. "swarm" is recommended when there is sufficient computation resource available, so it can reach the optimization target fast. "hybridsimplex" is recommended when the initial point is close to the optimal result. "genetic" can be chosen for a global optimizer with a random start point (0 should be avoided for any initial coordinate).
- `random_factor` — The factor to scale the step size for both parallel swarm and genetic methods.
- `n_restarts` — For the parallel "hybridsimplex" method, this number should be set larger than 1, so the the best result across all processors can be used for the next restart. The parameter is not used for the swarm method.
- `hybrid_simplex_tolerance` , `hybrid_simplex_tolerance_count` — For the parallel "hybridsimplex" method, these set, respectively, the tolerance value for changes between full iterations (restarts). If the result does not improve by more than `hybrid_simplex_tolerance` after `hybrid_simplex_tolerance_count` iterations, the optimization terminates.

- `hybrid_simplex_comparison_interval` — For the parallel "hybridsimplex" method, sets the interval between comparisons of progress among the several optimizations, in units of function evaluations. Once any of the optimizations is below the `target` value, all optimizations are sent an abort command. Ignored if zero or negative, in which case all optimizations run to completion. Depending on the time required to perform a single function evaluation, setting this to a small value may *increase* the required run time due to the overhead of frequent interprocessor communication.
- `simplex_log` — For the parallel "hybridsimplex" method, rootname for files to which data from each simplex optimization will be written. Intended only for debugging as it will adversely impact performance.
- `simplex_log_interval` — Interval at which `simplex_log` files will be updated.
- `n_iterations` — The maximal number of generations/iterations for the parallel genetic and particle swarm optimization.
- `population_size` — The number of individuals to be generated for each generation/iteration for the swarm and genetic method. For the hybridsimplex method, the number of individuals is equal to the number of processors used.
- `max_no_change` — The maximal number of generations in which no change in the best evaluation is allowed before the genetic method stops (genetic method only).
- `n_evaluations` — This is not used as a stop condition in the genetic optimization. The n_iterations or max_no_change can be used instead. For the hybridsimplex method, this is the number of allowed evaluations per restart.
- `population_log` — An SDDS file to which the best individual in a population can be written after each iteration as optimization proceeds. Recommended value: "%s.pop". For the parallel genetic method, user can choose to print out all the individuals (See print_all_individuals).
- `print_all_individuals` — If nonzero, all the strings in a population will be recorded in the population_log file. This is supported for the genetic method only.
- `output_sparsing_factor` — For genetic optimization, this is used to set the frequency of printing strings in the log file with the number of generations as the interval.
- `crossover` — For genetic optimization, it allows the user to choose a crossover type from "onepoint", "twopoint" and "uniform". "twopoint" is the default crossover type. If the dimension is 2, it will be set to onepoint crossover.

Note:

- Genetic optimization in `Pelegant` terminates when at least one of the stopping rules specified has been met. The two stopping rules are:
  - generation limit (n_iterations) exceeded
  - no change in the best solution found in a given number of generations. The default is to stop when the generation limit (10000 is the default value) is reached. While the max_no_change is more favorite to use, as it will stop until the result can not be improved after a certain number of iterations (10000 is the default value). The n_iterations can be set to a very large number to use this rule as the stop condition alone.

- step size control – The mutation step size in the genetic optimization is selected from a Gaussian distribution with mean 0 and standard deviation step_size, where step_size is provided by user. All the dimensions will use the same standard deviation for an iteration. The step_size of the first dimension provided by user will be used as the original step size for all the dimensions. The step size will be reduced by the golden ratio (1.618) if the best value is unchanged after every 3000 iterations. After every 3000 iterations since the last time the step size is reduced, the step size will be increased by the golden ratio.

- As the genetic optimization implementation in `Pelegant` internally updates individuals with a relative change of the current value for a variable, 0 should be avoided to use as an initial value.

7.49 optimization_term

- type: setup command.
- function: define optimization equation via individual terms
- sequence: must follow `optimization_setup` and precede beam definition (`bunched_beam` or `sdds_beam`).
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&optimization_term
    STRING term = NULL;
    double weight = 1.0;
    STRING field_string = NULL;
    long field_initial_value = 0;
    long field_final_value = 0;
    long field_interval = 1;
    STRING input_file = NULL;
    STRING input_column = NULL;
    long verbose = 0;
&end
```

- `term` — An `rpn` expression giving one term to be optimized. If more than one `optimization_term` command is given, then the terms are added. The advantage of using this command over giving an equation via `optimization_setup` is that `elegant` will report the value of each term as it performs the optimization (if a `log_file` is given to `optimization_setup`). This permits determination of which terms are causing problems for the optimization.

  Please see the entry for `equation` under `optimization_setup` for details on designing optimization terms.
- `weight` — The weight to assign to this term. If zero, the term is ignored.
- `field_string`, `field_initial_value`, `field_final_value`, `field_interval` — These parameters are used to perform substitution of a series of values into the string given by `term`. This can be used to make an identical constraint at a number of instances of the same marker. For example, to constraint `Cx` to zero at instances 1, 3, 5, ..., 39, of marker `M1`, one could use

  ```
  &optimization_term
    term = "M1#@.Cx sqr",
    field_string = @,
    field_initial_value = 1, field_final_value = 39, field_interval = 2
  &end
  ```
- `input_file`, `input_column` — If given, `input_file` is taken as the name of an SDDS file, which is expected to have a string column named by `input_column`. Each row of the column is taken as a separate optimization term.
- `verbose` — If nonzero, optimization terms are echoed to the terminal as they are created or read from the input file.

7.50 optimization_variable

- type: setup command.
- function: defines a parameter of an element to be used in optimization.
- sequence: must follow `optimization_setup` and precede beam definition (`bunched_beam` or `sdds_beam`).
- N.B.: It is not possible to optimize an element if the element name starts with one of the following characters: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., +, or -. The reason is that `elegant` will attempt to make an SDDS parameter name containing the element name, and these characters are disallowed at the beginning of such a name.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&optimization_variable
    STRING name = NULL;
    STRING item = NULL;
    double lower_limit = 0;
    double upper_limit = 0;
    long differential_limits = 0;
    double step_size = 1;
    double fractional_step_size = -1;
    long disable = 0;
    long force_inside = 0;
    long no_element = 0;
    double initial_value = 0;
&end
```

- `name` — The name of the element.
- `item` — The parameter of the element to be varied.
- `lower_limit`, `upper_limit` — The lower and upper limits allowed for the parameter. If these are equal, the range of the parameter is unlimited.
- `differential_limits` — If nonzero, then the lower and upper limits are being given relative to the initial value, rather than in absolute terms.
- `step_size`, `fractional_step_size` — Specifies the initial step size ("simplex" optimization) or the grid size in this dimension ("grid" or "sample" optimization). If `fractional_step_size` is positive, it is used in place of `step_size`; in that case, the step size is determined by using the indicated fraction of the difference between `upper_limit` and `lower_limit`.
- `disable` — If nonzero, the variable is ignored.
- `force_inside` — If nonzero, the initial value is forced inside the allowed range defined by the `lower_limit` and `upper_limit` parameters.

- no_element, initial_value — Allows defining a variable that is not connected to a beamline element, and giving the initial value for the variable. The variable can them be used in other optimization-related commands, e.g., optimization_covariable.

7.51 print_dictionary

- type: action command.
- function: print dictionary of supported accelerator elements.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&print_dictionary
    STRING filename = NULL;
    long SDDS_form = 0;
&end
```

- `filename` — The name of a file to which the dictionary will be written. By default, the output is in LaTeXformat.
- `SDDS_form` — If non-zero, then the output is in SDDS format.

7.52 ramp_elements

- type: setup command.
- function: define parameters for time-dependent ramping of elements
- sequence: must follow `run_setup`.
- N.B.: if the ramped element is modeled with a matrix, a significant performance hit may be seen. It is best to use symplectic variants of the elements, since these don't invoke the matrix calculation.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&ramp_elements
    STRING name = NULL;
    STRING item = NULL;
    STRING type = NULL;
    long start_pass = 0;
    long end_pass = LONG_MAX;
    double start_value = 0;
    double end_value = 0;
    long refresh_matrix = 0;
    long differential = 1;
    long multiplicative = 0;
    long start_occurence = 0;
    long end_occurence = 0;
    double exponent = 1;
    double s_start = -1;
    double s_end = -1;
    STRING before = NULL;
    STRING after = NULL;
    long verbose = 0;
    STRING record = NULL;
&end
```

N.B.: This command will produce unpredictable results when used with `error_element`, `alter_elements`, `modulate_elements`, and `load_parameters` (except when `change_defined_values=1`). It will also not work well if matrix concatenation is invoked. It should work properly with `link_elements` in turn-by-turn mode when the source element is ramped, but not when the target element is ramped.

- `name` — A possibly-wildcard-containing string giving the names of the elements to modulate. If not specified, then one must specify `type`.
- `item` — The name of the parameter to modulate.
- `type` — A possibly-wildcard-containing string giving the names of element *types* to modulate. May be specified with `name` or by itself.

- `start_pass`, `end_pass` — The starting and ending pass, i_{start} and i_{end} for the ramp. For passes less than `start_pass`, the ramp value is `start_value`. For passes greater than `end_pass`, the ramp value is `end_value`.
- `start_value`, `end_value` — The end-point values S (start) and E (end) of the ramp.
- `exponent` — The exponent p for the variation of values between the start and end of the ramp. The ramp function $R(i)$ is

$$R(i) = S + (E - S) * \left(\frac{i - i_{\mathrm{start}}}{i_{\mathrm{end}} - i_{\mathrm{start}}}\right)^p. \quad (10)$$

  Note that $i = 0$ on the first pass.
- `refresh_matrix` — Frequently there is a matrix associated with an element even if tracking through the element does not use the matrix. In this case, `elegant` doesn't normally update the matrix for the element as it modulates the element, since that may involve a significant time penalty. If this parameter is set to a non-zero value, the matrix will be updated. For elements that use a matrix for tracking, the matrix is always updated.
- `differential`, `multiplicative` — Determine how the amplitude function $A(t)$ is used to obtain the new value of the parameter. There are four cases
  - `differential=1`, `multiplicative=0`: $v(t) = v_0 + R(i)$ (default).
  - `differential=0`, `multiplicative=0`: $v(t) = R(i)$.
  - `differential=1`, `multiplicative=1`: $v(t) = v_0 + v_0 R(i)$.
  - `differential=0`, `multiplicative=1`: $v(t) = v_0 R(i)$.
- `start_occurence`, `end_occurence` — If nonzero, these give the starting and ending occurrence numbers of elements that will be modulated. N.B.: if wildcards are used, occurrence number counting is for each set of identically-named elements separately, rather than for the sequence of matched elements.
- `s_start`, `s_end` — If non-negative, these give the gaving and ending position limits for the end-of-element locations of elements to be modulated.
- `after` — The name of an element. If given, the modulation is applied only to elements that follow the named element in the beamline.
- `before` — The name of an element. If given, the modulation is applied only to elements that precede the named element in the beamline.
- `verbose` — If nonzero, information is printed to the standard output as changes are made. Use for debugging only, since otherwise it may slow the simulation.
- `record` — Gives a possibly incomplete filename to which will be written a record of the values of the ramp.

7.53 rf_setup

- type: setup/action command.
- function: set up rf cavity frequency, phase, and voltage for a storage ring
- sequence: must follow `run_setup`. In action mode, must follow action-mode instance of `twiss_output`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&rf_setup
    STRING filename = NULL;
    STRING name = NULL;
    long start_occurence = -1;
    long end_occurence = -1;
    double s_start = -1;
    double s_end = -1;
    long set_for_each_step = 0;
    double near_frequency = 0;
    long harmonic = -1;
    double bucket_half_height = 0;
    double over_voltage = 0;
    double total_voltage = 0;
    long disable = 0;
    long output_only = 0;
    long track_for_frequency = 0;
&end
```

This command must follow a `twiss_output` command that includes radiation integral computation, since the energy loss per turn is needed to set up the rf cavities. Note that the command includes features to allow selecting a subset of the RFCA elements in the beamline. The selected subset is assumed to include all of the cavities that will impart net energy to the beam.

This command stores values for bunch length in symbols `Sz0` and `St0`, and also stores the fractional energy spread in `Sdelta0`, where they can be used in rpn expressions in subsequent commands, e.g.,

```
&bunched_beam
 sigma_dp = "(Sdelta0)",
 sigma_s = "(Sz0)",
 ...
&end
```

If performing rf setup in the presence of a non-zero orbit, the best procedure is to run `rf_setup` twice, in a sequence like the following

```
...
! Compute nominal radiation integrals
&twiss_output
```

```
        radiation_integrals = 1
&end
! Perform nominal rf cavity setup
&rf_setup
        name = *,
        harmonic = 1296
        total_voltage = 9e6
&end
! Set up to compute the closed orbit, which may depened on radiation losses and rf cavity location
&closed_orbit
        output = %s.clo
&end
! Set up to compute twiss parameters and radiation integrals
&twiss_output
        filename = %s.twi
        radiation_integrals = 1
        output_at_each_step = 1
&end
! Set up to perform rf cavity setup
&rf_setup
        filename = %s.rf
        name = *,
        harmonic = 1296
        total_voltage = 9e6
        set_for_each_step = 1
&end
...
```

- `filename` — Name of a file to which data related to the rf settings will be written.
- `name` — A possibly-wildcard-containing string giving the names of the elements to set. If not given, all RFCA elements are selected.
- `start_occurence`, `end_occurence` — If nonzero, these give the starting and ending occurrence numbers of elements that will be set.
- `s_start`, `s_end` — If non-negative, these give the gaving and ending position limits for the end-of-element locations of elements to be set.
- `set_for_each_step` — If nonzero, then the setup is repeated at each simulation step. In this case, one must also give `output_at_each_step=1` for `twiss_output`.
- `near_frequency` – If nonzero, then the rf frequency is chosen to be the closest harmonic to the given frequency.
- `harmonic` — If nonzero, then the rf frequency is set to the given harmonic of the revolution frequency.
- `bucket_half_height` — If nonzero, the voltage is computed so as to give the specified bucket half height.

$$\left(\frac{\Delta p}{p}\right)_{\mathrm{bucket}} = \sqrt{\frac{U_0}{\pi \alpha h E}} \sqrt{F(q)}, \tag{11}$$

where U_0 is the energy loss per turn, α is the momentum compaction factor, h is the harmonic, E is the beam energy,

$$F(q) = 2\left(\sqrt{q^2-1} - \arccos\frac{1}{q}\right), \tag{12}$$

and q is the overvoltage factor, related to the rf voltage by $q = V/U_0$. (See Wiedemann, Vol. 1, 8.2.2.)

- `over_voltage` — If nonzero, the voltage is set to the given factor relative to the energy loss per turn.
- `total_voltage` — If nonzero, the total rf voltage is set to the given value. The frequency and phase are computed for this voltage.
- `disable` — If nonzero, command does nothing.
- `output_only` — If nonzero, command generates output file but does not change rf cavity settings.
- `track_for_frequency` — If nonzero, particle tracking is used to determine the rf frequency. If zero, the ideal length of the lattice is used.

7.54 replace_elements

- type: action command.
- function: Replace old element with a newly defined element, or just remove it from beamline. This is a convenient way to modify lattice in an elegant run. See also `transmute_elements`.
- sequence: must follow `run_setup`.
- notes: The modified lattice can be saved through `save_lattice` command. Be sure to use "output_seq = 1" option in that command.
- warning: The element's occurrence is re-calculated after each usage of this command. If you need to repeat this command for SAME named element several times, you have to re-calculate it occurrence every time. For example, you want to remove Q1 at occurrence position (1,3,5), and use 'replace_elements" twice. If in the first command you use "occurence[0]=1,3", then in the second command you have to use "occurence[0]=3", since after remove of (1,3) Q1s, the 5th Q1 now becoming 3rd Q1.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&replace_elements
        STRING name = NULL;
        STRING type = NULL;
        STRING exclude = NULL;
        long skip = 1;
        long disable = 0;
        STRING element_def = NULL;
        long total_occurrences = 0;
        long occurrence[100]={0};
&end
```

- `name` — Possibly wild-card containing string specifying the name of the elements to be removed or replaced.
- `type` — Possibly wild-card containing string specifying the type of the elements to be removed or replaced.
- `exclude` — Possibly wild-card containing string specifying the name of elements to be excluded from this command.
- `skip` — The element is removed or replaced at every n^{th} specified location.
- `disable` — If nonzero, the command is ignored.
- `element_def` — If NULL, the specified elements are removed from the beamline. If not NULL, the specified elements are replaced with the new element defined here. The definition of the element should be just as it would be entered in the lattice file.

- `total_occurrences`, `occurrence` — These parameters are used to replace or delete specified occurrences of the element `name`. `total_occurrences` specifies how many elements to replace or delete up to a maximum of 100, while the entries in the array `occurrence` specify the occurrences to replace or delete. If `total_occurrences` is non-zero, then **skip** must be set to zero and the **name** must be the exact name (no wild-card matching).

7.55 rpn_expression

- type: action/setup command.
- function: pass an expression directly to the rpn submodule for execution.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&rpn_expression
    STRING expression = NULL;
&end
```

- `expression` — An `rpn` expression. This expression is executed immediately and can be used, for example, to read in `rpn` commands from a file or store values in `rpn` memories.

7.56 rpn_load

- type: action/setup command.
- function: load data from SDDS file into RPN variables.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&rpn_load
    STRING tag = NULL;
    STRING filename = NULL;
    STRING match_column = NULL;
    STRING match_column_value = NULL;
    long matching_row_number = -1;
    STRING match_parameter = NULL;
    STRING match_parameter_value = NULL;
    long use_row = -1;
    long use_page = -1;
    long load_parameters = 0;
&end
```

This command is used to facilitate multi-stage optimization runs by allowing convenient loading of data from SDDS files into RPN variables. For example, one may match the final Twiss parameters of a lattice to the parameters stored in an SDDS file from a different run.

- `tag` — Option string that will be pre-pended to the names of all the numerical columns in the file in order to create RPN variable names. E.g., if the input file was from the `twiss_output` command and `tag = tw1` was given, then RPN variables `tw1.betax`, `tw1.alphax`, etc. would be used. *N.B.: If the tag is blank, then nothing is appended to the names from the file. This can be dangerous since the names may conflict with the names of other variables!*
- `filename` — The (incomplete) name of the SDDS file from which to read data. By default, data is taken from all columns from the last row of the last page of the file. This default behavior can be altered using one or more of the following parameters:
  - `match_column` — The name of a string column to use in selecting the row from which data will be taken.
  - `match_column_value` — The value that the column named by `match_column` must have to be selected from the file. By default, the last row with a matching value is used.
  - `matching_row_number` — If a nonnegative value is given, then the `matching_row_number`th matching row is selected (0 is the first row, 1 the second, etc). Otherwise, the last match row is used. Ignored if `match_column` is not given.
  - `match_parameter` — The name of a string parameter to use in selecting the page from which data will be taken.
  - `match_parameter_value` — The value that the parameter named by `match_parameter` must have to be selected from the file. By default, the last page with a matching value is used.

- use_row — If nonnegative, specifies the row number to use, starting at 0 for the first row. Ignored if match_column is given.
- use_page — If nonnegative, specifies the page number to use, starting at 1 for the first page. Takes precedence over \match_parameter if both are given.
- load_parameters — If nonzero, specifies loading the SDDS parameter data rather than the column data.

7.57 run_control

- type: setup command.
- function: set up the number of simulation steps and passes.
- sequence: must follow `run_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&run_control
    long n_steps = 1;
    double bunch_frequency = 0;
    long n_indices = 0;
    long n_passes = 1;
    long n_passes_fiducial = 0;
    long reset_rf_for_each_step = 1;
    long first_is_fiducial = 0;
    long restrict_fiducialization = 0;
&end
```

- `n_steps` — The number of separate repetitions of the action implied by the next action command. If random errors are defined, this is also the number of separate error ensembles.
- `bunch_frequency` — The frequency to use in calculating the time delay between repetitions.
- `n_indices` — The number of looping indices for which to expect definitions in subsequent `vary_element` commands. If nonzero, then `n_steps` is ignored.
- `n_passes` — The number of passes to make through the beamline per repetition.
- `n_passes_fiducial` — The number of passes to make through the beamline per repetition for the fiducial beam. If non-positive, use `n_passes`. For ring tracking, should probably always be set to 1.
- `reset_rf_for_each_step` — If nonzero, the rf phases are established anew for each bunch tracked. Should be zero to simulate phase and timing jitter.
- `first_is_fiducial` — If nonzero, the first bunch seen is taken to establish the reference phases and momentum profile. If zero, each bunch is treated as a new fiducializing bunch.
- `restrict_fiducialization` — If nonzero, then momentum profile fiducialization occurs only after elements that are intended change the momentum, such as rf cavities. If zero, then each element is fiducialized to the average momentum of the beam. Active only if `first_is_fiducial=1` and overrides the `always_change_p0` setting in `run_setup`.

7.58 run_setup

- type: setup command.
- function: set global parameters of the simulation and define primary input and output files.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&run_setup
    STRING lattice = NULL;
    STRING use_beamline = NULL;
    STRING rootname = NULL;
    STRING output = NULL;
    STRING centroid = NULL;
    STRING bpm_centroid = NULL;
    STRING sigma = NULL;
    STRING final = NULL;
    STRING acceptance = NULL;
    STRING losses = NULL;
    long losses_include_global_coordinates = 0;
    double losses_s_limit[2] = {-DBL_MAX, DBL_MAX};
    STRING magnets = NULL;
    STRING profile = NULL;
    STRING semaphore_file = NULL;
    STRING parameters = NULL;
    long suppress_parameter_defaults = 0;
    STRING rfc_reference_output = NULL;
    long combine_bunch_statistics = 0;
    long wrap_around = 1;
    long final_pass = 0;
    long default_order = 2;
    long concat_order = 0;
    long print_statistics = 0;
    long show_element_timing = 0;
    long monitor_memory_usage = 0;
    long random_number_seed = 987654321;
    long correction_iterations = 1;
    double p_central = 0.0;
    double p_central_mev = 0.0;
    long always_change_p0 = 0;
    STRING expand_for = NULL;
    long tracking_updates = 1;
    long echo_lattice = 0;
    STRING search_path = NULL;
    long element_divisions = 0;
    long load_balancing_on = 0;
    long back_tracking = 0;
```

```
    double s_start = 0;
&end
```

- `lattice` — Name of the lattice definition file.
- `echo_lattice` — If nonzero, the lattice input is echoed to the standard output as the lattice is parsed. This can help detect certain problems with the lattice that cause `elegant` to crash.
- `use_beamline` — Name of the beamline to use.
- `rootname` — Filename fragment used in forming complete names from incomplete filenames. By default, the filename minus extension of the input file is used.
- `output` — The (incomplete) name of an SDDS file into which final phase-space coordinates will be written. Recommended value: "%s.out".
- `centroid` — The (incomplete) name of an SDDS file into which beam centroids as a function of s will be written. Recommended value: "%s.cen".
- `bpm_centroid` — The (incomplete) name of an SDDS file into which beam centroids at beam position monitors only as a function of s will be written. Recommended value: "%s.bpmcen".
- `sigma` — The (incomplete) name of an SDDS file into which the beam sigma matrix as a function of z will be written. Recommended value: "%s.sig". N.B.: confusion sometimes occurs about some of the quantities related to the `s` coordinate in this file. Please see Section 4 above.
- `final` — The (incomplete) name of an SDDS file into which final beam and transport parameters will be written. Recommended value: "%s.fin". N.B.: confusion sometimes occurs about some of the quantities related to the `s` coordinate in this file. Please see Section 4 above.
- `acceptance` — The (incomplete) name of an SDDS file into which the initial coordinates of transmitted particles will be written. Recommended value: "%s.acc".
- `losses` — The (incomplete) name of an SDDS file into which information on lost particles will be written. Recommended value: "%s.lost".
- `losses_include_global_coordinates` — If nonzero, the losses output file includes the global coordinates of lost particles.
- `losses_s_limit[2]` — Specifies the minimum and maximum `s` coordinate for logging of lost particles.
- `magnets` — The (incomplete) name of an SDDS file into which a magnet layout representation will be written. Recommended value: "%s.mag".
- `magnets` — The (incomplete) name of an SDDS file into which (s, ρ, K_1, K_2, K_3) data will be written. Recommended value: "%s.pro".
- `semaphore_file` — The (incomplete) name of file that will be created just before exit from the program, but only if no errors occured. If the file exists, it is deleted. This file can be used to record the fact that the run completed without error.

- parameters — The (incomplete) name of an SDDS file into which parameters of accelerator elements are written.
- suppress_parameter_defaults — If non-zero, then the parameters output file will not contain rows for parameters whose values are identical to the then-current default values. This can result in significantly smaller parameter files and faster loading. One downside is that future changes to defaults would possible result in difficulty reproducing a result from a saved parameter file. Another, more serious, risk is that the parameter file may not reflect changes made via, e.g., alter_elements or load_parameters if those changes *restore* default values for quantities that have non-default values in the lattice file. Users are warned to use this feature with caution.
- rfc_reference_output — The (incomplete) name of an SDDS file into which the internally-determined reference times for RFCA and RFCW elements are written. This file can be loaded with load_parameters to exactly reproduce cavity phasing, e.g., for backtracking.
- combine_bunch_statistics — A flag indicating whether to combine statistical information for all simulation steps. If non-zero, then the sigma and centroid data will be combined over all simulation steps.
- wrap_around — A flag indicating whether the s coordinate should wrap-around or increase monotonically in multipass simulations. If zero, then the centroid and sigma data is computed for each turn with the s coordinate increasing continuously.
- final_pass — A flag indicating whether the centroid and sigma output should be computed only from the data from the final pass. By default, the statistics include data from all passes.
- default_order — The default order of transfer matrices used for elements having matrices.
- concat_order — If non-zero, the order of matrix concatenation used. Rarely needed, but may increase speed at the expense of accuracy.
- print_statistics — A flag indicating whether to print information as each element is tracked. If greater than 0, information is printed after each element from the beginning of tracking. If equal to n with $n < 0$, information is printed only after pass $|n|$.
- show_element_timing — A flag indicating whether to collect and report execution time statistics binned by element type.
- monitor_memory_usage — A flag indicating whether to monitor memory usage during tracking to detect memory leaks.
- random_number_seed — A seed for the random number generators. If zero, a seed will be generated from the system clock.
- correction_iterations — Number of iterations of orbit, tune, and chromaticity correction. Setting this to a value larger than 1 will cause orbit, tune, and chromaticity correction to be repeated, which improves overall convergence.
- p_central — Central momentum of the beamline, about which expansions are done. This is $\beta\gamma$.

- p_central_mev — Central momentum of the beamline in MeV/c, about which expansions are done. Ignored if p_central is nonzero.
- always_change_p0 — If nonzero, then elegant will match the reference momentum to the beam momentum after each element. For example, in a beamline with radiation losses, one might want to adjust downstream magnets to match the energy of the incoming beam.
- expand_for — Name of an SDDS file containing particle information, from which the central momentum will be set. The file contents are the same as required for elegant input with the sdds_beam namelist.
- tracking_updates — A flag indicating whether to print summary information about tracking.
- search_path — Specify a list of pathnames in which to look for input files, including lattice files, wakefield input, particle input, etc. This allows storing common input files in a convenient location without having to put the location into every filename.
- element_divisions — Specify how many pieces to split elements into. Only certain elements (basically, those with a matrix) are split. Results in creation of element_divisions new elements having the same name as each split element.
- load_balancing_on — If 1, load-balancing is performed for parallel mode. This can result in non-deterministic results if the load-balancing is different on two otherwise identical runs. Load-balancing variations may occur in heterogeneous clusters, clusters with multiple users, or for other reasons. In such situations, turning off load balancing can be useful if, for example, one is performing parameter scans and wishes to eliminate spurious sources of variation. If -1, then the load balance is checked and reported, but no rebalancing takes place.
- back_tracking — If nonzero, then back-tracking is performed. The beamline is reversed in order and the beam is propagated backwards through the elements. Only a selection of elements are supported at present, including CHARGE, CSBEND, DRIF, EDRIFT, EHCOR, EHVCOR, EVCOR, HMON, KOCT, KQUAD, KSEXT, MARK, MONI, QUAD, RFCA, SBEN, SEXT, TRWAKE, UKICKMAP, VMON, WAKE, and WATCH.
- s_start — The initial s coordinate. Changing this will affect not only output data (e.g., the s values in files from twiss_output and floor_coordinates), but also the behavior of commands (e.g., alter_elements) the provide s-based filtering.

7.59 sasefel

- type: setup/action command.
- function: set parameters for computation of SASE FEL gain and other properties.
- sequence: must follow `run_setup` and precede beam definition (`bunched_beam` or `sdds_beam`).
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&sasefel
    STRING output = NULL;
    STRING model = "Ming Xie";
    double beta = 0;
    double undulator_K = 3.1;
    double undulator_period = 0.033;
    double slice_fraction = 0.0;
    long n_slices = 0;
&end
```

- `output` — The (incomplete) filename of an SDDS file to which output will be written.
- `model` — The name of the FEL model used. At present, only one model is supported; the "Ming-Xie" model is based on the simple parametrization M. Xie[13].
- `beta` — The value of the beta function, in meters.
- `undulator_K` — The K parameter of the undulator.
- `undulator_period` — The undulator period, in meters.
- `slice_fraction`, `n_slices` — The fraction of beam beam contained by each analysis slice and the number of such slices. By default, no slice analysis is done. Instead, the beam is analyzed only as a whole. If `slice_fraction`*`n_slices` is less than 1, then the slice analysis is centered on the median of the time distribution. E.g., if `n_slices`=1 and `slice_fraction`=0.1, then the central 10% of the beam would be analyzed. More typically, one gives values such that `slice_fraction`*`n_slices` is equal to 1, so that every part of the beam is analyzed. There are separate values in the output file for each slice, plus the whole-beam and slice-averaged results.

7.60 save_lattice

- type: action command.
- function: save the current accelerator element and beamline definitions.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&save_lattice
    STRING filename = NULL;
    long output_seq = 0;
    long suppress_defaults = 1;
&end
```

- `filename` — The (incomplete) name of a file to which the element and beamline definitions will be written. Recommended value: "%s.new".
- `output_seq` — If non-zero, the lattice will be saved as a single beamline sequence, in one of two ways: For `output_seq=1`, the sequence is contained in a series of 40-element sub-beamlines. For `output_seq=2`, the sequence is contained in a single long beamline definition. Elements used for the beamline are re-arranged according to their type. Note: sub-beamline definitions in the original lattice file will be destroyed from the output file. This feature is intended to be used together with `insert_elements` and `replace_elements`.
- `suppress_defaults` — If non-zero, the lattice will not contain default values of parameters.

7.61 sdds_beam

- type: setup command.
- function: set up for tracking and histogram analyzing of particle coordinates stored in an SDDS file.
- sequence: must follow `run_control`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&sdds_beam
    STRING input = NULL;
    STRING input_list = NULL;
    STRING input_type = "elegant";
    long n_particles_per_ring = 0;
    STRING selection_parameter = NULL;
    STRING selection_string = NULL;
    long one_random_bunch = 0;
    long reuse_bunch = 0;
    long prebunched = -1;
    long track_pages_separately = 0;
    long use_bunched_mode = 0;
    long fiducialization_bunch = 0;
    long sample_interval = 1;
    long n_tables_to_skip = 0;
    long center_transversely = 0;
    long center_arrival_time = 0;
    double sample_fraction = 1;
    double p_lower = 0.0;
    double p_upper = 0.0;
    long save_initial_coordinates = 1;
    long reverse_t_sign = 0;
    long n_duplicates = 0;
    double duplicate_stagger[6] = {0, 0, 0, 0, 0, 0};
&end
```

- `input` — Name of an SDDS file containing coordinates of input particles.
- `input_type` — May be “elegant” or “spiffe”, indicating the name of the program that wrote the input file. The expected data quantities for the different types are:
  - `elegant`: $(\mathrm{x}, \mathrm{xp}, \mathrm{y}, \mathrm{yp}, \mathrm{t}, \mathrm{p})$, where x and y are in meters, $\mathrm{xp} = \mathrm{x}'$ and $\mathrm{xp} = \mathrm{y}'$ are dimensionless, t is in seconds, and p = fifl is the dimensionless momentum. If this file is to be generated by the user, the expected units string in the column definitions should be “m”, “s”, and “mbenc” for meters, seconds and the dimensionless momentum, respectively. The `particleID` column may also be given; it should contain a positive integer that is unique for each particle.

- spiffe: (r, z, pr, pz, pphi, t), where r and z are in meters, pr = $\text{fi}_\text{r}\text{fl}$, pz = $\text{fi}_\text{z}\text{fl}$, $\text{p}_{\text{Œ}}$ = !rfl/c, and t is in seconds. If this file is to be generated by the user use the units strings described above.

- n_particles_per_ring — For spiffe data, gives the number of particles to generate for each ring of charge.
- selection_parameter — The name of a parameter in the SDDS file to be used for selection of pages of data.
- selection_string — The value of the selection_parameter selection parameter required for a page to be used. E.g., if one has a file from the shower program containing positrons, electrons, and photons, one might want to select only the positrons.
- one_random_bunch — A flag indicating whether, for spiffe data, a new random distribution should be calculated for each step of the simulation.
- reuse_bunch — A flag indicating whether to use the bunch again or not. If set, then the first bunch in the file is used repeatedly for as many tracking steps as requested. Otherwise, each bunch is used only once and the number of steps is limited to the number of bunches (e.g., the number of pages in the file when prebunched=0).
- prebunched — Deprecated. Use track_pages_separately instead.
- track_pages_separately — If non-zero, then separate pages of the input file are tracked separately. Otherwise, the entire file is tracked together.
- use_bunched_mode — If non-zero, then the IDSlotsPerBunch parameter is used to determine the bunch assignment of particles in the beam based on values in the particleID column. In particular, the bunch number is $\lfloor (I-1)/S \rfloor$, where I is the particle ID and S=IDSlotsPerBunch.
- fiducialization_bunch — If non-negative, then rf cavities (e.g., RFCA, RFDF, RAMPRF) are phased to the indicated bunch (0 is the first bunch). Otherwise, rf cavities are phased to the entire beam (which is probably not what is wanted).
- sample_interval — If non-zero, only every sample_interval$^{\text{th}}$ particle is used.
- n_tables_to_skip — Number of SDDS pages to skip at the beginning of the file.
- center_transversely — If non-zero, the transverse centroids of the distribution are made to be zero.
- center_arrival_time — If non-zero, the mean arrival time of particles at the start of the accelerator is set to zero.
- sample_fraction — If non-unity, the randomly selected fraction of the distribution to use.
- p_lower, p_upper — If different, the lower and upper limit on fifl of particles to use.
- save_initial_coordinates — A flag that, if set, results in saving initial coordinates of tracked particles in memory. This is the default behavior. If unset, the initial coordinates are not saved, but are reread from disk each time they are needed. This is more memory efficient and is useful for tracking very large numbers of particles.

- `n_duplicates` — This specifies duplicating the particles from the input file to allow tracking more particles. `n_duplicates` specifies the number of duplications, where the default value of 0 indicates no duplication. If n-fold duplication is invoked, the particle ID of a new particle is equal to the particle ID of its parent particle plus iN_p, where $i = 1, ..., n+1$ is the duplication index and N_p is the number of particles in the parent bunch. This should be kept in mind when using the particle ID to segregate the beam into bunches.

- `duplicate_stagger` — Specifies offsetting of the coordinates x, x', y, y', t, and δ for each duplication by the specified amounts. One assumes that some stochastic process such as synchrotron radiation will cause further differentiation of duplicate particles. One can also use `SCATTER` or `DSCATTER` elements in the beamline for this purpose.

7.62 semaphores

- type: setup command.
- function: set up names for semaphore files, which are used to mark the start and end of program execution.
- sequence: must precede `run_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&semaphores
        STRING started = ``%s.started'';
        STRING done    = ``%s.done'';
        STRING failed  = ``%s.failed'';
&end
```

- `started` — Gives the (incomplete) filename of a file to create when a valid `run_setup` command is given.
- `done` — Gives the (incomplete) filename of a file to create when the program exits without error. If the file exists, it is deleted when a valid `run_setup` command is given.
- `failed` — Gives the (incomplete) filename of a file to create when the program exits with an error. If the file exists, it is deleted when a valid `run_setup` command is given.

7.63 set_reference_particle_output

- type: setup command.
- function: Allows defining a reference set of particle coordinates to which tracked coordinates will be compared for purposes of optimization.
- sequence: must follow `optimization_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- Usage notes: The purpose of this command is to allow optimization of a transport system to produce the same particle distribution as was obtained by tracking through some other system. For example, one might track a collection of particles through a `CWIGGLER` or `BGGEXP` element, then attempt to match the output particles with a different element or set of elements that offer faster tracking. In this case, the optimization run must use the same input distribution as the run that is being matched.

```
&set_reference_particle_output
        STRING match_to = NULL;
        double weight[6] = {1, 1, 1, 1, 0, 1};
        STRING comparison_mode = NULL;
&end
```

- `match_to` — Name of an SDDS file from which a particle distribution will be read. The coordinates of this distribution will be compared to those from tracking to compute a contribution to the optimization penalty function.
- `weight` — Weight to be assigned to each plane. By default, path-length coordinates are not compared.
- `comparison_mode` — May be one of "max-ad", "sum-ad", and "sum-sqr", corresponding to maximum absolute deviation, sum of absolute deviations, and sum of squared deviations. The default is maximum absolute deviation.

7.64 slice_analysis

- type: setup command.
- function: set parameters for slice analysis of the beam along a beamline. Also, results in placing the final slice analysis (at the end of the beamline) in symbols for use in optimization equations. The names of the symbols are the same as the names of the columns in the output file.
- sequence: must follow `run_setup` and precede beam definition (`bunched_beam` or `sdds_beam`).
- N.B.: slice analysis uses an approximate computation of the normalized emittance, regardless of the setting of the `exact_normalized_emittance` flag in the `global_settings` command.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&slice_analysis
STRING output = NULL;
long n_slices = 0;
double s_start = 0;
double s_end = 1e300;
long final_values_only = 0;
&end
```

- `output` — The (incomplete) filename of the output file. Recommended value is "%s.slan".
- `n_slices` — Number of slices to use.
- `s_start`, `s_end` — Position in beamline at which to start and stop performing slice analysis.
- `final_values_only` — If nonzero, then slice quantities are computed only at the end of the beamline.

7.65 subprocess

- type: action command.
- function: execute a system command in a shell.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&subprocess
    STRING command = NULL;
&end
```

- `command` — The text of the command to execute. The command may use the sequence "%s" for substitution of the rootname as set by `run_setup`. A literal "%s" must be entered as "%%s".

7.66 steering_element

- type: setup command.
- function: setup for use of a given parameter of a given element as a steering corrector.
- sequence: must precede `correct`.
- N.B.: any use of this command disables the built-in definition of HKICK, VKICK, and HVKICK elements as steering elements.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&steering_element
    STRING name = NULL;
    STRING element_type = NULL;
    STRING item = NULL;
    STRING plane = "h";
    double tweek = 1e-3;
    double limit = 0;
    long start_occurence = 0;
    long end_occurence = 0;
    long occurence_step = 1;
    double s_start = -1;
    double s_end = -1;
    STRING after = NULL;
    STRING before = NULL;
&end
```

- `name` — Optional: the (possibly wild-carded) name of the element to add to the steering list. If not given, then `element_type` must be given.
- `element_type` — Optional: the (possibly wild-carded) name of the element type to add to the steering list. If not given, then `name` must be given.
- `item` — The parameter of the element to be varied.
- `plane` — May be either "h" or "v", for horizontal or vertical correction.
- `tweek` — The amount by which to change the item to compute the steering strength.
- `limit` — The maximum allowed absolute value of the item.
- `start_occurence`, `end_occurence` — If nonzero, these give the starting and ending occurence numbers of elements that will be included. N.B.: if wildcards are used, occurence number counting is for each set of identically-named elements separately, rather than for the sequence of matched elements.
- `s_start`, `s_end` — If non-negative, these give the gaving and ending position limits for the end-of-element locations of elements to be included.

- `after` — The name of an element. If given, only elements that follow the named element in the beamline are included.
- `before` — The name of an element. If given, only elements that precede the named element in the beamline are included.

7.67 touschek_scatter

- type: setup/action command.
- function: Simulate Touschek scattering process at each `TSCATTER` element based on Monte Carlo method. The local scattering rate is calculated by using Piwinski's formula and from the Monte Carlo simulation. Scattered particles can be tracked through the entire beamline for one or more passes, and beam loss information is recorded.
- sequence: must follow `run_setup` and `twiss_output`.
- can use parallel resources (`Pelegant`)
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- notes:
  - A momentum aperture file is required previous using this command. It should contain momentum aperture at least at each `TSCATTER` element and can be obtained by running `momentum_aperture` command.
  - The simulation can be done for a Gaussian distributed beam or an arbitrary particle distribution given by histogram file(s) (See `MHISTOGRAM`).
  - When using histogram file as input, it should contain data at least at each `TSCATTER` element. This can be done by inserting a `MHISTOGRAM` element following each `TSCATTER` element. With `lumped=1` option, a multi page SDDS file will be output automatically or you can combine individual output file into a multi page SDDS file before using this command.
  - The input particle distribution can be given in 3 ways: 2D(x-x')+2D(y-y')+2D(dt-dp); or 4D(x-x'-y-y')+2D(dt-dp); or 6D(x-x'-y-y'-dt-dp); base on user's choice. We recommend to use lower "order" histogram table if the original particle number which used to generate these table is not large enough.
  - The `emit_*`, `emit_dp` and `sigma_s` is always required for running the simulation (Used for Piwinski's rate). Use closed value when simulate a non-Gaussian distributed bunch.

```
&touschek_scatter
        double charge = 0;
        double frequency = 1;
        double emit_x = 0;
        double emit_nx = 0;
        double emit_y = 0;
        double emit_ny = 0;
        double sigma_dp = 0;
        double sigma_s = 0;
        double distribution_cutoff[3] = {3, 3, 3};
        double Momentum_Aperture_scale = 0.85;
        STRING Momentum_Aperture = NULL;
        STRING XDist = NULL;
```

```
        STRING YDist = NULL;
        STRING ZDist = NULL;
        STRING TranDist = NULL;
        STRING FullDist = NULL;
        STRING bunch = NULL;
        STRING loss = NULL;
        STRING distribution = NULL;
        STRING initial = NULL;
        STRING output = NULL;
        long nbins = 100;
        double sbin_step = 1;
        long n_simulated = 5000000;
        double ignored_portion = 0.01;
        long i_start = 0;
        long i_end = 1;
        long do_track = 0;
        long match_position_only = 0;
        long overwrite_files = 1;
        long verbosity = 0;
&end
```

- `charge` — Bunch charge in Coulombs. May not be zero.
- `frequency` — Bunch repetition frequency in Hz. The product of the `charge` and `frequency` gives the average current in Amps.
- `emit_x`, `emit_y` — RMS emittance for the x and y planes. Ignored if RMS normalized emittance is nonzero.
- `emit_nx`, `emit_ny` — RMS normalized emittance for the x and y planes.
- `sigma_dp`, `sigma_s` — Rms fractional momentum spread, σ_δ, and rms bunch length.
- `distribution_cutoff` — The number of sigmas to use in each plane for Gaussian beam.
- `Momentum_Aperture` — Input file containing the estimated momentum aperture at each `TSCATTER` element. This can be obtained from the `momentum_aperture` command in a separate run. (If using the parallel version to obtain the momentum aperture, it will be necessary to use `output_mode=0` or else reorganize the data if `output_mode\neq 0`. Also, it will be necessary to use `sddssort` to sort the data by the `s` column.)
- `Momentum_Aperture_scale` — This value times the aperture value from `Momentum_Aperture` file sets up the limit on δ_m in the simulation. Only particles that have $\delta > \delta_m$ will be kept for tracking. And the scattering rate is calculated at this value.
- `XDist`, `YDist`, `ZDist` — Input filename of 2D histogram table of X, Y, and Z plane. X and Y are ignored when `TranDist` or `FullDist` is present.
- `TranDist` — Input file name of the 4D histogram table of transverse plane. Has to be used together with `ZDist`.
- `FullDist` — Input file name of the 6D histogram table. If present, all other tables are ignored.

- bunch — The (incomplete) name of an SDDS file to which the phase-space coordinates of the simulated scattered particles are to be written. Recommended value: "%s-%03ld.bun". If "%03ld" or the equivalent is not provided then only the last simulated bunch is kept (one bunch for one TSCATTER element).
- loss — The (incomplete) name of an SDDS file to which the original and final phase-space coordinates of the lost simulated scattered particles are to be written. Recommended value: "%s-%03ld.los". Used together with do_track = 1.
- distribution — The (incomplete) name of an SDDS file to which the one-dimensional histogram of simulated scattered particles are to be written. Recommended value: "%s-%03ld.dis"
- initial — The (incomplete) name of an SDDS file to which the one dimension histogram of simulated particles before scattering are to be written. Recommended value: "%s-%03ld.ini"
- output — The (incomplete) name of an SDDS file. The average loss rate (particles per second) over a step size of sbin_step at location s is written to this file. Recommended value: "%s-%03ld.out"
- sbin_step — Bin size for loss rate summary output to the output file.
- nbins — Number of bins used for the distribution and initial table.
- n_simulated — The total number of simulated scattered particles with $\delta > \delta_m$. Choosing too small a value will cause unreliable results. Note: use an integer number here. A number such as 5E6 sometimes will cause you trouble.
- ignored_portion — Fraction of the total scattering rate ignored in tracking. Using this parameter will greatly increase the tracking speed. For example, if the total loss rate is 50% of the total scattering rate, then ignoring for tracking purposes 5% (0.05) of the scattered particles will cause a ∼10% error, but the simulation is greatly sped up.
- i_start, i_end — The simulation will be done from the i_startth to the i_endth TSCATTER element along the beamline.
- do_track — If non-zero, scattered particles will be tracked from their generation location for n_passes (given by run_control). If non-zero, the run_control command must proceed the |touschek_scatter| command. The loss property can be analysed using output or loss.
- match_position_only — If non-zero, then matching of the momentum aperture data to the lattice is done using the position data only (s column), rather than the element names. Can be helpful if errors appear about files ending prematurely or data not matching.
- overwrite_files — If non-zero, then output files will be overwritten. If set to zero, then when output files are found, the corresponding computations are skipped. This can be used to restart a Touschek scattering run, provided the output filenames are index (e.g., of the form "%s-%03ld.los" rather than "%s.los".)

Note: If using Pelegant to compute the momentum aperture with output_mode=1, it is necessary to first run the script reorganizeMmap to put the data into the form needed by touschekLifetime.

7.68 transmute_elements

- type: setup command.
- function: Changes the type of selected elements, which may be used to turn off unneeded diagnostics and speed up tracking when concatenation is being used.
- Must be preceded by `run_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- notes:
  - Any number of these commands may be given.
  - The only property of the original element that is preserved is the length. For example, transmuting a SBEN into a CSBEN will not have the expected result.

```
&transmute_elements
        STRING name = NULL,
        STRING type = NULL,
        STRING exclude = NULL,
        STRING new_type = "DRIF",
        long disable = 0;
        long clear = 0;
&end
```

- `name` — Possibily wild-card containing string specifying the elements to which the transmutation specification is to be applied.
- `type` — Possibily wild-card containing string specifying the element types to which the transmutation specification is to be applied.
- `exclude` — Possibily wild-card containing string specifying elements to be excluded from the specified transmutation. Does not affect elements transmuted due to other specifications.
- `new_type` — Type into which specified elements will be transmuted.
- `disable` — If nonzero, the command is ignored.
- `clear` — If nonzero, all prior transmutation specifications are deleted.

7.69 tune_footprint

- type: action/setup command.
- function: compute frequency map from tracking and use it to determine the chromatic and amplitude tune footprints.
- sequence: must follow `run_control`.
- can use parallel resources (`Pelegant`)
- N.B.: the number of turns tracked is set by the `run_control` command.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&tune_footprint
    STRING delta_output = NULL,
    STRING xy_output = NULL,
    double xmin =  -0.02,
    double xmax =  0.02,
    double ymin =  1e-6,
    double ymax =  0.02,
    double x_for_delta = 1e-6,
    double y_for_delta = 1e-6,
    double delta_min = 0,
    double delta_max = 0,
    long ndelta = 21,
    long separate_xy_for_delta =  0;
    long nx = 20,
    long ny = 21,
    long verbosity = 1,
    long quadratic_spacing = 1,
    long compute_diffusion = 1;
    long diffusion_rate_limit = -5,
    long immediate = 0
    long filtered_output = 1;
    long ignore_half_integer = 0;
&end
```

- `delta_output` — The optional (incomplete) name of an SDDS file to send tune and diffusion rate vs δ output to. Recommended value: "%s.dtf". If optimization is done, this file is written only at the end of optimization.
- `xy_output` — The optional (incomplete) name of an SDDS file to send tune and diffusion rate vs (x, y) output to. Recommended value: "%s.atf". If optimization is done, this file is written only at the end of optimization.
- `xmin`, `xmax` — Limits of grid of initial x coordinates for tracking.

- ymin, ymax — Limits of grid of initial y coordinates for tracking. ymin should typically be a small, positive value so that there is some betatron oscillation from which to get the tune.
- delta_min, delta_max — Limits of grid of initial δ coordinates for tracking. Not that particles are not centered around the dispersive closed orbit.
- ndelta — Number of values of δ coordinate in the grid. If zero, chromatic footprint is not determined.
- separate_xy_for_delta — If nonzero, tracking for the x and y momentum-dependent tunes will be done separately, so that when x-plane tracking is performed, $y = 0$ initially. This might be helpful if nonlinear coupling of y motion into the x plane causes the x tune to be poorly determined for small x amplitudes. Increase the tracking time by a factor of two.
- nx — Number of values of x coordinate in the grid. If zero, amplitude footprint is not determined.
- ny — Number of values of y coordinate in the grid. If zero, amplitude footprint is not determined.
- verbosity — If nonzero, prints possibly useful information while running.
- quadratic_spacing — If nonzero, points are spaced "quadratically," which actually means that their squares are spaced linearly. It is highly recommended to keep this turned on, since otherwise problems determining the tune when $x \approx 0$ may result in invalid results.
- compute_diffusion — If nonzero, diffusion is computed, which requires tracking twice as many turns.
- diffusion_rate_limit — Value of the diffusion rate d_r above which the particle is considered unstable, where

$$d_r = \log_{10}\left(\frac{\Delta\nu_x^2 + \Delta\nu_y^2}{N}\right), \tag{13}$$

  where N is the number of turns tracked to determine each tune (equal to half of n_passes).
- immediate — If nonzero, the calculations take place immediately. If zero, then two modes are possible
  - If you wish to compute parameters on a closed orbit or after other calculations, be sure to set this control to zero and ask for an output file with xy_output or delta_output.
  - If you want to use this command to create quantities for optimization (see below), be sure to set this control to zero and *do not* ask for an output file with xy_output or delta_output.
- filtered_output — If nonzero, output is only provided for particles inside the stable footprint.
- ignore_half_integer — If nonzero, half-integer resonances are ignored in determining the tune footprint.
- chromaticity_fit_order — Order of polynomial fits used to obtain chromaticities.

This command makes available the following quantities for optimization. All quantities are limited by particle survival, crossing of integer and half-integer resonances, and the diffusion rate limit.

- `FP.nuxSpreadChrom`,`FP.nuySpreadChrom` — Spread in tunes due to chromaticity.
- `FP.nuxChromMin`, `FP.nuxChromMax`, `FP.nuyChromMin`, `FP.nuyChromMax` — Minimum and maximum values of the x and y tunes from chromatic tune footprint.
- `FP.deltaLimit` — Minimum of absolute values of positive and negative δ limits.
- `FP.nuxSpreadAmp`, `FP.nuySpreadAmp` — Spread in tunes due to amplitude.
- `FP.nuxAmpMin`, `FP.nuxAmpMax`, `FP.nuyAmpMin`, `FP.nuyAmpMax` — Minimum and maximum values of the x and y tunes from amplitude tune footprint.
- `FP.xSpread`, `FP.ySpread` — Spread in x and y values.
- `FP.xyArea` — Area of the limited x-y region, comparable to a dynamic acceptance. However, this area is determined from a fixed grid and is not suitabl to optimization by itself.
- `FP.diffusionRateMaxChrom`, `FP.diffusionRateMaxAmp` — Maximum diffusion rates in chromatic and amplitude scans.
- `FP.chromx1`, `FP.chromy1` — Linear chromaticities from fits to data.

Typically, one strives to minimize `FP.nuxSpreadChrom`,`FP.nuySpreadChrom`, `FP.nuxSpreadAmp`, `FP.nuySpreadAmp`, `FP.diffusionRateMaxChrom`, and/or `FP.diffusionRateMaxAmp` while maximizing `FP.deltaLimit`, `FP.xSpread`, and/or `FP.ySpread`, and ensuring that `FP.xyArea`, at minimum, doesn't decrease. I.e., one wants the maximum stable region for momentum and position deviations with the minimum spread in tunes and minimum diffusion.

7.70 twiss_analysis

- type: setup command.
- function: analyze Twiss parameters within a user-defined region for purposes of optimization.
- sequence: must precede `twiss_output`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&twiss_analysis
        STRING match_name = NULL;
        STRING start_name = NULL;
        STRING end_name = NULL;
        double s_start = -1;
        double s_end = -1;
        STRING tag = NULL;
        long verbosity = 0;
        long clear = 0;
&end
```

- `match_name` — Optional wildcard string to match to element names for selection of elements to inculde in the analysis.
- `start_name` — Name of the element at which to start analysis. If the element occurs more than once, the first occurrence is used.
- `end_name` — Name of the element at which to end analysis. If the element occurs more than once, the first occurrence is used.
- `s_start` — Position (in meters) at which to start analysis.
- `s_end` — Position (in meters) at which to end analysis.
- `tag` — Name prefix for quantities computed by the analysis. The quantity names will have the form *tag.statistic.quantity*, where *statistic* is one of `min`, `max`, and `ave`, and *quantity* is one of `betax`, `betay`, `etax`, `etay`, `alphax`, `alphay`, `etaxp`, and `etayp`. E.g., if *tag* is `region1`, then one could use expressions like `region1.max.betax` in optimization.
- `clear` — If nonzero, all previously defined analysis regions are deleted.

7.71 twiss_output

- type: action/setup command.
- function: compute and output uncoupled Twiss parameters, or set up to do so.
- sequence: must follow `run_setup`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- N.B.: the output of this command is strictly correct only when the beamline has vanishingly small x-y coupling. For rings, use of `coupled_twiss_output` is an option when that requirement is not sufficiently well satisfied.

```
&twiss_output
    STRING filename = NULL;
    long matched = 1;
    long output_at_each_step = 0;
    long output_before_tune_correction = 0;
    long final_values_only = 0;
    long statistics = 0;
    long radiation_integrals = 0;
    long concat_order = 3;
    long higher_order_chromaticity = 0;
    long higher_order_chromaticity_points = 5;
    double higher_order_chromaticity_range = 4e-4;
    double chromatic_tune_spread_half_range = 0;
    long quick_higher_order_chromaticity = 0;
    double beta_x = 1;
    double alpha_x = 0;
    double eta_x = 0;
    double etap_x = 0;
    double beta_y = 1;
    double alpha_y = 0;
    double eta_y = 0;
    double etap_y = 0;
    STRING reference_file = NULL;
    STRING reference_element = NULL;
    long reference_element_occurrence = 0;
    long reflect_reference_values = 0;
    long cavities_are_drifts_if_matched = 1;
    long compute_driving_terms = 0;
    long leading_order_driving_terms_only = 0;
    STRING s_dependent_driving_terms_file = NULL;
    long local_dispersion = 1;
&end
```

- `filename` — The (incomplete) name of an SDDS file to which the Twiss parameters will be written. Recommended value: "%s.twi".

- matched — A flag indicating, if set, that the periodic or matched Twiss parameters should be found. If zero, calculations are performed in transport line mode starting from the given initial values of betax, alphax, etc. As a special case, if matched=-1 the solution is for a half periodic cell, with mirror symmetry; this will probably cause problems for higher-order calculations.

  N.B.: This may give different values for the chromaticity even if the initial values are identical to those for a periodic solution. The reason has to do with different assumptions about the initial conditions for particles in a transport line vs a ring.

- output_at_each_step — A flag indicating, if set, that output is desired at each step of the simulation. If you wish to compute Twiss parameters on a closed orbit or after other calculations, be sure to set this control to a nonzero value.

- output_before_tune_correction — A flag indicating, if set, that output is desired both before and after tune correction.

- final_values_only — A flag indicating, if set, that only the final values of the Twiss parameters should be output, and not the parameters as a function of s.

- statistics — A flag indicating, if set, that minimum, maximum, and average values of Twiss parameters should be computed and included in output.

- radiation_integrals — A flag indicating, if set, that radiation integrals should be computed and included in output. *N.B.: Radiation integral computation is not correct for systems with vertical bending, nor does it take into account coupling. See the* moments_output *command if you need such computations.*

- beta_X, alpha_X, eta_X, etap_X — If matched is zero, the initial values for the X plane.

- concat_order — Order of matrix concatenation to use for determining matrix for computation of Twiss parameters. Using a lower order will result in inaccuracy for nonlinear lattices with orbits and/or momentum errors. However, for on-momentum conditions with zero orbit, it is much faster to use concat_order=1.

- higher_order_chromaticity — If nonzero, requests computation of the second- and third-order chromaticity. To obtain reliable values, the user should use concat_order=3 in this namelist and the highest available order for all beamline elements. elegant computes the higher-order chromaticity by finding the trace of off-momentum matrices obtained by concantenation of the matrix for higher_order_chromaticity_points values of δ over the full range higher_order_chromaticity_range. If quick_higher_order_chromaticity is nonzero, then a quicker concatenation method is used that gives the second-order chromaticity only.

- chromatic_tune_spread_half_range — Half range of δ for which the chromatic tune spread is computed. The results are available in for optimization and in the twiss output file under the names nuxChromUpper, nuxChromLower, and similarly for the y plane. This computation uses the chromaticities.

- reference_file — If given, the name of a file from which twiss parameter data will be taken to give the starting values. Ignored if matched is nonzero. The file should have the beta and alpha functions with the same names as the file created by this command.

- `reference_element` — Element in `reference_file` at which to take the twiss parameter values. If not given, the values at the last element in `reference_file` are used.

- `reference_element_occurrence` — Ignored if `reference_element` is not given. Otherwise, the occurence number of `reference_element` to use. If 0, the last occurence is used.

- `reflect_reference_values` — If nonzero, reference values of $\alpha_{x,y}$ and $\eta'_{x,y}$ are multiplied by -1. This permits matching backwards from the reference point.

- `cavities_are_drifts_if_matched` — By default, if `matched=1`, `elegant` treats rf cavities as drift spaces, allowing the user to have a cavity in the ring definition without it affecting the lattice functions. By setting `cavities_are_drifts_if_matched=0`, one can force `elegant` to use the actual matrix for the rf cavity. The differences between the results are generally small, but the default behavior disagrees with the results of `moments_output`. This feature is not available for cavities that change the beam energy (`CHANGE_P0=1` in element definition or `always_change_p0=1` on `run_setup`). Setting this to 0 for a ring is unusual, but allows computing the effect of energy modulation around a ring if combined with the `SR_IN_ORDINARY_MATRIX=1` on `CSBEND`, `KQUAD`, and other elements.

- `compute_driving_terms` — If nonzero, then resonance driving terms [29, 36, 37] and tune shifts with amplitude are computed by summing over dipole, quadrupole, sextupole, and octupole elements. For dipoles, only the effects of gradients and sextupole terms are included; curvature effects are not present in the theory. In addition, these quantities may be optimized by using those names in optimization terms (see list below).

- `leading_order_driving_terms_only` — If nonzero, only the leading order driving terms are computed. I.e., terms involving double sums over sextupole and quadrupole strengths are not computed. However, leading-order octupole terms are computed, even though they affect the same terms as the second-order sextupole and quadrupole terms. This option is provided because computing the higher-order terms is time-consuming and not always worthwhile.

- `s_dependent_driving_terms_file` — The (incomplete) name of a SDDS file to which magnitude, real and imaginary parts of s-dependent driving terms will be written. If you wish to compute s-dependent driving terms, be sure to set this parameter. The following first order resonant driving terms are implemented as defined in [42]: `f10010`, `f10100`, `f30000`, `f12000`, `f10200`, `f01200`, `f01110`, `f00300`, `f00120`, `f20100`, `f20010` and `f11010`. Please note that the notation and meaning of the driving terms differs from those computed when `compute_driving_terms=1`!

- `local_dispersion` — Normally, `elegant` will ignore acceleration in computing the dispersion. That is, the dispersion would be the "local" dispersion $\frac{\partial x}{\partial \delta}$, where δ was the local fractional momentum deviation. In a linear system, the local dispersion is related to the beam moments by $\eta_x = \langle x\delta\rangle / \langle \delta^2 \rangle$. In a linac or other systems with rf elements, one might also be interested in the "global" dispersion $\frac{\partial x}{\partial \delta_0}$, where δ_0 is the energy deviation at the beginning of the system. In this case, set `local_dispersion=0`. Alternatively, one may look at the R_{i6} elements of the matrix from `matrix_output`.

The output file from this command contains the following columns, giving values of quantities at the exit of each element, unless otherwise noted.

- `s` — The arc length.

- `ElementName` — The name of the element.
- `ElementType` — The type name of the element.
- `betax` and `betay` — The horizontal and vertical beta functions.
- `alphax` and `alphay` — The horizontal and vertical alpha functions, where $\alpha = -\frac{d\beta}{2ds}$.
- `psix` and `psiy` — The horizontal and vertical betatron phase advance in radians.
- `etax` and `etay` — The horizontal and vertical dispersion functions.
- `etaxp` and `etayp` — The slopes of the horizontal and vertical dispersion functions.
- `xAperture` and `yAperture` — The horizontal and vertical apertures. If undefined, will have a value of 10m. If the beam trajectory is non-zero, then the aperture will be changed (usually reduced) accordingly. Hence, these are best understood as the `effective` apertures. They are used in determining the horizontal and vertical acceptance parameters, `Ax` and `Ay`.
- `pCentral0` — The central momentum ($\beta\gamma$) at the **entrance** to the element.
- `dI`n — Contribution to radiation integral `I`n. Radiation integrals take account of horizontal bending only.

The output file contains the following parameters. Note that chromatic quantities depend on the order settings of the individual elements, the default order (in `run_setup`), and the concatenation order given in the `twiss_output` command. These quantities pertain to the end of the lattice or to the lattice as a whole.

- `nux` and `nuy` — The horizontal and vertical tunes.
- `dnux/dp` and `dnuy/dp` — The horizontal and vertical chromaticities, defined as $d\nu/d\delta$.
- `dnux/dp2` and `dnuy/dp2` — The horizontal and vertical 2nd-order chromaticities, defined as $d^2\nu/d\delta^2$. Will be zero if `higher_order_chromaticity` is zero.
- `dnux/dp3` and `dnuy/dp3` — The horizontal and vertical 3rd-order chromaticities, defined as $d^3\nu/d\delta^3$. Will be zero if `higher_order_chromaticity` is zero.
- `dbetax/dp` and `dbetay/dp` — Chromatic derivatives of the horizontal and vertical beta functions, defined as $\frac{d\beta}{d\delta}$.
- `dalphax/dp` and `dalphay/dp` — Chromatic derivatives of the horizontal and vertical alpha functions, defined as $\frac{d\alpha}{d\delta}$.
- `etax2`, `etax3`, `etay2`, `etay3` — Higher order dispersion in the horizontal and vertical planes. For example, for the horizontal plane, the closed orbit at the end of the lattice depends on δ according to $x = \eta_x\delta + \eta_{x2}\delta^2 + \eta_{x3}\delta^3$. This differs from the chromaticity expansion, which is given in terms of successive derivatives of $\nu(\delta)$.
- `dnux/dAx`, `dnux/dAy`, `dnuy/dAx`, `dnuy/dAy` — Tune shifts with amplitude, where amplitude is defined as $A_q = (1 + \alpha_q)q^2/\beta_q$, with $q = x$ or $q = y$. These will be zero unless the `tune_shift_with_amplitude` command is given.

- h11001, h00111, h20001, h00201, h10002, h21000, h30000, h10110, h10020, h10200, h22000, h11110, h00220, h31000, h40000, h20110, h11200, h20020, h20200, h00310, h00400— Resonance driving terms[29]. These will be zero unless compute_driving_terms is nonzero. See table 2 for an explanation of each term.
- dnux/dJx, dnux/dJy, and dnuy/dJy — Tune shifts with amplitude from Bengtsson's theory [29]. Note that $J_q = \frac{A_q}{2}$, where q is x or y. See documentation for tune_shift_with_amplitude for discussion and comparison with dnux/dAx etc. These will be zero unless compute_driving_terms is nonzero.
- Ax and Ay — The horizontal and vertical acceptance. These will be zero if no apertures are defined.
- alphac, alphac2, alphac3 — First-, second, and third-order momentum compaction. The path length is $s = s_o + \alpha_c L\delta + \alpha_{c2} L\delta^2 + \alpha_{c3} L\delta^2$. Regarding α_{c3}, users are cautioned that the analytical matrices for most elements are limited to second-order, so using tracking-derivce matrices is necessary where supported, and gives limited accuracy.
- couplingIntegral, couplingDelta, and emittanceRatio — These quantities are defined in section 3.1.4.4 of [19]. The computations include tilted quadrupoles, vertical orbit in sextupoles, vertical sextupole displacement, and solenoids. Note that the emittance ratio *does not* include the effect of vertical dispersion.
- In — The n^{th} radiation integral.
- taux, tauy, taudelta — Radiation damping times for x, y, and δ.
- Jx, Jy, Jdelta — Damping partition factors for x, y, and δ.
- ex0, enx0 — Horizontal equilibrium geometric and normalized emittances.
- Sdelta0 — Equilibrium fractional rms energy spread.
- U0 — Energy loss per turn.

N.B.: the higher-order dispersion and higher-order chromaticity are computed using the concatenated third-order matrix. However, elegant only has third-order matrices for three elements: alpha magnets, quadrupoles, and sextupoles. This may be acceptable if any dipoles (for example) have large bending radius. Users who are concerned about these effects should perform off-energy tracking using canonical elements (i.e., CSBEND, KQUAD, KSEXT, and MULT), which include energy dependence to all orders.

Also, note that by default all elements are computed to second order only. You must change the default_order parameter on run_setup to 3 in order to use the third-order matrices for alpha magnets, quadrupoles, and sextupoles. You may also use the ORDER parameter on individual element definitions.

Table 2: Meaning of the various driving terms[29].

Term Name	Explanation
h11001	drives x chromaticity
h00111	drives y chromaticity
h20001	drives synchro-betatron resonances
h00201	drives momentum-dependence of beta functions
h10002	drives second order dispersion
h21000	drives ν_x
h30000	drives $3\nu_x$
h10110	drives ν_x
h10020	drives $\nu_x - 2\nu_y$
h10200	drives $\nu_x + 2\nu_y$
h22000	drives $d\nu_x/dJ_x$
h11110	drives $d\nu_x/dJ_y$
h00220	drives $d\nu_y/dJ_y$
h31000	drives $2\nu_x$
h40000	drives $4\nu_x$
h20110	drives $2\nu_x$
h11200	drives $2\nu_y$
h20020	drives $2\nu_x - 2\nu_y$
h20200	drives $2\nu_x + 2\nu_y$
h00310	drives $2\nu_y$
h00400	drives $4\nu_y$

`track`

7.72 track

- type: major action command.
- function: track particles.
- sequence: must follow `run_setup`, `run_control`, and beam definition with `bunched_beam` or `sdds_beam`.
- can use parallel resources (`Pelegant`)
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&track
    long center_on_orbit = 0;
    long center_momentum_also = 1;
    long offset_by_orbit = 0;
    long offset_momentum_also = 1;
    long soft_failure = 1;
    long stop_tracking_particle_limit = -1;
```

```
    long check_beam_structure = 0;
    STRING interrupt_file = "%s.interrupt";
&end
```

- `center_on_orbit` — A flag indicating whether to center the beam transverse coordinates on the closed orbit before tracking.
- `center_momentum_also` — A flag indicating whether to center the momentum coordinate also.
- `offset_by_orbit` — A flag indicating whether to offset the transverse beam coordinates by the closed orbit before tracking. Similar to `center_on_orbit`, but the initial centroids of the beam are preserved. The beam is simply displaced by the closed orbit rather than being centered on it.
- `offset_momentum_also` — A flag indicating whether to also offset the beam momentum to the momentum of the closed orbit. If the `start_from_centroid` or `start_from_dp_centroid` parameters are used on the `closed_orbit` command, this flag should be set to 0; otherwise, one will offset the beam central momentum by its own value.
- `soft_failure` — If there is an error during tracking (e.g., a failure of orbit correction), continue to produce file output. This creates essentially empty slots in the files corresponding to the failed steps.
- `stop_tracking_particle_limit` — If a non-negative is given, then `elegant` will stop tracking when the number of particles falls below the given value. It will be as if all the particles were lost.
- `check_beam_structure` — For debugging use only.
- `interrupt_file` — Gives the (possibly incomplete) name of a file to monitor as a semaphore to interrupt the tracking. If the file is created or updated during tracking, then tracking will terminate on completion of the next pass. Output already written to `WATCH` files is preserved, but unwritten data (e.g., buffered, but not written to disk) is lost.

There are also several deprecated parameters:

- `use_linear_chromatic_matrix` — For each particle, a first-order matrix is computed for the particular momentum offset of the particle using the linear chromaticity and linear dependence of the beta functions on momentum. Use `ILMATRIX` elements instead.
- `longitudinal_ring_only` — Tracks longitudinal coordinates only for a ring. Use `ILMATRIX` elements instead.

7.73 tune_shift_with_amplitude

- N.B.: this command is deprecated, because it is too difficult to tune it to get reliable answers. The use of driving term computation in `twiss_output` is recommended instead, even though it doesn't include all possibly relevant effects. For tune-spread calculations, the `tune_footprint` command provides more versatility.
- type: setup command.
- function: prepare for computation of tune shifts with amplitude.
- sequence: must follow `twiss_output`.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.
- methods:

Method 1 : tune shifts with amplitude are computed via tracking a series of particles at different amplitudes or by a matrix method. NAFF is used to determine the tunes from the tracking data. It is the user's responsbility to optimize the parameters to ensure that results are reasonable.

Method 2 : tune shifts are computed using a concatenated multi-turn third-order matrix. This appears to be reliable for many cases we've tested.

Method 3 : tune shifts can be computed quickly using Bengtsson's formulae [29] by setting `compute_driving_terms=1` in `twiss_output`. For cases where all methods are valid, the results will be larger by a factor of 2 than the results obtained with this command, since $J_q = \frac{A_q}{2}$, where q is x or y. Note that the present command has more general validity because it includes dipole curvature effects.

The quantities computed are $\frac{\partial}{\partial A_x^n \partial A_y^m}\nu_p$, where $n \geq 0$ and $m \geq 0$ are integers and p is x or y. $A_q = (1+\alpha_q)q^2/\beta_q$, with $q = x$ or $q = y$.

```
&tune_shift_with_amplitude
    long turns = 2048;
    double x0 = 1e-6;
    double y0 = 1e-6;
    double x1 = 3e-4;
    double y1 = 3e-4;
    long grid_size = 6;
    long lines_only = 0;
    long spread_only = 0;
    double nux_roi_width = 0.02;
    double nuy_roi_width = 0.02;
    double scale_down_factor = 2;
    double scale_up_factor = 1.05;
    double scale_down_limit = 0.01;
    double scale_up_limit = 1e-4;
    long scaling_iterations = 10;
```

```
    long use_concatenation = 0;
    long verbose = 0;
    long order = 2;
    STRING tune_output = NULL;
&end
```

- `turns` — The number of turns to track. If zero, then the concatenated matrix is used instead of tracking, and all other parameters of this command are irrelevant. The matrix method doesn't work well with all lattices. The order of the concatenated matrix is given by the `concat_order` control in `twiss_output`.

- `x0`, `y0` — The initial x and y amplitudes to use for determining the small-amplitude tunes.

- `x1`, `y1` — The initial x and y amplitudes to user for determining the tune shifts. These values should be small enough to ensure linearity in the tune shift.

- `grid_size` — Size of the grid of points in x and y.

- `lines_only` — If nonzero, then instead of a full set of `grid_size`^{2} particles, only two lines of particles with $x = 0$ and/or $y = 0$ are tracked. In this case, no $A_x^i * A_y^j$ terms are computed (except for $i = 0$ or $j = 0$). However, in addition to being faster, the results may be more reliable, e.g., $\partial\nu_x/\partial A_y = \partial\nu_y/\partial A_x$ may be more closely satisfied.

- `sparse_grid` — Deprecated. If nonzero, then instead of a full set of `grid_size`^{2} particles, a sparse grid of particles is tracked. Will save time at the expense of inaccurate higher-order terms. Not recommended.

- `spread_only` — Compute the tune spread only and don't bother with the tune shift coefficients. These tune spreads can be optimized and appear in the twiss output file under the names `nuxTswaLower`, `nuxTswaUpper`, and similarly for the y plane. This is the recommended way to reduce tune shift with amplitude, as the tune spread is more reliable than the coefficients of the expansion. (Particles that get lost are automatically ignored in both types of computations.)

- `nux_roi_width`, `nuy_roi_width` — Widths of the region of interest for x and y tunes. As the grid is filled in, `elegant` finds the tune for each tracked particle on the grid. Successive tune values are looked for in the region of the given width around the previous tune value. This prevents jumping from the main tune peak to another peak, which can happen when the tune spectrum has many lines.

- `scale_down_factor`, `scale_up_factor`, `scale_down_limit`, `scale_up_limit`, `scaling_iterations` — These control automatic scaling of the amplitudes. If `elegant` sees a tune shift larger than `scale_down_limit` it will decrease `x0` (or `y0`) by the factor `scale_down_factor`. If `elegant` sees a tune shift smaller than `scale_up_limit` it will increase `x0` (or `y0`) by the factor `scale_up_factor`. Suggestion: if you find yourself playing with these values and the initial amplitudes in order to get reliable TSWA coefficients, try just using the tune spread.

- `verbose` — If nonzero, information about the progress of the algorithm is printed to the screen.

- use_concatenation — If nonzero, then tracks with the concatenated matrix instead of element-by-element. The order of the concatenated matrix is given by the concat_order control in twiss_output. The user should experiment with this option to see if the results are reliable for a particular lattice.

7.74 vary_element

- type: setup command.
- function: define an index and/or tie a parameter of an element to it.
- sequence: must follow `run_control`
- N.B.: It is not possible to vary an element if the element name starts with one of the following characters: `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `.`, `+`, or `-`. The reason is that `elegant` will attempt to make an SDDS parameter name containing the element name, and these characters are disallowed at the beginning of such a name.
- Command syntax, including use of equations and subcommands, is discussed in 7.2.

```
&vary_element
    long index_number = 0;
    long index_limit = 0;
    STRING name = NULL;
    STRING item = NULL;
    double initial = 0;
    double final = 0;
    long differential = 0;
    long multiplicative = 0;
    long geometric = 0;
    STRING enumeration_file = NULL;
    STRING enumeration_column = NULL;
    long disable = 0;
&end
```

- `index_number` — A non-negative integer giving the number of the index.
- `index_limit` — A positive integer giving the number of values the index will take. Must be given if this `index_number` has not been listed in a previous `vary_element` command, unless `enumeration_file` is given.
- `name` — The name of an element.
- `item` — The parameter of the element to vary.
- `initial`, `final` — The initial and final values of the parameter.
- `enumeration_file` — Name of an SDDS file giving values for the item.
- `enumeration_column` — Column of the SDDS file giving the values.
- `differential` — If nonzero, the initial and final values are taken as offsets from the predefined value of the parameter.
- `multiplicative` — If nonzero, the initial and final values are taken as multipliers to be applied to the predefined value of the parameter in order to obtain the actual initial and final values.

- `geometric` — If nonzero, then variation is geometric rather than arithmetic.
- `disable` — If nonzero, the command is ignored.

8 Specialized Tools for Use with `elegant`

A number of specialized programs are available that work with `elegant`. Most are SDDS-compliant, so they will also work with any program that reads or writes appropriate SDDS data. The following is a brief description of each program. Full descriptions for many programs are available on subsequent pages. Most programs will return a help message if the program name is given with no arguments, which should be sufficient documentation and may be more up-to-date than these manual pages.

- `abrat` — A program to integrate particles through a 3D magnetic field map. The name stands for Asymmetric Bend RAy trace. This program uses the same method as the `BRAT` element in `elegant`.
- `analyzeMagnets` — Generates SDDS and latex files giving magnet parameters. (Program by M. Borland.)
- `astra2elegant` — Converts ASCII particle output from ASTRA [30] to a binary SDDS file suitable for use with `elegant`. This program is recommended over the astra2sdds program on the ASTRA website, because the latter produces ASCII SDDS files that are quite slow to read and does not perform the correct computations for low-energy beams. (Program by M. Borland.)
- `bremsstrahlungLifetime` — Computes gas bremsstrahlung lifetime from local momentum acceptance and Twiss parameter output, assuming a constant gas pressure. (Program by M. Borland.)
- `bremsstrahlungLifetimeDetailed` — Computes gas bremsstrahlung lifetime from local momentum acceptance and Twiss parameter output, using a user-supplied, s-dependent gas pressure. (Program by M. Borland.)
- `computeCoherentFraction` — Computes the coherent fraction for undulator radiation.
- `computeCBGGE` — Computes generalized gradients from data on a circular cylinder for use with the `BGGEXP` element.
- `computeGeneralizedGradients` — Deprecated. Use `computeCBGGE`.
- `computeRBGGE` — Computes generalized gradients from data on a rectangular boundary for use with the `BGGEXP` element.
- `computeSCTuneSpread` — Compute space charge tune spread.
- `coreEmittance` — Computes the slice emittance for the beam core (e.g., 80% of the beam). (Program by X. Dong.)
- `csrImpedance` — Computes the shielded steady-state CSR impedance for a dipole magnet. The output can be used immediately with `elegant`'s `ZLONGIT` element. (Program by Y. Wang, H. Shang, and M. Borland.) See also the `makeSummedCsrZ` script.
- `doubleDist6` — Increases the number of particles in a particle input file by successively doubling the number. Intended to be used to increase the number of particles produced by a photoinjector simulation to improve stability of CSR and LSC simulations. See also `smoothDist6`. (Program by M. Borland.)

- `curvedDipoleFringeCalc` — Computes fringe integrals needed by the `CSBEND` fringe model method 5.

- `elasticScatteringAnalysis` — Computes elastic gas scattering lifetime and loss distribution from multi-location tracking data, Twiss parameter output, and gas pressure distribution. Use with output of the `elastic_scattering` command in `Pelegant`. (Program by M. Borland.)

- `elasticScatteringLifetime` — Computes elastic gas scattering lifetime from single-location dynamic acceptance and Twiss parameter output, assuming a constant gas pressure. (Program by M. Borland.)

- `elasticScatteringLifetimeDetailed` — Computes elastic gas scattering lifetime from single-location dynamic acceptance and Twiss parameter output, using a user-supplied, s-dependent gas pressure. (Program by M. Borland.)

- `elegant2astra` — This program translates `elegant` phase space files into ASTRA [30] format. (Program by M. Borland.)

- `elegant2track` — This program translates `elegant` phase space files into TRACK [32] format. The ASCII version of TRACK is assumed. (Program by M. Borland.)

- `elegant2genesis` — This program performs slice analysis of particle output files, which are suitable for use with the SDDS-compliant APS version of GENESIS[14]. This program is part of the SDDS toolkit. See the SDDS toolkit manual for documentation. (Program by R. Soliday and M. Borland.)

- `elegantto` — Translates an `elegant`-style lattice file (or a MAD file, with some restrictions) into formats accepted by other programs, such as COSY, PARMELA, PATPET, PATRICIA, TRANSPORT, XORBIT, and MAD8. Will also generate an SDDS file containing lattice data. (Program by M. Borland.)

- `generateBunch` — Generates a gaussian-distributed bunch.

- `generateBunchTrain` — Generates a very flexible multi-train bunch file.

- `haissinski` — Computes the steady-state longitudinal distribution in an electron storage ring. Requires as input a file containing the Twiss parameters around the ring, such as that provided by the `twiss_output` command. Wakes can be specified with either a L, R model, a BBR resonator model or a wake function. Other inputs are external rf system parameters, with possibility of a harmonic cavity. Output is a charge or current profile with longitudinal time coordinate (front of bunch is at positive times). (Program by L. Emery and M. Borland.)

- `ibsEmittance` — Computes local intra-beam scattering rates for both storage ring and linac. Also computes the equlibrium transverse and longitudinal emittances of a beam in an electron storage ring, resulting from the combination of quantum excitation, damping, and intra-beam scattering. Requires as input a file containing the Twiss parameters, such as that provided by the `twiss_output` command. (Program by L. Emery, M. Borland, and A. Xiao)

- `impact2elegant` — Tranlates IMPACT-T [31] output into `elegant` conventions. (Program by M. Borland.)

- `impact2sdds` — Translates IMPACT-T output files into SDDS for easier postprocessing. (Program by M. Borland.)
- `ionTrapping` — Uses lattice function data from `elegant` to compute ion trapping condition in a ring. (Program by M. Borland.)
- `LFBFirSetup` — This script prepares data that can be used to configure turn-by-turn longitudinal feedback using `TFBDRIVER` and `TFBPICKUP` elements. (Program by M. Borland.)
- `longitCalcs` — Performs calculations of longitudinal dynamics parameters in storage rings, using output from `elegant`'s `twiss_output` command. Can also compute voltages for bunch lengthening and output these to a file that can be use with `load_parameters`. (Program by M. Borland.)
- `makeSummedCsrZ` — Computes the shielded or free-space steady-state CSR impedance for a ring composed of one or more types of dipole magnet. The output can be used immediately with `elegant`'s `ZLONGIT` element. (Program by M. Borland.)
- `plotTwiss` – Plots the twiss parameters using data from the `twiss_output` command. (Program by L. Emery and M. Borland.)
- `plotTwissBeamsize` – Plots the beam sizes using data from the `twiss_output` command.
- `prepareTAPAs` — Allows processing files from `twiss_output` into a form that is accepted by the Android App TAPAs [46]. The resultant files can be copied to, e.g., the downloads area on the Android device, from which they can be read by TAPAs for configuration of the Storage Ring Scaling activity. (Program by M. Borland.)
- `radiationEnvelope` — A tool for use with the output of `sddsbrightness` and `sddsfluxcurve`. It analyzes data for many harmonics and produces a single curve that shows the envelope of maximum brightness or flux over all harmonics. (Program by M. Borland.)
- `removeBackDrifts` — Allows post-processing s-dependent files to remove negative drifts, which improves the appearance of plots and is needed for certain types of analysis. (Program by M. Borland.)
- `sddsanalyzebeam` — Analyzes a beam of macro-particles and produces an SDDS file containing beam moments, emittances, equivalent beta functions, etc. The beam file is of the type written by `elegant` using the `output` field of the `run_setup` command, or the WATCH element. (Program by M. Borland.)
- `sddsbrightness` — Uses twiss parameter output or data from `sddsanalyzebeam` to compute undulator brightness curves. (Program by H. Shang, R. Dejus, M. Borland, X. Jiao.)
- `sddsbs` — Computes bending magnet spectra. (Program by H. Shang and M. Borland.)
- `sddsbunchingfactor` — Computes bunching factor vs frequency from phase space data. (Program by M. Borland.)
- `sddsemitproc` — Analyzes quadrupole scan emittance measurement data. Accepts a file containing the transport matrix for each point and measured beam sizes. The file may, for example, be the file produced by the `final` field of the `run_setup` command. The quadrupole scan can be executed inside of `elegant` using `vary_elements`. (Program by M. Borland.)

- `sdds4x4sigmaproc` — Analyzes quadrupole scan beam moment measurement data to determine the initial 4x4 sigma matrix of the beam. Accepts a file containing the transport matrix for each point and measured beam sizes. The file may, for example, be the file produced by the `final` field of the `run_setup` command. The quadrupole scan can be executed inside of `elegant` using `vary_elements`. (Program by M. Borland.)

- `sdds5x5sigmaproc` — Analyzes quadrupole scan beam moment measurement data to determine the initial 5x5 sigma matrix of the beam. Accepts a file containing the transport matrix for each point and measured beam sizes. The file may, for example, be the file produced by the `final` field of the `run_setup` command. The quadrupole scan can be executed inside of `elegant` using `vary_elements`. To work, requires a horizontal bending magnet in the beamline and variation quadrupoles before and after the bending magnet. (Program by M. Borland.)

- `sddsfindresonances` — Uses output from frequency map analysis to find and identify resonance lines. (Program by H. Shang, M. Borland.)

- `sddsfluxcurve` — Uses twiss parameter output or data from `sddsanalyzebeam` to compute undulator flux tuning curves. (Program by M. Borland, H. Shang, R. Dejus.)

- `sddsmatchmoments` — Transforms a beam of macro-particles to match a given set of 6x6 beam moments, where the moments are stored in an output file from `moments_output`.

- `sddsmatchtwiss` — Transforms a beam of macro-particles to match to given beta functions and dispersion. The beam file is of the type written by `elegant` using the `output` field of the `run_setup` command, or the WATCH element. (Program by M. Borland.)

- `sddsws` — Computes wiggler spectra, using code from WS (by R. Dejus). (Program by H. Shang.)

- `sddsurgent` — Uses algorithms from the programs US (by R. Dejus) and URGENT (by R. Walker) for computation of undulator radiation properties, including power density and intensity distributions. (Program by H. Shang, R. Dejus, M. Borland, X. Jiao.)

- `sddsrandmult` — Simulates the effect of random mechanical errors in a quadrupole or sextupole, generating multipole error data that can be used with `elegant`'s `KQUAD` and `KSEXT` elements. (Program by M. Borland.)

- `sddssampledist` — This program allows creating particle distributions from user-designed distribution functions. It is thus a more flexible alternative to `bunched_beam`. This program is part of the SDDS toolkit. See the SDDS toolkit manual for documentation. (Program by M. Borland and H. Shang.)

- `smoothDist6s` — Increases the number of particles in an input particle distribution. At the same time, smooths the distribution and adds optional energy and density modulation. Intended to be used to increase the number of particles produced by a photoinjector simulation to improve stability of CSR and LSC simulations. Also useful in studying the growth rate for energy and density modulations. See also `doubleDist6`. (Program by M. Borland.)

- The script `spiffe2elegant` allows converting the output of the PIC code `spiffe` to the same form as output by `elegant`. Note that `elegant` will read `spiffe` output directly. This script just allows converting the data for use with related programs, such as `sddsanalyzebeam`. (Program by M. Borland.)

- `straightDipoleFringeCalc` — Computes fringe integrals needed by the `CCBEND` fringe model. Computes fringe integrals and other parameters needed by the `LGBEND` segmented dipole model.

- `TFBFirSetup` — This script prepares data that can be used to configure turn-by-turn transverse feedback using `TFBDRIVER` and `TFBPICKUP` elements. (Program by M. Borland.)

- `touschekLifetime` — This program calculates Touschek lifetime using A. Piwinski's formula. Input files are generated from "twiss_output" and "momentum_aperture". (Program by A. Xiao and M. Borland.)

- `track2sdds` — Translates output files, including phase space files, from version 39 of TRACK (with ASCII output [32]) into SDDS. (Program by M. Borland.)

- `track2mag` — Uses TRACK output files to create a file similar to the magnets outupt file from `elegant`. This gives a profile of the beamline that can be plotted with other data. (Program by M. Borland.)

- `trwake2impedance` — Translates a transverse wake (e.g., used for `TRWAKE`) into an impedance usable with `ZTRANSVERSE`. (Script by M. Borland.)

- `view3dGeometry` — Uses freewrl viewer to display 3D geometry of a lattice. (Program by A. Petrenko and M. Borland.)

- `wake2impedance` — Translates a longitudinal wake (e.g., used for `WAKE`) into an impedance usable with `ZLONGIT`. (Script by M. Borland.)

- The scripts `makeSkewResponseCP` and `correctCoupling` can be used to compute the cross-plane response matrices for skew quadrupoles and to perform coupling correction using those matrices. (Program by M. Borland.)

8.1 abrat

- **description:** Integrates particle trajectories through an symmetric or asymmetric bending magnet. The name stands for "Asymmetric Bend RAy Tracing." Features include the ability to optimize the magnet strength and position to ensure, if possible, that the magnet joins two user-defined trajectories. The results of these optimizations can be used in `elegant` with the `BRAT` element.

- **synopsis:**

```
abrat field-file [-3dFieldFile]
[-interpolateField=parameterName,[,order=n][,extrapolate][,permissive]]
[-scan=x | xp | y | yp | delta,lower,upper,number | -beamFiles=input,output
] -vertex=x-in-meters,z-in-meters -nominalEntrance=x,y -nominalExit=x,y
-theta=targetInDegrees -rigidity=Tesla-meters [-output=filename]
[-fsc=value] [-dxDipole=m] [-dzDipole=m] [-yawDipole=value]
[-optimize[=verbose][fse,dx,dz,yaw]] -fseLimit=min,max -dxLimit=min,max
-dzLimit=min,max -yawLimit=min,max
[-fieldmapOutput=filename,zmin,zmax,nz,xmin,xmax,nx]
[-tolerance=integration-tolerance] [-quiet]
```

- **files:**
  - *field-file* — Field map file. Normally, needs to contain columns `x`, `z`, and `B`, giving the field in the midplane. In 3D mode, when the `-3dFieldFile` option is given, then the file should contain `x`, `y`, `z`, `Bx`, `By`, and `Bz`. In all cases, the beam is assumed to move from left ($z < 0$) to right ($z > 0$) with the field bending counter clockwise. Positive `x` is away from the center of curvature.

- **switches:**
  - `-3dFieldFile` — If given, then *field-file* is expected to contain a 3D field map. See above for details.
  - `-interpolate` — If given, then *field-file* is expected to contain at least two 2D field maps on separate pages of the file. These field maps could be, for example, from measurements with different excitation currents, with the excitation current for each case being stored in a named parameter; the pages must be arranged so that the parameter values increase monotonically. ábrat will then automatically interpolate among the field maps to determine the required excitation current (for example); this overrides the `fse` parameter of the `-optimize` option. By default, linear interpolation is used (`order=1`). By default, the search will not go outside the range of the parameter values in the data; if `extrapolate` is given, however, extrapolation outside this range is performed. By default, the grid parameters of the several pages must match exactly; if `permissive` is given, however, this requirement is not enforced.
  - `-scan` — If given, then the value of the named accelerator coordinate is scanned to create a bundle of incoming rays. Output is provided for each ray.

- `-beamFiles` — If given, then an `elegant`-style beam is read and the particles therein are tracked through the dipole. A simular file is created for the output coordinates. Coordinates are defined at the nominal entrance and exit planes. Back-drifts are used to ensure that integration begins and ends outside the magnetic field region (i.e., all of the defined field is included).
- `-output` — If given, particle trajectories are written to the named file.
- `-fsc` — If given, the fractional strength change to apply to the field. Typically taken from a previous optimization run.
- `-dxDipole` — If given, the x positional change to apply to the field. A positive value moves the field away from the center of curvature. Typically taken from a previous optimization run.
- `-dzDipole` — If given, the z positional change to apply to the field. A positive value moves the field further from the incoming beam. Typically taken from a previous optimization run.
- `-yawDipole` — If given, the yaw to apply to the field. A positive value rotates the magnet in the direction of bending. Typically taken from a previous optimization run.
- `-optimize`, `-fseLimit`, `dxLimit`, `dzLimit`, `yawLimit` — Invokes optimization of the various strength and alignment parameters and specifies the allow range of variation.
- `-fieldMapOutput` — Requests output of a field map, allowing confirmation of the input data.
- `-tolerance` — Integration tolerance.
- N.B.: The usage message describes additional switches that have had limited testing. Use with caution.

• **authors:** M. Borland (ANL/APS).

8.2 astra2elegant

- **description:** Converts ASCII particle output from ASTRA to a binary SDDS file suitable for use with `elegant`. This program is recommended over the astra2sdds program on the ASTRA website, because the latter produces ASCII SDDS files that are quite slow to read.
- **synopsis:**

```
astra2elegant [inputFile] [outputFile] [-centerReference]
[-pipe=[input][,output]]
```

- **files:**
  - `inputFile` — ASCII particle output file from ASTRA.
  - `outputFile` — SDDS file containing phase space data. May be used directly with `elegant`.
- **switches:**
  - `-centerReference` — Normally, `astra2elegant` offsets the arrival time of all particles by the arrival time of the reference particle. This behavior can be suppressed by giving the `-centerReference` option. In that case, the arrival time of the reference particle is defined as 0.
  - `-pipe[=input][,output]` — Standard SDDS toolkit pipe option.
- **authors:** M. Borland (ANL/APS).

8.3 computeCBGGE

- **description:** Compute generalized gradients [50] from data on a circular-cylinder boundary for use with `elegant`'s `BGGEXP` element.
- **synopsis:**

```
computeCBGGE
-input=<filename>[,z=<colName>][,phi=<colName>][,Brho=<colName>][,Bz=<columnName>][,rho=<paramName>]
-normal=<output> [-skew=<output>] [-derivatives=<integer>]
[-multipoles=<integer>] [-fundamental=<integer>]
[-evaluate=<filename>[,nrho=<integer>][,nphi=<integer>]
[-autotune=[,significance=<fieldValue>][,minimize=rms|mav|maximum]
[,increaseOnly][,verbose][,log=<filename>]]
```

- **switches:**
  - `input` — Specify name of the input file, which by default contains three columns giving `z`, `phi`, and `Brho`, which specify B_ρ as a function of longitudinal coordinate z and azimuthal angle ϕ. The `z`, `phi`, and `Brho` options may be used to give different names for these columns. The file by default also contains a parameter `rho` giving the radius of the cylinder. The `rho` option may be used to give a different name for the parameter. The data must form a uniform grid in z and ϕ. The N_ϕ values of ϕ should range from 0 to $\Delta\phi(N_\phi - 1)$ where $\Delta\phi = 2\pi/N_\phi$ If the `Bz` qualifier is provided, the named data is used for computation of the solenoidal fields.
  - `normal` — Output file for normal-component generalized gradients. Supply to `NORMAL_FILENAME` parameter of `BGGEXP`.
  - `skew` — Output file for skew-component generalized gradients. Supply to `SKEW_FILENAME` parameter of `BGGEXP`. NB: if B_z is non-zero on axis, this option is essential in getting valid results.
  - `derivatives` — Number of derivatives vs z desired in output. Default: 7
  - `multipoles` — Number of multipoles desired in output. Default: 8
  - `fundamental` — Fundamental multipole of sequence. 0=none (default), 1=dipole, 2=quadrupole, etc.
  - `evaluate` — Asks to evaluate the GGE and place the results in a file. By default, this is done for the cylinder radius and with the same spacing of ϕ values. This can be changed with the `nrho` and `nphi` parameters.
  - `autotune` — Optimizes the number of derivaties and multipoles up the to values given with the `-derivatives` and `-multipoles` options, in order to minimize the deviation of the GGE-derived fields from the values given in the 3D field map file. The region of evaluation is automatically limited by the bounding planes, even if the 3D field map has a larger extent. By default, minimizes the maximum deviation, but user can ask to minimize the rms or mean-absolute-value deviation. Differences below the `significance` value are ignored. The user may request verbose output to see results printed to the terminal, and also a log file for a detailed record. The `increaseOnly` qualifier specifies

that the scan over multipoles m and derivatives d is restricted to never be less than the previous optimal values; for example, if the best value so far was obtained with $m = 4$ and $d = 3$, the remainder of the scan would be restricted to $m \geq 4$ and $d \geq 3$; this can save considerable run time.

- **authors:** M. Borland, R. Soliday, R. Lindberg, (ANL/APS).

8.4 computeRBGGE

- **description:** Compute generalized gradients [50] from data on a rectangular boundary [57] for use with `elegant`'s `BGGEXP` element.
- **synopsis:**

```
computeRBGGE -yminus=<filename> -yplus=<filename> -xminus=<filename>
-xplus=<filename> -normal=<output> [-skew=<output>] [-derivatives=<number>]
[-multipoles=<number>] [-fundamental=<number>] [-evaluate=<filename>]
[-autotune=<3dMapFile>[,significance=<fieldValue>][,minimize=rms|mav|maximum]
[,radiusLimit=<meters>][,increaseOnly][,verbose][,log=<filename>]
[,minDerivatives=<number>],[,minMultipoles=<number>]]
```

- **switches:**
  - `yplus` — SDDS file containing `x`, `y`, `z`, and `By`, map stored in columns for positive-y plane. If skew components are desired, file must also supply `Bz`. Units are meter and Tesla.
  - `yminus` — SDDS file containing `x`, `y`, `z`, and `By`, map stored in columns for negative-y plane. If skew components are desired, file must also supply `Bz`.
  - `xplus` — SDDS file containing `x`, `y`, `z`, and `Bx`, map stored in columns for positive-x plane. If skew components are desired, file must also supply `Bz`.
  - `xminus` — SDDS file containing `x`, `y`, `z`, and `Bx`, map stored in columns for negative-x plane. If skew components are desired, file must also supply `Bz`.
  - `normal` — Output file for normal-component generalized gradients. Supply to `NORMAL_FILENAME` parameter of `BGGEXP`.
  - `skew` — Output file for skew-component generalized gradients. Supply to `SKEW_FILENAME` parameter of `BGGEXP`. NB: if B_z is non-zero on axis, this option is essential in getting valid results.
  - `derivatives` — Number of derivatives vs z desired in output. Default: 7
  - `multipoles` — Number of multipoles desired in output. Default: 8
  - `fundamental` — Fundamental multipole of sequence. 0=none (default), 1=dipole, 2=quadrupole, etc.
  - `evaluate` — Asks to evaluate the GGE and place the results in a file. The GGE is evaluted over the region bounded by the four planes, using the same coordinate intervals.
  - `autotune` — Optimizes the number of derivaties and multipoles up the to values given with the `-derivatives` and `-multipoles` options, in order to minimize the deviation of the GGE-derived fields from the values given in the 3D field map file. The region of evaluation is automatically limited by the bounding planes, even if the 3D field map has a larger extent. By default, minimizes the maximum deviation, but user can ask to minimize the rms or mean-absolute-value deviation. Differences below the `significance` value are ignored. The user may request verbose output to see results printed to the terminal, and also a log file for a detailed record. The `increaseOnly` qualifier specifies

that the scan over multipoles m and derivatives d is restricted to never be less than the previous optimal values; for example, if the best value so far was obtained with $m = 4$ and $d = 3$, the remainder of the scan would be restricted to $m \geq 4$ and $d \geq 3$; this can save considerable run time. The `minDerivatives` and `minMultipoles` parameters allow requiring the auto-tuner to use a mimimum number of derivatives and multipoles, respectively; by default, as few as 1 of each might be used.

- **authors:** R. Lindberg, R. Soliday, M. Borland (ANL/APS).

8.5 coreEmittance

- **description:** Computes the slice emittance for 80%, 85%, 90%, 95%, and 100% fractions of the beam.
- **synopsis:**

```
coreEmittance -input inputFilename [-nSlices numberOfSlices] [-pngRoot
<string>] [-pngThickness <integer>(2)]
```

- **files:**
  - The input file is a particle output file from `elegant` or a compatible program.
- **switches**
  - `-input` — Specify the name of the input file.
  - `-nSlices` — Optionally specify the number of longitudinal slices. The default is 100.
  - `-pngRoot` — Optionally specify the file rootname for PNG graphics files. If omitted, no PNG files are created.
  - `-pngThickness` — Optionally change the thickness of lines for PNG graphics. The default is 2.
- **author:** X. Dong.

8.6 csrImpedance

- **description:** Computes the steady-state CSR impedance with shielding by parallel plates. By default, the computed impedance is for a dipole magnet that bends the beam in a complete circle.

- **synopsis:**

```
csrImpedance outputFile | -pipe[=out] -height=valueInMeters
-radius=valueInMeters -frequencyLimit=maximum=valueInHz[,minimum=valueInHz]
-n=integer [-filter=cutoff1,cutoff2] [-angle=radians]
```

- **files:**
  - `outputFile` — SDDS file containing computed impedance. May be used directly with `elegant`'s `ZLONGIT` element.

- **switches:**
  - `-height` — The full height of the vacuum chamber, in meters.
  - `-radius` — The radius of the bending magnet, in meters.
  - `-angle` — The angle of the bending magnet, in radians. The default is 2π.
  - `frequencyLimit` — Allows specifying the upper frequency limit (required), as well as the lower frequency limit, for the computed impedance. `elegant` will not accept the data if the lower limit is not 0. If the rms bunch length is σ_t, then it is suggested to have the maximum frequency much greater than $1/sigma_t$.
  - `-n` — Allows specifying the number of data points to be computed. The number of points computed is $2^n + 1$, which is required by `elegant`. A reasonable value is $n = 10$ to $n = 14$.
  - `-filter` — Allows specifying the starting and ending frequency for a simple low-pass filter. The frequencies are given as fractions of the maximum frequency. The filter ramps linearly from 1 to 0 between the two cutoff values. If, for example, the cutoff is 0.2, then the highest frequency for which the impedance is unmodified corresponds to a wavelength of 10 bins (2/0.2) in `elegant`. The intention of this feature is to provide a way to taper the impedance down to reduce high-frequency noise; another option is to apply a gaussian filter externally, e.g., using `sddsprocess`.

- **authors:** Y. Wang, H. Shang, ANL/APS. Based on a simplified form[26] of Warnock's [25] formula.

- **Note:** The script `makeSummedCsrZ` is more convenient for computing the CSR impedance of rings with several types of dipoles, and also handles the free-space case.

8.7 doubleDist6

- **description:** Increases the number of particles in a particle input file by successively doubling the number. Intended to be used to increase the number of particles produced by a photoinjector simulation to improve stability of CSR and LSC simulations.

  The algorithm is as follows:

  – For each doubling, insert a new particle "near" every pair of existing particles in time. The particle has a new t value, but the same (x, xp, y, yp, p) as one of the original particles.

  – Bin the beam according to t into a large number of bins. Randomize the assignment of p values relative to other coordinates across particles in the same bin, while additionally adding a small random value to each p value.

- **synopsis:**

  `doubleDist6 -input` *`name`* `-output` *`name`* `-doublings` *`number`* `-nt` *`bins`*

- **files:**

  – `input` — A particle distribution file, such as might be used with `sdds_beam`.

  – `output` — A particle distribution file, such as might be used with `sdds_beam`.

- **switches:**

  – `-doublings` *`n`* — The number of times to double the size of the distribution. The number of particles in the output file is 2^n times the number in the input file.

  – `-nt` *`bins`* — The number of time bins to use for momentum randomization. This helps to avoid having many particles with exactly same momentum.

- **author:** M. Borland, ANL/APS.

- **see also:** `smoothDist6s`

8.8 haissinski

- **description:** `haissinski` solves the Haissinski equation for the bunch steady-state longitudinal distribution in the presence of various impedances.
- **synopsis:**

```
haissinski twissFile resultsFile
-wakeFunction=file,tColumn=name,wColumn=name |
-model=[L=Henry|Zn=Ohms],R=Ohm -charge=C | -particles=value |
-bunchCurrent=A -steps=numberOfChargeSteps -outputLastStepOnly
-RF=Voltage=V,harmonic=value[,phase=offsetInRadians] | -length=s
-harmonicCavity=Voltage=V,factor=harmonicFactor[,phase=radians]
-superPeriods=number -energy=GeV
-integrationParameters=deltaTime=s,points=number,startTime=s,
iterations=number,fraction=value,tolerance=value
```

- **files:**
  - *twissFile* — Twiss output file from `elegant`, including radiation integral calculations.
  - `resultsFile` — SDDS file containing computed bunch longitudinal distributions as columns, along with analysis and conditions as parameters.
- **switches:**
  - `-wakeFunction=`*file*`,tColumn=`*name*`,wColumn=`*name* — Optionally specifies the impedance as a Greens function using values in an SDDS file. The time points must be equi-spaced.
  - `-model=[L=`*Henry*`|Zn=`*Ohms*`],R=`*Ohm* — Optionally specifies the impedance as an inductor `L` or broad-band value `Zn`, along with a resistance `R`.
  - `-charge=`*C* `| -particles=`*value* `| -bunchCurrent=`*A* — Various ways to specify the charge in each bunch.
  - `-steps=`*numberOfChargeSteps* — Number of values of bunch charge to compute up to the value specified with on the just-described options. Using more values can help convergence, as the result of each prior step is used as the starting point for the new step.
  - `-outputLastStepOnly` — Requests output for the last charge step (full charge) only.
  - `-RF=Voltage=`*V*`,harmonic=`*value*`[,phase=`*offset*`] | -length=`*s* — Two ways to specify the nominal bunch length. The phase value is an offset from the synchronous phase, in radians, and is used only when a harmonic cavity is included.
  - `-harmonicCavity=Voltage=`*V*`,factor=`*harmonicFactor*`[,phase=`*radians*`]` — Specifies a harmonic cavity voltage, phase, and the ratio of the harmonic cavity frequency to the main frequency.
  - `-superPeriods=`*number* — Number of superiods of the lattice specified in *twissfile* to simulate. If one has an N cell ring but only gives 1 cell in the input, this value should be N. If one gives the whole ring, this value should be 1.

– `-energy=`*GeV* — Beam energy. If not given, the value in the *twissfile* is used.

– `-integrationParameters=deltaTime=`*s*`,points=`*number*`,startTime=`*s*`,iterations=`*number*`,fraction=`*value*`,tolerance=`*value* — Integration parameters, which must be set. `deltaTime` is the time interval for wake function and charge density evaluation. `points` is the number of time points, while `startTime` is the time (relative to synchronous phase) at which the time region starts. These values must be set by the user based on knowledge of the likely bunch length. For the others, we suggest 1000 iterations, a fraction of 0.01, and a tolerance of 10^{-4}.

- **authors:** L. Emery, M. Borland, ANL/APS.

8.9 ibsEmittance

- **description:** `ibsEmittance` computes growth rates and equilibrium emittances for electron rings due to intrabeam scattering (IBS). It will also integrate the growth rates to show the time evolution of the emittances. The IBS algorithm is based on the Bjorken and Mtingwa's [15] formula, and with an extension of including vertical dispersion. The program can also estimate IBS growth rates for and transport line or linac beam, provided special attention paid to the beam's energy change (splitting RF cavities as needed).

- **examples:** This example computes the IBS equilibrium parameters and the contributions to the growth rates (at equilibrium) vs position in the APS lattice.

```
ibsEmittance aps.twi aps.ibs -charge=5 -coupling=0.02
-rf=voltage=9,harmonic=1296
```

- **synopsis:**

```
ibsEmittance twissFile resultsFile -charge=nC|-particles=value
-coupling=value|-emityInput=value [-emitInput=value] [-deltaInput=value]
[-emit0=value] [-delta0=value] [-superperiods=value] [-isRing=1|0]
-RF=Voltage=MV,harmonic=value|-length=mm [-energy=MeV] [-growthRatesOnly |
-integrate=turns=number[,stepSize=number]] [-noWarning]
```

- **files:** *twissFile* is a twiss parameter file from the `twiss_output` command of `elegant`. You must use the `radiation_integrals` flag in `twiss_output`.

- **switches:**
  - **-charge**, **-particles** — Give the charge (in nanocoulombs) or the number of electrons.
  - **-coupling** — Give the emittance or "coupling" ratio, ϵ_y/ϵ_x.
  - `-emityInput` — Give the initial vertical emittance in meters.
  - **-emitInput** — Give the initial total emittance in meters. If not specified, the value from the parameter `ex0` in *twissFile* is used.
  - **-deltaInput** — Give the initial rms fractional momentum spread. If not specified, the value from the parameter `Sdelta0` in *twissFile* is used.
  - **-emit0**, **-delta0** — Redefine the equilibrium emittance and rms energy spread, if different from what is given in the twiss input file. Can be used, e.g., to include additional source of energy spread, such as microwave instability, from an external calculation.
  - **-superperiods=*value*** — If given, the number of superperiods in the lattice. *twissFile* is taken to pertain to a single sector.
  - **-isRing** — Specify the calculation is done for stored beam (isRing=1, default) or transport line/linac beam (isRing=0). When isRing is set to 0, the energy scaling and integration calculation will be disabled.
  - **-RF=Voltage=*MV*,harmonic=*value*** — Specify rf voltage and harmonic number.
  - **-length=*mm*** — Specify the rms bunch length.

- **-energy=*MeV*** — Specify the beam energy. By default, this is taken from the `pCentral` parameter in *twissFile*.
- **-growthRatesOnly** — If given, only the initial growth rates are computed. Equilibrium emittance values are not computed. *resultsFile* will contain columns of initial growth rate contributions from individual elements. Without this option, *resultsFile* would normally contain columns of growth rate contributions at equilibrium.
- **-integrate=turns=*number*[,stepSize=*number*]** — If given, then *resultsFile* contains the result of integrating the differential equations for the emittances for the given number of turns and not the contributions of individual elements of growth rates. The step size is the number of turns for each integration step, and can be adjusted to get faster results. The options -growthRatesOnly and -integrate are not compatible.
- **-noWarning** — Removes warning messages.

• **author:** A. Xiao, L. Emery, M. Borland, ANL/APS.

8.10 ionTrapping

- **description:** Computes ion trapping conditions using `elegant` twiss parameter output as input.
- **synopsis:**

```
ionTrapping -twiss filename -superPeriods number -kappa ratio -output
filename -current mA -bunches number
```

- **switches:**
  - `twiss` — Give the name of a Twiss output file from `elegant`. It is advisable to subdivide the elements finely enough to get smooth representations of the lattice functions. The file should be computed the radiation integrals turned on, since the natural emittance and energy spread are needed.
  - `superPeriods` — Give the number of superperiods of the basic cell described by the Twiss output file.
  - `kappa` — Give the ratio ϵ_y/ϵ_x. The emittances are computed from ϵ_0 using $\epsilon_x = \frac{\epsilon_0}{1+\frac{J_y}{J_x}\kappa}$ and $\epsilon_y = \kappa\epsilon_x$.
  - `output` — Give the name of the output file. The file contains the information in the input file, with the following added elements, among others:
    * Column `Acrit` — $A_{crit}(s)$ is defined as[49]

$$A_{crit}(s) = \frac{N_e r_p S_b}{2\min(\sigma_x(s), \sigma_y(s))(\sigma_x(s) + \sigma_y(s))}, \tag{14}$$

      where N_e number of electrons per bunch, r_p is the classical proton radius, S_b is the bunch separation in meters, $\sigma_x(s)$ is the local horizontal rms beam size, and $\sigma_y(s)$ is the local vertical rms beam size. Any singly-ionized species with atomic mass greater than A_{crit} will be trapped.
    * Parameters `ex`, `ey` — The horizontal and vertical emittances.
    * Parameter `AcritMin` — Minimum value of $A_{crit}(s)$.
    * Parameters *species*`TrappedFraction`, where *species* is `H2`, `H2O`, `CH4`, `CO`, and `CO2`. These give the fraction of the circumference over which H_2, H_2O, CH_4, CO, and CO_2, respectively, are trapped.
  - `current` — Give the total beam current milliAmps.
  - `bunches` — Give the number of bunches.
- **authors:** M. Borland (ANL/APS).

8.11 elegantto

- **description:** `elegantto` translates an `elegant`-style (or a MAD file, with some restrictions) into formats accepted by other programs, such as COSY, PARMELA, PATPET, PATRICIA, TRANSPORT, and XORBIT. Will also generate an SDDS file containing lattice data.
- **examples:** The following command would translate the `elegant` lattice file `lattice.lte` into a TRANSPORT lattice file with 10mm quadrupole aperture and 5mm sextupole aperture, at an energy of 1.5 GeV.

  ```
  elegantto lattice.lte lattice.trin -transport=10,5,1.5
  ```

- **synopsis:**

  ```
  elegantto inputfile outputfile {-patricia | -patpet |
  -transport[=quadAper(mm),sextAper(mm),p(GeV/c)] |
  -parmela[=quadAper(mm),sextAper(mm),p(GeV/c)] | -sdds[=p(GeV/c)]  |
  -cosy=quadAper(mm),sextAper(mm),p(MeV/c)  | -xorbit | -mad8 }
  [-angle_tolerance=value] [-flip_k_signs] [-magnets=filename]
  [-header=filename] [-ender=filename]
  ```

- **files:**
  - *inputfile* — An `elegant`-style lattice file.
  - *outputfile* — A file containing lattice data in the chosen format.
- **switches:**
  - **-cosy** — Provide data for the program COSY INFINITY. This can take a little while as the program must figure out the Enge coefficients that correspond to the `FINT` and `HGAP` values for all the dipoles. The user should test the output carefully.
  - **-mad8** — Provide data for the program MAD8.
  - **-patricia** — Provide data for the program PATRICIA.
  - **-patpet** — Provide data for the program PATPET, a merging of the programs PATRICIA and PETROS.
  - **-transport**[=***quadAper(mm)***,***sextAper(mm)***,***p(GeV/c)***] — Provide data for the program TRANSPORT (original style). One may give apertures for the quadrupoles and sextupoles, as well as the beam momentum in GeV/c.
  - **-parmela**[=***quadAper(mm)***,***sextAper(mm)***,***p(GeV/c)***] — Provide data for the program PARMELA. One may give apertures for the quadrupoles and sextupoles, as well as the beam momentum in GeV/c.
  - **-sdds**[=***p(GeV/c)***] — Provide data in SDDS form. One may give the beam momentum in GeV/c.
  - **-angle_tolerance=*value*** — PATPET and PATRICIA only allow sector and rectangular bends. This tolerance, in radians, determines how far from sector or rectangular a bend definition may be and still get processed.

- **-flip_k_signs** — Changes the signs of all quadrupoles.
- **-magnets=*filename*** — Results in output of an additional SDDS file with the magnet layout. This is the same file that would be generated by the `magnets` field of the `run_setup` command in `elegant`.
- **-header=*filename***, **-ender=*filename*** — Allow specification of files to be prepended and appended to the lattice output. For example, if additional commands are required prior to the lattice definition to set up the run, they would be put in the **header** file. If additional commands are needed after the lattice definition to initiate processing, they would be put in the `ender` file.

• **author:** M. Borland, ANL/APS.

8.12 sddsanalyzebeam

- **description:** `sddsanalyzebeam` analyzes a beam of macro-particles and produces an SDDS file containing beam moments, emittances, equivalent beta functions, etc. The beam file is of the type written by `elegant` using the `output` field of the `run_setup` command, or the WATCH element.

- **examples:**

```
sddsanalyzebeam run.out run.analysis
```

- **synopsis:**

```
sddsanalyzebeam [-pipe=[input][,output]] [inputfile] [outputfile]
[-nowarnings] [-correctedOnly] [-canonical]
```

- **files:**
  - *inputfile* — An SDDS file containing the columns `x`, `xp`, `y`, `yp`, `t`, and `p`, giving the six phase-space coordinates for a set of macroparticles. This file can be produced from `elegant`, for example, using the `output` field of the `run_setup` command, the `bunch` field of the `bunched_beam` command, or the WATCH element in coordinate mode.
  - *outputfile* — An SDDS file containing columns giving moments, emittances, equivalent Twiss parameters, and so on, for the macro-particles. Each row of this file corresponds to a page of the input file. The names and meanings of the columns are identical to what is used for `elegant`'s `final` output file from the `run_setup` command. The file from `elegant`, however, stores the results as parameters instead of columns; to convert `outputfile` to that convention, use the SDDS toolkit program `sddsexpand`.

- **switches:**
  - `pipe` — The standard SDDS Toolkit pipe option.
  - `nowarnings` — Suppressses warning messages.
  - `correctedOnly` — If given, only the "corrected" twiss parameters and emittances are computed and output. The corrected twiss parameters have the dispersive component subtracted. Normally, these are computed but given names like `betacx`, `ecx`, etc. whereas the uncorrected values are `betax`, `ex`, etc. The corrected parameters are the correct ones to match a beamline to, since they have the dispersive and mono-energetic terms properly separated. The uncorrected values are more relevant if the dispersion is spurious (i.e., uncorrected or due to something like CSR that doesn't admit of correction).
  - `-canonical` — If given, all computations are performed using canonical momenta $q_x = p_x/p_0 = x'(1+\delta)/\sqrt{1+x'^2+y'^2}$ etc.

- **author:** M. Borland, ANL/APS.

8.13 sddsbrightness

- **description:** `sddsbrightness` computes undulator brightness curves using Twiss parameter data from `elegant` or `sddsanalyzebeam`. Several calculation methods are available.
- **examples:**

```
sddsbrightness run.twi run.bri -harmonics=3
-Krange=start=0.2,end=2.2,points=100
-current=0.1 -totalLength=2.4 -periodLength=0.027 -coupling=0.01
```

```
sddsanalyzebeam run.out -pipe=out -correctedOnly
| sddsbrightness -pipe=in run.bri -harmonics=3
-Krange=start=0.2,end=2.2,points=100
-current=0.1 -totalLength=2.4 -periodLength=0.027 -coupling=0.01
```

- **synopsis:**

```
sddsbrightness [-pipe=[input][,output]] [twissFile] [SDDSoutputfile]
-harmonics=integer -Krange=start=value,end=value,points=integer
-current=Amps -totalLength=meters -periodLength=meters
[-emittanceRatio=value | -coupling=value] [-noSpectralBroadening]
[-method=string,device=string,neks=value]]
```

- **files:**
  - *twissFile* — A Twiss output file from `elegant`, with radiation integral calculations included, or an output from `sddsanalyzebeam`. In the latter case, the `-correctedOnly` option should be used.
  - *SDDSoutputFile* — Contains the brightness data in column form. For each requested harmonic i, there are columns `photonEnergy`*i*, `wavelength`*i*, and `Brightness`*i*.
- **switches:**
  - `pipe` — The standard SDDS Toolkit pipe option.
  - `harmonics` — The number of harmonics to compute.
  - `Krange=start=`*value*`,end=`*value*`,points=`*integer* — The range of the K parameter for the undulator and the number of points to compute on that range.
  - `-current=`*Amps* — The current in amperes. If one gives the average current, one gets the average brightness.
  - `-totalLength=`*meters* — The total length of the undulator, in meters.
  - `-periodLength=`*meters* — The period length of the undulator, in meters.
  - `-emittanceRatio=`*value* `| -coupling=`*value* — In the case of a twiss output file from `elegant`, which does not contain the vertical emittance, one must supply one of these options. If `-emittanceRatio=`*R* is given, $\epsilon_y = \epsilon_0 * R$ and $\epsilon_x = \epsilon_0$; this isn't how things work physically, but is provided for historical reasons. If `-coupling=`*k* is given, $\epsilon_x =$

$\epsilon_0/(1+Jy*k/Jx)$ and $\epsilon_y = k*\epsilon_x$. ϵ_0 is the equilibrium emittance from the twiss output of `elegant`.

In the case of twiss output from `sddsanalyzebeam`, both emittances are present and these options are ignored.

– `-method=`*string*`,device=`*string*`,neks=`*value*`]` — Choose which method to use for brightness calculations. Options are

  * `borland` — M. Borland's approximation method. Fast, but not as reliable as others.
  * `dejus` — R. Dejus' non-zero emittance, infinite-N+convolution method. This is the default.
  * `walkerinfinite` — R. Walker's method. Dejus' method is derived from this method.
  * `walkerfinite` — R. Walker's method using finite N without convolution. This is quite slow.

  The `device` qualifier may be `planar` or `helical`. `neks` is used to change the number of points used for finding the peak of the distribution.

• **authors:** M. Borland, H. Shang, R. Dejus (ANL).

8.14 sddsbunchingfactor

- **description:** `sddsbunchingfactor` computes bunching factors for beams from `elegant`, e.g., from `WATCH` elements in coordinate mode or the `output` file from `run_setup`.

  The bunching factor $B(\omega)$ is defined as

  $$B(\omega) = \frac{1}{N}\sqrt{\left(\sum_{i=1}^{N}\cos\omega t_i\right)^2 + \left(\sum_{i=1}^{N}\sin\omega t_i\right)^2}, \tag{15}$$

  where ω is the angular frequency and t_i is the time coordinate of the i^{th} of N particles.

- **examples:**

```
sddsbunchingfactor run.out run.bfac -omegaRange=1e9,1e12 -points=300
-mode=log
```

- **synopsis:**

```
sddsbunchingfactor [-pipe=[input][,output]] [SDDSinputfile]
[SDDSoutputfile>] [-omegaRange=lowerHz,upperHz] [-points=number]
[-mode={linear|logarithmic}] [-combinePages]
```

- **switches:**
  - `pipe` — The standard SDDS Toolkit pipe option.
  - `omegaRange` — Give the range of ω values, in Hz.
  - `points` — Give the number of points over the range of ω values.
  - `mode` — Choose linear or logarithmic spacing of ω values.
  - `combinePages` — Pages of the input file are combined, i.e., treated as a single bunch.

- **authors:** M. Borland (ANL).

8.15 sddsemitproc

- **description:**

  `sddsemitproc` analyzes quadrupole scan emittance measurement data. It accepts a file containing the transport matrix for each data point and measured beam sizes. Because `sddsemitproc` uses the matrix rather than a thin-lens model, it can analyze data from arbitrarily complex scans, involving, for example, multiple thick-lens quadrupoles.

  The matrix data can be prepared using `elegant`. For example, the `vary_element` command can be used to vary one or more quadrupoles. In addition, the beam size data may be prepared using `elegant`, to allow simulation of emittance measurements.

  `sddsemitproc` will perform error analysis using a Monte Carlo technique. A user-specified number of random error sets are generated and added to all measurements. Analysis is performed for each error set. Statistics over all the error sets provide most likely values and error bars.

  The beam parameters computed by `sddsemitproc` pertain to the beginning of whatever system is simulated in `elegant`.

- **examples:**

```
elegant quadScan.ele sddscollapse quadScan.fin -pipe=out
| sddsxref -pipe=in quadScan.data -take=SigmaX,SigmaY
| sddsemitproc -pipe=in emitResults.sdds
```

- **synopsis:**

```
sddsemitproc [inputfile] [outputfile] [-pipe=[input][,output]]
[-sigmaData=xName,yName] [-variableName=columnName] [-errorData=xName,yName
| -errorLevel=valueInm,[{gaussian,nSigmas | uniform}]] [-nErrorSets=number]
[-seed=integer] [-limitMode=resolution | zero[,reject]
[-deviationLimit=xLevelm,yLevelm] [-resolution=xResolutionm,yResolutionm]
[-verbosity=level]
```

- **files:**

  – *inputfile* — An SDDS file containing one or more pages with columns named R*ij*, where *ij* is 11, 12, 33, and 34. These give elements of the horizontal and vertical transport matrices from the beginning of a system to the observation point. The sigma matrix inferred will be that for the beginning of the system. Typically, one starts with the `final` file from the `run_setup` command in `elegant`, and collapses it using `sddscollapse`. Each page of *inputfile* corresponds to a different emittance measurement.

    In addition to this data, *inputfile* must also contain columns giving the rms beam sizes in x and y. The user supplies the names of the columns using the `-sigmaData` option; otherwise, they default to `Sx` and `Sy`. These columns may be from `elegant` (e.g., `Sx` and `Sy`), if one wants to simulate an emittance measurement. Note that the theory behind the emittance measurement is strictly correct only for true RMS beamsize measurements. Use of FWHM or some other measure will give unreliable results.

- *outputfile* — A file containing one page for each page of *inputfile*. The parameters of *outputfile* give the measured geometric rms emittance, sigma matrix, and Twiss parameters of the beam in the horizontal and vertical planes. If error sets were requested (using `-nErrorSets`), then there are also parameters giving the error bars ("sigma's") of the measured values.

- **switches:**
  - `-variableName=`*columnName* — Supplies the name of a column in *inputFile* that will be copied into *outputFile* for use in plotting. Does not affect any results.
  - `-sigmaData=`*xName*`,`*yName* — Supplies the names of the columns in *inputfile* from which the x and y rms beam sizes are to be taken. Default values are `Sx` and `Sy`, which are the data provided by `elegant`.
  - `-errorLevel=`*valueInm*`,[gaussian,`*nSigmas* `| uniform]` — Supplies the standard deviation of random errors to be added to the measured beam sizes for Monte Carlo error analysis.
  - `-errorData=`*xName*`,`*yName* — May be used to supply the names of columns in the input file that contain the error level for each measurement. This is an option instead of using `-errorLevel`, which allows varying the measurement error for each point.
  - `-nErrorSets=`*number* — The number of sets of random errors to generate and add to the measurements. Each error set is used to perturb the original measurement data. The results are analyzed separately for each error set, then combined to give means and error bars.
  - `-seed=integer` — Seed for the random number generator. Recommend a large, positive, odd integer less than 2^31. If no seed is given or if the given seed is negative, then a seed is generated from the system clock.
  - `-resolution=`*xResolutionm*`,`*yResolutionm* — The resolution of the beam size measurements, in meters. These values are subtracted in quadrature from the measured beam sizes to obtain the true beam sizes.
  - `-limitMode=resolution | zero[,reject]` — If measured or perturbed beam sizes are less than the resolution or less than zero, then errors will result. One can use this option to limit minimum beam size values or reject points. In general, if one has to do this the measurement is probably bad.
  - `-deviationLimit=`*xLevelm*`,`*yLevelm* — Specifies the maximum deviation, in meters, from the fit that data points may have and still be included. An initial fit is performed for each randomized set or the raw data, as appropriate. Outliers are then removed and the fit is repeated.
  - `-verbosity=`*level* — Higher values of *level* result in more informational printouts as the program runs.

- **author:** M. Borland, ANL/APS.

8.16 sddsfindresonances

- **description:** `sddsfindresonances` scans frequency map analysis data and identifies resonances.
- **examples:**

```
sddsfindresonances run.fma run.res -multipoles=dipole,quad,sext,oct
-type=skew sddsfindresonances run.fma run.res -multipoles=sext,oct
-type=skew,norm
```

- **synopsis:**

```
sddsfindresonances [-pipe=[input][,output]] [inputFile] [outputfile]
-multipoles=[all=integer]|[dipole,][quadrupole,][sextupole,][octupole,]
[-type=[normal,][skew]] [-variables=firstColumn,secondColumn]
```

- **files:**
  - *inputFile* — By default, frequency map analysis output file from `elegant`'s `frequency_map` command or equivalent, containing at minimum the columns `x`, `y`, `nux`, and `nuy`. Each page of the file is treated separately.
  - *outputFile* — Contains the identified resonance lines, one resonance line per page. The file contains the columns `x`, `y`, `nux`, and `nuy`, along with parameters that identify the resonance.
- **switches:**
  - `pipe` — The standard SDDS Toolkit pipe option.
  - `multipoles=[all=`*integer*`]|[dipole,][quadrupole,][sextupole,][octupole,]` — Choose what order of resonances to search for by naming the type of magnet that nominally drives it, or by giving the maximum order to search (`all` option).
  - `-type=[normal,][skew]` — Specify normal- or skew-driven resonances. Default is both.
  - `-variables=`*firstColumn,secondColumn* — Use to change the default names for the coordinate variables.
- **authors:** H. Shang, M. Borland. (ANL).

8.17 sddsfluxcurve

- **description:** `sddsfluxcurve` computes undulator fluxcurve curves using Twiss parameter data from `elegant` or `sddsanalyzebeam`. Several calculation methods are available.
- **examples:**

```
sddsfluxcurve run.twi run.bri -harmonics=3
-electronBeam=current=0.1,coupling=0.01
-undulator=period=0.033,numberOfPeriods=70,kmin=0.01,kmax=2.7,points=100
-pinhole=distance=30,xsize=0.0025,ysize=0.001
```

- **synopsis:**

```
sddsfluxcurve [-pipe=[input][,output]] [twissFile] [SDDSoutputfile]
[-harmonics=integer] [-method=methodName[,neks=integer]]
[-mode=pinhole|density|total]
-undulator=period=meters,numberOfPeriods=integer,kmin=value,kmax=value[,points=number]
[-electronBeam=current=amps,[,coupling=value | emittanceRatio=value]]
[-pinhole=distance=meters,xsize=meters,ysize=meters
[,xnumber=integer][,ynumber=integer][,xposition=meters][,yposition=meters]]
[-nowarnings]
```

- **files:**
  - *twissFile* — A Twiss output file from `elegant`, with radiation integral calculations included, or an output from `sddsanalyzebeam`. In the latter case, the `-correctedOnly` option should be used.
  - *SDDSoutputFile* — Contains the flux data in column form. For each requested harmonic *i*, there are columns `photonEnergy`*i* and `wavelength`*i*, plus a column for the flux (`TotalFlux`*i*, `PinholeFlux`*i*, or `FluxDensity`*i*).
- **switches:**
  - `pipe` — The standard SDDS Toolkit pipe option.
  - `harmonics` — The number of harmonics to compute.
  - `-method=`*string*`,neks=`*value*`]` — Choose which method to use for calculations. Options are
    - `dejus` — R. Dejus' non-zero emittance, infinite-N+convolution method. This is the default.
    - `walkerinfinite` — R. Walker's method. Dejus' method is derived from this method.

    `neks` is used to change the number of points used for finding the peak of the distribution.
- `mode=pinhole|density|total` — Specify whether to compute the flux through a pinhole, the flux density, or the total flux.

- `-undulator=period=`*meters*`,numberOfPeriods=`*integer*`,kmin=`*value*`,kmax=`*value*`[,points=`*number*`]` — Specify undulator parameters. `points` is the number of K values to use on the interval $[K_{min}, K_{max}]$.
- `electronBeam=current=`*amps*`,[,coupling=`*value* `| emittanceRatio=`*value*`]` — Specify parameters of the electron beam. The current defaults to 0.1 A. Either the coupling or emittance ratio must be given, unless the input file contains the parameter `ey0` or the column `ey`.
- `-pinhole=distance=`*meters*`,xsize=`*meters*`,ysize=`*meters*`[,xnumber=`*integer*`]` `[,ynumber=`*integer*`][,xposition=`*meters*`][,yposition=`*meters*`]` — Specify the parameters of the pinhole. Required for `-mode=pinhole`. By default `xnumber`=20, `ynumber`=20, `xposition=0`, and `yposition=0`.
- **authors:** M. Borland, H. Shang, R. Dejus (ANL).

8.18 sddsmatchtwiss

- **description:** `sddsmatchtwiss` transforms a beam of macro-particles to match to given beta functions and dispersion. This can be useful in taking macro-particle data from one simulation and using it in another. For example, a beam file from PARMELA could be given the right beta functions for use with a specific lattice in an `elegant` run, saving the trouble of rematching to join the two simulations. Similarly, a beam from `elegant` could be matched into an FEL simulation.

- **examples:**

```
sddsmatchtwiss elegantBeam.out FELBeam.in -xPlane=beta=1.0,alpha=-0.2
-yPlane=beta=0.5,alpha=0.2
```

- **synopsis:**

```
sddsmatchtwiss [-pipe=[input][,output]] inputfile outputfile
[-saveMatrices=filename] [-loadMatrices=filename]
[-xPlane=[beta=meters,alpha=value][,etaValue=meters][,etaSlope=value]]
[-yPlane=[beta=meters,alpha=value][,etaValue=meters][,etaSlope=value]]
[-zPlane=[deltaStDev=value][,tStDev=value]
  [,correlation=seconds|alpha=value][,chirp=1/seconds][,betaGamma=value]]
[-nowarnings]
```

- **files:**

inputfile is an SDDS file containing one or more pages of data giving the phase-space coordinates of macro particles. The macro particle data is stored in columns named `x`, `xp`, `y`, `yp`, and `p`. The units are those used by `elegant` for the `output` file from `run_setup`, the `bunch` file from `bunched_beam`, and the coordinate-mode output from the `WATCH` element. The data from these columns is used together with the commandline arguments to produce new values for these columns; the new values are delivered to `outputfile`. Other columns may be present in `inputfile`; if so, they are passed to *outputfile* unchanged.

- **switches:**
  - `-xPlane=[beta=`*meters*`,alpha=`*value*`][,etaValue=`*meters*`][,etaSlope=`*value*`]` — Specifies the desired parameters for the beam in the horizontal plane. `beta` and `alpha` give β and $\alpha = -\frac{1}{2}\frac{\partial\beta}{\partial s}$; they must both be given or both be omitted. `etaValue` and `etaSlope` give the dispersion, η, and its slope, $\frac{\partial\eta}{\partial s}$.
  - `-yPlane=[beta=`*meters*`,alpha=`*value*`][,etaValue=`*meters*`][,etaSlope=`*value*`]` — Same as `-xPlane`, except for the vertical plane.
  - `-zPlane=[deltaStDev=`*value*`][,tStDev=`*value*`] [,{correlation=`*seconds*`|alpha=`*value*`}][,chirp=`*seconds*`][,betaGamma=`*value*`]` — `deltaStDev` is $\sigma_\delta = \langle\sqrt{(\delta - \langle\delta\rangle)^2}$, `tStDev` is $\sigma_t = \langle\sqrt{(t - \langle t\rangle)^2}$, and `correlation` is $\sigma_{t,\delta} = \langle(\delta - \langle\delta\rangle)((t - \langle t\rangle)\rangle$, in terms of which the longitudinal emittance is $\epsilon = \sqrt{\sigma_t^2 * \sigma_\delta^2 - \sigma_{t,\delta}^2}$. `alpha` is $-\sigma_{t,\delta}/\epsilon$. The `chirp`, if requested, is added after generation of the beam according to the other parameters. If `betaGamma` is given, the beam is "accelerated" to the given average value of $\beta\gamma$ in a idealized sense, preserving the momentum spread and transforming the transverse coordinates by the factor $\sqrt{\langle\beta\gamma\rangle_0/(\beta\gamma)_{\rm desired}}$.

- `-saveMatrices=`*`filename`* — Requests saving the transformation matrices to a file.
- `-loadMatrices=`*`filename`* — Requests loading the transformation matrices from a file.
- `-nowarnings` — Suppresses warning messages.

* **authors:** M. Borland, H. Shang, ANL/APS.

8.19 sddsrandmult

- **description:** `sddsrandmult` computes the multipole errors in a quadrupole or sextupole due to various construction errors. The program is based on the analysis of Halbach[16], with which I'll assume the reader is familiar. Instead of separately evaluating the effect of certain types of mechanical errors, it allows one to simulate several types of errors in order to get statistical distributions for the multipole perturbations.
- **examples:**

  ```
  sddsrandmult quadpert.in
  ```

- **synopsis:**

  `sddsrandmult` *`inputFile`*

- **usage:**

  inputFile is a text file containing a series of namelist commands specifying the parameters of a quadrupole or sextupole, the type and amplitude of the errors to include, and the filenames for output. Each namelist command results in a complete computation and generation of output files.

  The namelist command is `perturbations`. It has the following fields:

  – `type` — A string value, either "quadrupole" (default) or "sextupole".
  – `name` — An optional string value giving the name of the element. This is used in preparing data for `elegant`.
  – `SDDS_output` — An required string value giving the name of an SDDS file to which data for each seed will be written. This file can be used to compute statistics or perform histograms.
  – `elegant_output` — An optional string value giving the name of a text file to which `elegant` commands and element definitions will be written. Note that this file is a mixture of commands and element definitions. As such, the user must manually edit the file and place the appropriate parts in the lattice file and the command file.
  – `kmult_output` — An optional string value giving the name of an SDDS file to which data will be written in the format accepted by the `RANDOM_MULTIPOLES` feature of the `KQUAD` and `KSEXT` elements. *This is the recommended data to use with* `elegant`.
  – `effective_length` — The effective length of the magnet, in meters.
  – `bore_radius` — The bore radius of the magnet, in meters.
  – `reference_radius` — The reference radius for the multipole output, in meters.
  – `dx_pole` — The rms error, in meters, to be imparted to the horizontal position of each pole.
  – `dy_pole` — The rms error, in meters, to be imparted to the vertical position of each pole.
  – `dradius` — The rms error, in meters, in the bore radius.

- dx_split — The rms error, in meters, to be imparted to the horizontal distance between the left and right sides of the magnet.
- dy_split — The rms error, in meters, to be imparted to the vertical distance between the top and bottom halves of the magnet.
- dphi_halves — The rms error, in radians, to be imparted to the relative rotation of the top and bottom halves of the magnet.
- n_cases — The number of cases to simulate (default is 1000).
- n_harm — The number of harmonics to simulate. The default is 0, which results in computing all the harmonics for which Halbach indicates his treatment applies.
- random_number_seed — The initial seed for the random number generator. Should be a large integer.
- long suppress_main_error — If non-zero, harmonics for the main multipole and lower orders are suppressed. It is implicitly assumed that these are correctable through alignment and calibration.

- **author:** M. Borland, ANL/APS.

8.20 sddsurgent

- **description:** `sddsurgent` uses algorithms from the program US (by R. Dejus) and URGENT (by R. Walker) for computation of undulator radiation properties, including power density and intensity distributions.
- **examples:** Take particle data from a tracking run and compute the power density using a 1 mm by 1 mm pinhole for a 72-period, 3.3-cm-period undulator set for a 5 keV first harmonic.

```
sddsanalyzebeam run.out -pipe=out -correctedOnly
| sddsurgent -pipe=in power.sdds -electronbeam=current=0.025
-calc=method=dejus,mode=powerDensity -us
-pinhole=dist=30,xsize=1,ysize=1,xnum=100,ynum=100
-undulator=period=0.033,number=72,energy=5e3
```

- **synopsis:**

```
sddsurgent inputFile outputFile
[-calculation=mode=modeString,method=methodString,harmonics=integer]
[-undulator=period=meters,numberOfPeriods=integer,
kx=value,ky=value,phase=value,energy=eV]
[-electronBeam=current=Amp,energy=GeV,spread=fraction,
xsigma=mm,ysigma=mm,xprime=mrad,yprime=mrad,nsigma=number]
[-pinhole=distance=m,xposition=value,yposition=value,
xsize=value,ysize=value,xnumber=integer,ynumber=integer]
[-alpha=steps=integer,delta=value] [-omega=steps=integer,delta=value]
[-nphi=integer] [-us] [-photonEnergy=maximum=eV,minimum=eV,points=number]
[-nowarnings] [-coupling=value | -emittanceRatio=value]
```

- **files:**
  - *inputFile* — A Twiss output file from `elegant`, with radiation integral calculations included, or an output from `sddsanalyzebeam`. In the latter case, the `-correctedOnly` option should be used with `sddsanalyzebeam`.
  - *outputFile* — Contains the output data, which varies depending on the calculation mode. Use `sddsquery` to view the file contents.
- **switches:**
  - `pipe` — The standard SDDS Toolkit pipe option.
  - `-calculation=mode=`*modeString*`,method=`*methodString*`,harmonics=`*integer*— Choose which calculation to perform and what method to us, as well as the number of undulator harmonics to compute. Values for *modeString* are
    - `1 | fluxDistribution`: Angular/spatial flux density distribution.
    - `2 | fluxSpectrum`: Angular/spatial flux density spectrum
    - `3 | brightness | brilliance`: On-axis brilliance spectrum

    * `4 | pinholeSpectrum`: Flux spectrum through a pinhole
    * `5 | integratedSpectrum`: Flux spectrum integrated over all angles
    * `6 | powerDensity`: Power density and integrated power

  Values for *methodString* are

    * 1: Non-zero emittance; finite-N.
    * 2: Non-zero emittance; infinite-N.
    * `3 | WalkerFinite`: Zero emittance; finite-N.
    * `4 | Dejus`: Non-zero emittance; infinite-N + convolution (Dejus, with `-us` only).
    * `14 | WalkerInfinite`: Non-zero emittance; infinite-N + convolution (Walker, with -tt us only).

- `-emittanceRatio=`*`value`* `| -coupling=`*`value`* — In the case of a twiss output file from `elegant`, which does not contain the vertical emittance, one must supply one of these options. If `-emittanceRatio=`*`R`* is given, $\epsilon_y = \epsilon_0 * R$ and $\epsilon_x = \epsilon_0$. If `-coupling=`*`k`* is given, $\epsilon_x = \epsilon_0/(1+k)$ and $\epsilon_y = k * \epsilon_x$. ϵ_0 is the equilibrium emittance from the twiss output of `elegant`.

  In the case of twiss output from `sddsanalyzebeam`, both emittances are present and these options are ignored.

- `undulator=period=`*`meters`*`,numberOfPeriods=`*`integer`*`,`
  `kx=`*`value`*`,ky=`*`value`*`,phase=`*`value`*`,energy=`*`eV`*] — Specify undulator parameters. If energy (of first-harmonic photons) is given, `kx=0` and `ky` is computed, corresponding to a horizontally deflecting undulator. `phase` specifies the phase difference in degrees for a canted undulator.

- `-electronBeam=current=`*`Amps`*`,energy=`*`GeV`*`,spread=`*`fraction`*`,`
  `xsigma=`*`mm`*`,ysigma=`*`mm`*`,xprime=`*`mrad`*`,yprime=`*`mrad`*`,nsigma=`*`number`*specifies electron beam parameters. Only the current is needed, as other data will be drawn from the input file.

    * `current` — electron beam current in A. (default is 0.1A).
    * `energy` — electron energy in Gev. (default is 7.0Gev).
    * `spread` — electron energy spread.
    * `xsigma` — horizontal RMS beam size (mm)
    * `ysigma` — vertical RMS beam size (mm)
    * `xprime` — horizontal RMS divergence (mrad)
    * `yprime` — vertical RMS divergence (mrad)
    * `nsigma` — no. of standard deviations of electron beam dimensions (size and divergence) to be included.

- `-pinhole=distance=`*`m`*`,xposition=`*`value`*`,yposition=`*`value`*`,`
  `xsize=`*`value`*`,ysize=`*`value`*`,xnumber=`*`integer`*`,ynumber=`*`integer`* — Specifies pinhole parameters. Pinhole parameters are not needed for computing on-axis brilliance (i.e., mode=3).

    * `distance` — distance from the source (m) (distance=0.0 gives angular flux).
    * `xposition` — X-coordinate for center of pinhole (mm) or (mrad for distance=0)
    * `yposition` — Y-coordinate for center of pinhole (mm) or (mrad for distance=0)
    * `xsize` — X-size of pinhole (full width) (mm) or (mrad for distance=0)
    * `ysize` — y-size of pinhole (full width) (mm) or (mrad for distance=0)

    * `xnumber` — Number of subdivisions of pinhole in X (max 500)
    * `ynumber` — Number of subdivisions of pinhole in Y (max 500)
  - `nphi=`*number* — Specifies number of steps in phi between 0 and $\pi/2$. Must be less than 100. used in (calculation mode=1,2,3,4,5 calculation method=1,2).
  - `alpha=steps=`*integer*`,delta=`*value* — Specifies the number of steps in angle alpha (gamma*theta) (¡100). Delta specifies range of angles in alpha^2 to be used, in units of the angular equivalent to 1/N. Used in (mode=1, method=1) and method=3.
  - `omegasteps=`*integer*`,delta=`*value* — Specifies the number of steps in photon energy for the natural lineshape (¡5000). delta specifies range of photon energies to be included in the natural lineshape in units (energy of fundamental/N). The default value covers the range $\pm 2/N$ of the natural lineshape. Used in mode=2,3,4,5 method=1.
  - `photonEnergy=maximum=`*eV*`,minimum=`*eV*`,points=`*number*— Specifies the maximum and minimum photon energy in eV, and the number of energy points to be computed.

- **authors:** H. Shang, R. Dejus, M. Borland, X. Jiao (ANL).

8.21 smoothDist6

- **description:** Increases the number of particles in a particle input file by sampling a simplified distribution based the input file. Intended to be used to increase the number of particles produced by a photoinjector simulation to improve stability of CSR and LSC simulations. Can also add energy and density modulations for performing gain studies.

  The algorithm is as follows:

  1. Fit a 12^{th}-order polynomial to p as a function of t. Evaluate the polynomial at 10,000 equispaced points to generate a lookup table for the momentum variation with time.
  2. Compute the standard deviation of the momentum p_{sd} for blocks of 2,000 successive particles. Fit this data with a 12^{th}-order polynomial and evaluate it a 10,000 equispaced points to generate a lookup table for p_{sd} as a function of t.
  3. Create a histogram of t and smooth it with a low-pass filter having a cutoff at 0.1 THz. This may resulting in ringing at the ends of the histogram, which is clipped off by masking with the original histogram.
  4. Optionally modulate the histogram H(t) with a sinusoid, by multiplying the histogram by $(1 + d_m)\cos 2\pi ct/\lambda_m$, where d_m is the modulation depth and λ_m is the modulation wavelength. For non-zero d_m, this will result in a longitudinal-density-modulated distribution when the histogram is used as a probability distribution and sampled to create time coordinates.
  5. Sample the time histogram N times using a "quiet start" Halton sequence with radix 2, where N is the number of desired particles. The sampling operation is performed by first numerically computing the cumulative distribution function $C(t) = \int_{-\infty}^{t} H(t')dt' / \int_{-\infty}^{\infty} H(t')dt'$. Inverting this to obtain $t(C)$, we generate each sample from $H(t)$ by evaluating $t(U)$, where U is a quantity on the interval [0, 1] generated from the Halton sequence.
  6. Create samples for other coordinates by quiet-sampling of gaussian distributions:
     (a) Scaled transverse coordinates $\hat{x}$, $\hat{x'}$, $\hat{y}$, and $\hat{y'}$ using Halton radices 3, 5, 7, and 11, respectively. For convenience in scaling (step 9), these are defined such that the standard deviation of each coordinate is 10^{-4} and all coordinates are uncorrelated.
     (b) Scaled fractional momentum deviation δ_1 using Halton radix 13, with unit standard deviation.
  7. Interpolate the look-up tables to determine the mean p_{mean} and standard deviation p_{sd} of the momentum at each particle's time coordinate. Use these to compute the individual particle momenta using $p = p_{mean} + \delta_1 p_{sd}$.
  8. Compute the projected transverse rms emittances and Twiss parameters for the original beam.
  9. Transform the scaled transverse phase-space coordinates to give the desired projected Twiss parameters in the x and y planes. The x and y planes are assumed to be uncorrelated.

- **synopsis:**

```
smoothDist6 -input name -output name -factor number -rippleAmplitude %
-rippleWavelength microns  -smoothPasses num(500) -energyMod % -betaSlices n
```

- **files:**
  - `input` — A particle distribution file, such as might be used with `sdds_beam`.
  - `output` — A particle distribution file, such as might be used with `sdds_beam`.
- **switches:**
  - `-factor` *number* — Factor by which to multiply the number of particles.
  - `-rippleAmplitude` *value* — Density ripple amplitude, in percent.
  - `-energyMod` *value* — Energy modulation amplitude, in percent. The wavelength is fixed at 1 μm.
  - `-rippleWavelength` *value* — Density ripple and energy modulation wavelength, in microns.
  - `-betaSlices` *n* — Number of longitudinal slices to use for analysis of twiss parameters. The twiss parameters of the beam will vary step-wise from slice to slice. This discontinuous variation may cause problems (e.g., unstable behavior).
  - `-smoothPases` *num* — Presently ignored.
- **author:** M. Borland, ANL/APS.
- **see also:** `doubleDist6`

8.22 TFBFirSetup

- **description:** `TFBFirSetup` computes FIR (Finite Impulse Response) filter coefficients for use with `TFBDRIVER` elements to perform turn-by-turn transverse feedback. The method uses time-domain least-squares fitting [47].
- **examples:**

```
TFBSetup -twiss Basic.twi -pickup XPICKUP -driver XDRIVER -plane x -output
xfb.param -terms 6
```

- **synopsis:**

```
TFBFirSetup -twiss twissFile -pickup elementName -driver elementName -plane
{x|y} -output filename -terms numberOfTerms
```

- **switches:**
  - `-twiss` — A twiss parameter file from `elegant`. The beamline used for the computations must include a `TFBDRIVER` and `TFBFEEDBACK` element for the plane in question.
  - `-pickup` — Specifies the name of the pickup element in the lattice. One and only one occurrence of the element is required in the *twissFile*. Note that generally the name of the pickup should be all uppercase.
  - `-driver` — Specifies the name of the driver element in the lattice. One and only one occurrence of the element is required in the *twissFile*. Note that generally the name of the driver should be all uppercase.
  - `-plane` — Specifies the plane of the feedback.
  - `-output` — Specifies output filename to which FIR configuration is written. The file should be loaded with `load_parameters`, e.g.,

    ```
    &load_parameters
        filename = xfb.param,
        change_defined_values = 1
    &end
    ```

  - `-terms` — Number of terms in the filter, between 1 and 30, inclusive.
- **author:** M. Borland, ANL/APS.
- **acknowledgments:** H. Shang, C.-Y. Yao.

8.23 touschekLifetime

- **description:** `touschekLifetime` computes Touschek lifetime using A. Piwinski's formula [23, 24]. A longitudinally non-Gaussian distributed bunch lifetime (such as ring with harmonic cavity) can be computed if the bunch profile is inputed through beam option.

- **examples:**

```
touschekLifetime aps.life -twiss=aps.twi -aper=aps.aper -part=2e10
-coupling=0.01 -length=6
```

- **synopsis:**

```
touschekLifetime outputFile -twiss=twissFile -aperture=momentumApertureFile
[-beam=beamProfile | -sliceAnalysis=filename] -charge=nC|-particles=value
{-coupling=value|-emityInput=value} -RF=Voltage=MV,harmonic=value[,limit] |
-length=mm [-emitInput=valueInMeters] [-deltaInput=value] [-verbosity=value]
[-ignoreMismatch] [-deltaLimit=valueInPercent] [-method=0/1]
```

- **files:** *outputFile* — Contains resulting Touschek lifetime.

- **switches:**
  - `-twiss` — A twiss parameter file from `elegant`. You must use the `radiation_integrals` flag in `twiss_output`.
  - `-aperture` — A momentum aperture file from `elegant`. This file can contain a subset of elements of twissFile (for example: only Quadrupole elements). However, the Twiss and momentum aperture files *must* cover the same beamline. Having one file for a part of beamline (e.g., a few sectors) and one for the entire ring will yield incorrect results.
  - `-beam` — Give beam profile file from elegant2genesis. If this option is given, other input beam parameters are ignored. You can use this option to compute touschek lifetime for a non-Gaussian longitudinally distributed bunch.
  - `-sliceAnalysis` — Give slice analysis file from the `SLICE` element in `elegant`. If this option is given, other input beam parameters are ignored. You can use this option to compute touschek lifetime for a non-Gaussian longitudinally distributed bunch.
  - `-charge`, `-particles` — Give the charge (in nanocoulombs) or the number of electrons.
  - `-emitInput` — Give the initial total emittance in meters (if `-coupling` is used) or the initial x emittance in meters (if `-emityInput` is used).. If not specified, the value from the parameter `ex0` in *twissFile* is used.
  - `-coupling` — Give the emittance coupling ratio, ϵ_y/ϵ_x. This is used to compute the horizontal and vertical emittance from the natural emittance.
  - `-emityInput` — Give the vertical emittance in meters.
  - `-deltaInput` — Give the initial rms fractional momentum spread. If not specified, the value from the parameter `Sdelta0` in *twissFile* is used.

- -RF=Voltage=*MV*,harmonic=*value*[,limit] — Specify rf voltage and harmonic number. The limit qualifier, if given means that the momentum acceptance is limited by the bucket half-height. N.B.: If the data files cover only a portion of the ring, using this option will give incorrect results!
- -length=*mm* — Specify the rms bunch length. This is an alternative to giving rf parameters.
- -verbosity — If nonzero, program execution information is printed to the standard output.
- -ignoreMismatch — If given, then mismatch of element names between the twiss and momentum aperture files is ignored. May be useful if there are zero-length elements.
- -deltaLimit — Give the maximum value for the momentum aperture, in percent. If not specified, the values in the momentum aperture input file are used, possibly altered by the use of the -RF option with the limit qualifier. If both -deltaLimit and -RF=limit... are given, the smaller is enforced.
- -method — The integral of Piwinski's formula can be done in two ways. "0" - direct integral of parameter τ, this method is also used in elegant. 1 - substitute variable τ with variable k, with $\tau = \tan^2(k)$. These two methods give you same results.

• **Note:** If using Pelegant to compute the momentum aperture with output_mode=1, it is necessary to first run the script reorganizeMmap to put the data into the form needed by touschekLifetime.

• **author:** A. Xiao, ANL/APS.

8.24 view3dGeometry

- **description:** Allows viewing the 3D geometry of a beamline using the freewrl viewer.
- **examples:** To generate 3d data and view:

  ```
  view3dGeometry -rootname aps -showNames ``*QUAD* *BEN*'' -showCoordinates
  ``*MON*''
  ```

  To view again:

  ```
  freewrl aps.x3d
  ```

- **synopsis:**

  ```
  view3dGeometry -rootname string -showNames listOfElementTypes
  -showCoordinates listOfElementTypes [-nviewpoints number(10)]
  ```

- **input files:**
  - *rootname*.flr — Contains floor coordinate output from `elegant` (`floor_coodinates` command).
  - *rootname*.param — Contains parameter output from `elegant` (`run_setup` command).
- **output files:** *rootname*.x3d — Input data to freewrl.
- **switches:**
  - `-rootname` — Gives the rootname of the run, used to identify the input and output files.
  - `-showNames` — Gives list of element types, with optional wildcards, for which the element name will be shown in the viewer. Default: "*SBEN*".
  - `-showCoordinates` — Gives list of element types, with optional wildcards, for which the local coordinate system will be shown in the viewer. Default: "MARK* WATCH*".
  - `nviewpoints` — Number of viewpoints to generate and embed in file. Moving between viewpoints using keystroke commands is easier than "flying" using the keypad.
- **author:** A. Petrenko, BINP. (Modified by M. Borland.)

9 Accelerator and Element Description

As mentioned in the introduction, `elegant` uses a variant of the MAD input format for describing accelerators. With some exceptions, the accelerator description for one program can be read by the other with no modification. Among the differences:

- `elegant` does not support the use of MAD-style equations to compute the value of a quantity. The `link_elements` namelist command can be used for this purpose, and is actually more flexible than the method used by MAD. Also, `rpn`-style equations may be given in double-quotes; these are evaluated once only when the lattice is parsed.
- `elegant` does not support substitution of parameters in beamline definitions.
- `elegant` contains many elements that MAD does not have, such as kick elements, wake fields, and numerically integrated elements.
- The length of an input line is not limited to 80 characters in `elegant`, as it is in MAD. However, for compatibility, any lattice created by `elegant` will conform to this limit.
- The maximum length of the name of an element or beamline is 100 characters.

`elegant`'s lattice parser translates all input into upper case, except where the input is protected by double quotes. However, various commands (such as `vary_element` or `link_elements`) that accept element names as input do not perform any translation. Hence, when referring to element names in commands, the user must enter the names in upper case unless they are protected by double quotes in the lattice file.

When the lattice file is very complex, it is sometimes convenient to separate it into several files. These can then be imported into a main lattice file using the `#INCLUDE` directive, as in

```
#INCLUDE: part1.lte
#INCLUDE: part2.lte
```

The rules for naming elements and beamlines are as follows:

- The name should start with an alphabetic character (i.e., a-z A-Z).
- The name may contain any of the following characters in addition to alphabetic characters and numbers: `~ @ $ % ^ & - _ + = { } [ ] \ | / ? < > . : |\verb`
- The name should not contain any of the following: `# * ! ‘‘ ’ ‘`
- The name should not contain spaces, tabs, or non-printing characters.

If using unusual characters in a name, it is a good idea to enclose the name in double quotes. This is required if `:` is in the name.

`elegant`'s `print_dictionary` command allows the user to obtain a list of names and short descriptions of all accelerator elements recognized by the program, along with the names, units, types, and default values of all parameters of each element. The present output of this command is listed in the next section. The reader is referred to the MAD manual[2] for details on sign conventions for angles, focusing strength, and so forth.

Comments may be embedded in the lattice file by *starting* a line with an exclamation point ("!"). Rpn expressions may be embedded separately from an element definition by starting a line with a percent sign ("%"). For example

```
! Define Pi (actually "pi" is already defined, but this is just an example)
% 1 atan 4 * sto Pi
% Pi 40 / sto myAngle
! Define a rectangular bend for a ring with 80 equal bends
B1: SBEN,L=1.0,ANGLE="myAngle",E1="myAngle 2 /",E2="myAngle 2 /"
```

Note that to use an RPN expression the value of a parameter, one must enclose the expression in double quotes.

9.1 Magnet Strength

There are many conventions for specifying magnetic fields in terms of a multipole, polynomial, or Taylor expansion, which leads to potential confusion. In elegant (as in MAD[2]), magnet strengths are specified in terms of Taylor series. For normal multipoles and $y = 0$, the expansion is

$$B_y(x, 0) = \sum_{n=0}^{\infty} \frac{B_n x^n}{n!}, \tag{16}$$

where B_0 is the dipole, B_1 is the quadrupole, etc. In general,

$$B_n = \left(\frac{\partial^n B_y}{\partial x^n} \right)_{x=y=0}. \tag{17}$$

`elegant` follows MAD [2] in using a right-handed coordinate system (x, y, z) in which z is along the beam direction, x is to the left, and y is up.

This expansion for the normal multipole terms can be related to a multipole expansion that includes both normal and skew components. In this convention, positive normal multipole coefficients give positive B_y for $x > 0$ and $y = 0$. Rotating a positive normal multipole with N poles π/N clockwise about the vector along the beam direction will convert it into a positive skew multipole. As a result, for a positive skew multipole, B_y will be non-negative and B_x will be negative for $x > 0$ along the line $\phi = \pi/N$.

We can satisfy these conventions if we write the scalar potential as

$$V = \sum_{n=1}^{\infty} \frac{iA_{n-1} - B_{n-1}}{n!} (x + iy)^n e^{-in\Delta\phi}, \tag{18}$$

where, as we'll see, A_m are skew components and B_m are normal components for a $2(m + 1)$-pole. The coordinates (x, y) are in a right-handed system with the longitudinal coordinate z. $\Delta\phi$ is the rotation angle of the magnet, where a clockwise rotation about the nominal trajectory corresponds to $\Delta\phi > 0$. The minus sign in $e^{-in\Delta\phi}$ is because we rotate the magnet while keeping the coordinate system fixed.

The magnetic fields are

$$B_y = -\Im \frac{\partial V}{\partial y} = \Im \sum_{n=0}^{\infty} \frac{A_n + iB_n}{n!} (x + iy)^n e^{-i(n+1)\Delta\phi}, \tag{19}$$

and

$$B_x = -\Im \frac{\partial V}{\partial x} = \Im \sum_{n=0}^{\infty} \frac{-iA_n + B_n}{n!} (x + iy)^n e^{-i(n+1)\Delta\phi}, \tag{20}$$

We can relate the coefficients to the B_m quantities used in `MAD` and `elegant` by noting that for $\Delta\phi = 0$

$$B_m = \left(\frac{\partial^m B_y}{\partial x^m}\right)_{x=y=0} \tag{21}$$

and

$$A_m = -\left(\frac{\partial^m B_x}{\partial x^m}\right)_{x=y=0} \tag{22}$$

Note the minus sign in the last equation, which differs from commonly asserted conventions.

Multipole errors are typically specified as fractions of the main field harmonic at a reference radius R, e.g.,

$$F_n = \frac{K_n R^n/n!}{K_m R^m/m!}, \tag{23}$$

where m is the main harmonic and n is the error harmonic.

For electrons, the deflection from a thin element is

$$\theta(x, y=0) = \frac{1}{H}\int B(x, y=0)dl, \tag{24}$$

where $H = B\rho = -p/e$ is the beam rigidity and $p = m_e c\beta\gamma$ is the momentum. The geometric strengths K_n are defined as

$$K_n = \frac{B_n}{H}. \tag{25}$$

By convention in `elegant`, a positive K_n value deflects a particle at $x > 0$ toward $x = 0$. E.g., a positive K_1 value indicates a horizontally focusing quadrupole.

10 Element Dictionary

10.1 ALPH—An alpha magnet implemented as a matrix, up to 3rd order.

An alpha magnet implemented as a matrix, up to 3rd order.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
XMAX	M	double	0.0	size of alpha
XS1	M	double	0.0	inner scraper position relative to XMAX
XS2	M	double	0.0	outer scraper position relative to XMAX
DP1		double	-1	inner scraper fractional momentum deviation
DP2		double	1	outer scraper fractional momentum deviation
XPUCK	M	double	-1	position of scraper puck
WIDTHPUCK	M	double	0.0	size of scraper puck
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
TILT		double	0.0	rotation about incoming longitudinal axis
PART		short	0	0=full, 1=first half, 2=second half
ORDER		short	0	matrix order [1,3]
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element provides a matrix-based implementation of an alpha magnet [5]. Matrices up to third order are available [4].

The parameter `XMAX` determines the size of the alpha, which is related to the gradient g in the magnet and the central momentum $\beta\gamma$ by

$$x_{max}[m] = 0.07504986\sqrt{\frac{\beta\gamma}{g[T/m]}}. \tag{26}$$

The path length of the central particle is $2.554x_{max}$.

Because an alpha magnet has large dispersion at the midplane, it is often used for momentum filtration in addition to bunch compression. The dispersion at the center is given by the simple relation

$$R_{16} = -\frac{1}{2}x_{max}. \quad (27)$$

To use an alpha magnet for momentum filtration in `elegant`, one must split the alpha magnet into two pieces. One may then either use the scraper features of the `ALPH` element or other elements such as `SCRAPER` or `RCOL`.

To split an alpha magnet, one uses the `PART` parameter. E.g.,

```
! First half, with momentum filter between -5% and +2.5%
AL1: ALPH,XMAX=0.11,PART=1,DP1=-0.05,DP2=0.025
! Second half
AL2: ALPH,XMAX=0.11,PART=2
AL:  LINE=(AL1,AL2)
```

One can also separate the momentum filtration for better visualization

```
! First half, with momentum filter between -5% and +2.5%
AL1: ALPH,XMAX=0.11,PART=1
! Momentum filtration
ALM: ALPH,XMAX=0.11,PART=3,DP1=-0.05,DP2=0.025
! Second half
AL2: ALPH,XMAX=0.11,PART=2
! Phase space monitors
WP: WATCH,FILENAME="%s-%ld.wp",MODE="coordinates"
AL:  LINE=(WP,AL1,WP,ALM,WP,AL2,WP)
```

As just illustrated, the parameters `DP1` and `DP2` may be used to filter the momentum by providing fractional momentum deviation limits. These are implemented in a physical fashion by computing the corresesopnding horizontal position deviations and imposing these as limits on the particle coordinates. One may also do this directly using the `XS1` and `XS2` parameters, which specify maximum acceptable deviations from the nominal horizontal position. `XS1` is the allowed deviation on the low-energy side while `XS2` is the allowed deviation on the high-energy side.

10.2 APCONTOUR—An aperture (or its inverse) defined by (x, y) points in an SDDS file.

An aperture (or its inverse) defined by (x, y) points in an SDDS file.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
RESOLUTION	M	double	0.0	z resolution of finding intersection
XFACTOR		double	1	factor by which to multiply x contour values
YFACTOR		double	1	factor by which to multiply y contour values
INVERT		short	0	if non-zero, contour defines an obstruction rather than an aperture
STICKY		short	0	if non-zero, effect persists in downstream elements until canceled or replaced
CANCEL		short	0	if non-zero, sole effect is to cancel previous sticky APCONTOUR
HOLD_OFF		short	0	if non-zero and STICKY=1, then effect only seen in the next element downstream
FILENAME		STRING	NULL	name of file containing contour data
XCOLUMN		STRING	NULL	name of column containing x data
YCOLUMN		STRING	NULL	name of containing y data
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The `STICKY` parameter results in the aperture contour being applied inside subsequent `CCBEND`, `CSBEND`, `CSRCSBEND`, `KQUAD`, `KSEXT`, `KOCT`, and `KQUSE` elements, as well as at the end of other downstream elements. This continues until another `APCONTOUR` element asserts a new contour, or uses `CANCEL=1` to cancel the feature.

For versions 2022.1 and later, the input file may have multiple pages, each with (x, y) points specifying a closed contour. The combination of the effect of these contours is specified using the `Logic` parameter in the input file, utilizing the logical stack and logical operators of the `rpn` module. For each particle, the program loops over each contour. If the particle is inside (outside) the contour, a `true` (`false`) value is pushed onto the stack, followed by execution of the indicated logic. If the final value is `true`, the particle survives, otherwise it is lost. (The `invert` parameter on the element definition can be used to invert this.)

For example, if the aperture file is

```
SDDS1
&column name=x type=float units=m &end
&column name=y type=float units=m &end
&parameter name=Logic type=string &end
&data mode=ascii no_row_counts=1 &end
""
-0.01   -0.015
-0.01   -0.0075
-0.015   -0.0075
-0.015   -0.015
-0.01   -0.015

"||"
0.01   -0.015
0.01   -0.0075
0.015   -0.0075
0.015   -0.015
0.01   -0.015
```

then particles survive only if inside one of the two rectangles.

10.3 BEAMBEAM—An element to provide kicks from another beam

An element to provide kicks from another beam
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
CHARGE	C	double	0.0	charge of opposing beam
XCENTER	M	double	0.0	horizontal center position of opposing beam
YCENTER	M	double	0.0	vertical center position of opposing beam
XSIZE	M	double	0.0	horizontal size of opposing beam
YSIZE	M	double	0.0	vertical size of opposing beam
DISTRIBUTION	$NULL$	STRING	gaussian	distribution type of opposing beam
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element provides a simple model of beam-beam kicks from a stiff opposing beam defined by the bunch charge, transverse centroids, and transverse sizes.

Three distribution types are supported:

- `gaussian` — A transversely-gausssian charge distribution is used. The parameters `XSIZE` and `YSIZE` are interpreted as the rms sizes of the distribution. The computations are based on the Bassetti-Erskine formula [59].

- `uniform` — A uniformly-filled two-dimensional ellipsoidal distribution is used.

$$\rho(x, y, z) = \frac{Q}{\pi abL} \tag{28}$$

  when $(x/a)^2 + (y/b)^2 \leq 1$ and $|z| \leq L/2$, and zero otherwise. The parameters `XSIZE` and `YSIZE` are interpreted as a and b, respectively. L is the length of the slug of charge, which is assumed to be sufficiently short that the impulse approximation is valid. The computations are based on Furman's expressions [60].

- `parabolic` — A parabolic two-dimensional ellipsoidal distribution is used.

$$\rho(x, y, z) = \frac{2Q}{\pi abL}(1 - (x/a)^2 - (y/b)^2) \tag{29}$$

when $(x/a)^2 + (y/b)^2 \leq 1$ and $|z| \leq L/2$, and zero otherwise. The parameters XSIZE and YSIZE are interpreted as a and b, respectively. L is the length of the slug of charge, which is assumed to be sufficiently short that the impulse approximation is valid. The computations are based on Furman's expressions [60].

10.4 BGGEXP—A magnetic field element using generalized gradient expansion.

A magnetic field element using generalized gradient expansion.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	insertion length
LFIELD	M	double	-1	expected length of the field map. If negative, use L.
FILENAME	$NULL$	STRING	NULL	name of file containing generalized gradient data for normal terms, original convention
NORMAL_FILENAME	$NULL$	STRING	NULL	name of file containing generalized gradient data for normal terms, new convention
SKEW_FILENAME	$NULL$	STRING	NULL	name of file containing generalized gradient data for skew terms, new convention
STRENGTH	$NULL$	double	1	factor by which to multiply field
FACTOR0	$NULL$	double	1	factor by which to multiply field from m=0 (solenoid) terms
FACTOR1	$NULL$	double	1	factor by which to multiply field from m=1 (dipole) terms
FACTOR2	$NULL$	double	1	factor by which to multiply field from m=2 (quadrupole) terms
FACTOR3	$NULL$	double	1	factor by which to multiply field from m=3 (sextupole) terms
FACTOR4	$NULL$	double	1	factor by which to multiply field from m=4 (octupole) terms
BXFACTOR	$NULL$	double	1	factor by which to multiply x component of field. Requires SYMPLECTIC=0.
BYFACTOR	$NULL$	double	1	factor by which to multiply y component of field. Requires SYMPLECTIC=0.

A magnetic field element using generalized gradient expansion.

Parameter Name	Units	Type	Default	Description
BZFACTOR	*NULL*	double	1	factor by which to multiply z component of field. Requires SYMPLECTIC=0.
TILT	*RAD*	double	0.0	rotation about longitudinal axis
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
DZ	*M*	double	0.0	misalignment
BX	*T*	double	0.0	add BX*STRENGTH to Bx field
BY	*T*	double	0.0	add BY*STRENGTH to By field
MAXIMUM_M		short	-1	data with m greater than this is ignored
MAXIMUM_2N		short	-1	data with 2*n greater than this is ignored
Z_INTERVAL		short	1	input z data is sampled at this interval
SYMPLECTIC		short	0	if nonzero, use implicit symplectic integrator. At minimum, should always be used to validate the sufficiency of the non-symplectic integrator.
SYNCH_RAD		short	0	if nonzero, include classical, single-particle synchrotron radiation
ISR		short	0	if nonzero, include incoherent synchrotron radiation (quantum excitation)
PARTICLE_OUTPUT_FILE		STRING	NULL	name of file for phase-space and field output. Use for debugging only!
IS_BEND		short	0	if nonzero, magnet is a bending magnet; vertex, entry, and exit points should be defined.
XVERTEX	*M*	double	0.0	For dipoles: x position of vertex in coordinate system of the fields.

A magnetic field element using generalized gradient expansion.

Parameter Name	Units	Type	Default	Description
ZVERTEX	M	double	0.0	For dipoles: z position of vertex in coordinate system of the fields.
XENTRY	M	double	0.0	For dipoles: x position of reference entry point in coordinate system of the fields.
ZENTRY	M	double	0.0	For dipoles: z position of reference entry point in coordinate system of the fields.
XEXIT	M	double	0.0	For dipoles: x position of reference exit point in coordinate system of the fields.
ZEXIT	M	double	0.0	For dipoles: z position of reference exit point in coordinate system of the fields.
DXEXPANSION	M	double	0.0	x position of the generalized gradient expansion relative to the reference trajectory.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates transport through a 3D magnetic field specified in terms of a generalized gradient expansion [50]. After reconstructing the field, it simply integrates the equations of motion based on the Lorentz force equation in cartesian coordinates.

The generalized gradients are provided in SDDS files. In addition to several columns describing the gradients, the file must contain a parameter:

- `m` — The multipole index, using the convention where $m = 0$ is solenoid, $m = 1$ is dipole, $m = 2$ is quadrupole, etc. N.B.: this convention conforms with [50] but is not the usual one used by `elegant`. This should be stored as a short integer.

The files may also include optional parameters `xCenter` and `yCenter` giving the center of the expansion in meters.

In the original implementation, which is still supported, only normal field components were included In that case, the user should use the `FILENAME` field to provide a file with the following floating-point columns:

- `z` — Longitudinal coordinate. Units should be "m".
- `Cnm`n — The n^{th} generalized gradient of the m^{th} harmonic, where $n = 0, 2, 4, ...$ There is no preset limit to the number of generalized gradients. Units are ignored, but should be SI.

- `dCnm`n`/dz` — The longitudial derivative of the n^{th} generalized gradient, for the m^{th} harmonic, where $n = 0, 2, 4, ...$ The number of derivatives must match the number of generalized gradients **Cnm*n***.

The field expansion in this case is

$$
\begin{aligned}
B_r &= \sum_{m=1}^{\infty}\sum_{n=0}^{\infty} \frac{(-1)^n m!(2n+m)}{4^n n!(n+m)!} r^{2n+m-1} \left\{ C_m^{[2n]}(z)\sin m\phi \right\} \\
B_\phi &= \sum_{m=1}^{\infty}\sum_{n=0}^{\infty} \frac{(-1)^n m!(2n+m)}{4^n n!(n+m)!} r^{2n+m-1} \left\{ C_m^{[2n]}(z)\cos m\phi \right\} \\
B_z &= \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \frac{(-1)^n m!}{4^n n!(n+m)!} r^{2n+m} \left\{ C_m^{[2n+1]}(z)\sin m\phi \right\}
\end{aligned}
\tag{30}
$$

where it is understood that the expansion is about the `xCenter` and `yCenter` values, if given.

Note that there is potential confusion between the `xCenter` parameter in the input files and the `DXEXPANSION` parameter in the element definition. These provide similar functionality and only one is needed. Both give the position of the horizontal center of the expansion relative to the magnetic field coordinate system.

In version 2020.5 and later, both normal and skew expansions are supported. In this case, the user may provide filenames via the `NORMAL_FILENAME` and `SKEW_FILENAME` fields. In this, case, the files must contain the following floating-point columns:

- `z` — Longitudinal coordinate. Units should be "m".
- `CnmS`n (normal) or `CnmC`n (skew) — The n^{th} generalized gradient of the m^{th} harmonic, where $n = 0, 2, 4, ...$ There is no preset limit to the number of generalized gradients. Units are ignored, but should be SI. Note that the "S" in the name for the normal components make be confusing. It refers to the fact that these terms appear in the potential with a factor of $\sin m\phi$, whereas the skew terms have $\cos m\phi$ factors (hence the "C").
- `dCnmS`n`/dz` (normal) or `dCnmC`n`/dz` (skew) — The longitudial derivative of the n^{th} generalized gradient, for the m^{th} harmonic, where $n = 0, 2, 4, ...$ The number of derivatives must match the number of generalized gradients **Cnm*n***.

The field expansion in this case is

$$
\begin{aligned}
B_r &= \sum_{m=1}^{\infty}\sum_{n=0}^{\infty} \frac{(-1)^n m!(2n+m)}{4^n n!(n+m)!} r^{2n+m-1} \left\{ C_{m,s}^{[2n]}(z)\sin m\phi + C_{m,c}^{[2n]}(z)\cos m\phi \right\} + \sum_{n=1}^{\infty} \frac{(-1)^n 2n}{4^n n! n!} r^{2n-1} C_{0,c}^{[2n]}(z) \\
B_\phi &= \sum_{m=1}^{\infty}\sum_{n=0}^{\infty} \frac{(-1)^n m!(2n+m)}{4^n n!(n+m)!} r^{2n+m-1} \left\{ C_{m,s}^{[2n]}(z)\cos m\phi - C_{m,c}^{[2n]}(z)\sin m\phi \right\} \\
B_z &= \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \frac{(-1)^n m!}{4^n n!(n+m)!} r^{2n+m} \left\{ C_{m,s}^{[2n+1]}(z)\sin m\phi + C_{m,c}^{[2n+1]}(z)\cos m\phi \right\}
\end{aligned}
\tag{31}
$$

where it is understood that the expansion is about the `xCenter` and `yCenter` values, if given. Users should note that the skew field sign convention used by [50] and `BGGEXP` differs from that used in `elegant`. In particular, to convert a normal field to a skew field while conforming to `elegant`'s conventions, one must use $C^p_{m,s} \to -C^p_{m,c}$ and $dC^p_{m,s}/dz \to -dC^p_{m,c}/dz$.

Data for use with `BGGEXP` can be prepared with the programs `computeCBGGE` (section 8.3) and `computeRBGGE` (section 8.4), which are distributed with `elegant`.

Synchrotron radiation can be included by setting `SYNCH_RAD=1` for classical radiation only and also `ISR=1` for incoherent (quantum) effects. This will impact the results of `moments_output` calculation as well as tracking.

Important notes and limitations:

1. The calculations of `twiss_output`, including radiation integrals, are at this point not affected, nor is the setup of rf cavities for storage rings via the `rf_setup` command.

2. The symplectic integrator, in addition to being symplectic, is typically more accurate than the non-symplectic integrator. It is also considerably slower. However, at minimum, users should use the symplectic integrator to verify that the accuracy of the non-symplectic integrator is adequate.

3. The `BX` and `BY` parameters allow imposing uniform horizontal and vertical magnetic fields on the device. This can be helpful if the terminal trajectory deviates from the expected value, e.g., an on-axis particle ends up off-axis. This may happen if the device has a dipolar field that is truncated at the ends before it has decayed sufficiently. Note that these values are multiplied by the `STRENGTH` factor before being applied to the beam.

In addition to the `STRENGTH` factor, there are five parameters that can be used to scale multipoles of different orders: `FACTOR0`, `FACTOR1`, `FACTOR2`, `FACTOR3`, and `FACTOR4` scale the solenoidal, dipolar, quadrupolar, sextupolar, and octupolar fields, respectively. The `BXFACTOR`, `BYFACTOR`, and `BZFACTOR` allow multiplying the indicated field components by the given factors. The the exception of the `STRENGTH` factor, these scaling parameters may result in unphysical fields.

If `IS_BEND` is non-zero, the magnet is assumed to be a bending magnet, in which case additional parameters are required.

- `ZVERTEX`, `XVERTEX` — Coordinates of the vertex point in coordinate frame of the field data. For a symmetric dipole, `ZVERTEX` is typically zero, while `XVERTEX` would be the displacement of the vertex point from the cylinder axis.

- `ZENTRY`, `XENTRY` — Coordinates of the nominal entry plane.

- `ZEXIT`, `XEXIT` — Coordinates of the nominal exit plane.

10.5 BMAPXY—A map of Bx and By vs x and y.

A map of Bx and By vs x and y.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
STRENGTH	*NULL*	double	0.0	factor by which to multiply field
ACCURACY	*NULL*	double	0.0	integration accuracy
METHOD	*NULL*	STRING	NULL	integration method (runge-kutta, bulirsch-stoer, modified-midpoint, two-pass modified-midpoint, leap-frog, non-adaptive runge-kutta
FILENAME	*NULL*	STRING	NULL	name of file containing columns (x, y, Fx, Fy) giving normalized field (Fx, Fy) vs (x, y)
FX	*NULL*	STRING	NULL	rpn expression for Fx in terms of x and y
FY	*NULL*	STRING	NULL	rpn expression for Fy in terms of x and y
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates transport through a transverse magnetic field specified as a field map. It does this by simply integrating the Lorentz force equation in cartesian coordinates. It does not incorporate changes in the design trajectory resulting from the fields. I.e., if you input a dipole field, it is interpreted as a steering element.

The field map file is an SDDS file with the following columns:

- **x**, **y** — Transverse coordinates in meters (units should be "m").
- **Fx**, **Fy** — Normalized field values (no units). The field is multiplied by the value of the STRENGTH parameter to convert it to a local bending radius. For example, if Fx=y and Fy=x, then STRENGTH is the K1 quadrupole parameter.
- **Bx**, **By** — Field values in Tesla (units should be "T"). The field is still multiplied by the value of the STRENGTH parameter, which is dimensionless. Note: the default value of STRENGTH is 0, so if you don't set it to something, you'll get no effect!

The field map file must contain a rectangular grid of points, equispaced (separately) in x and y. There should be no missing values in the grid (this is not checked by `elegant`). In addition, the x values must vary fastest as the values are accessed in row order. To ensure that this is the case, use the following command on the field file:

sddssort *fieldFile* -column=y,incr -column=x,incr

10.6 BMXYZ—A map of (Bx, By, Bz) vs (x, y, z), for straight elements only

A map of (Bx, By, Bz) vs (x, y, z), for straight elements only
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	insertion length
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
TILT	RAD	double	0.0	rotation about longitudinal axis
LFIELD	M	double	-1	expected length of the field map. If negative, determined from field data.
STRENGTH	$NULL$	double	1	factor by which to multiply field
FSE	$NULL$	double	0.0	Fractional Strength Error.
BXFACTOR	$NULL$	double	1	factor by which to multiply x component of field.
BYFACTOR	$NULL$	double	1	factor by which to multiply y component of field.
BZFACTOR	$NULL$	double	1	factor by which to multiply z component of field.
BXINSIDE	$NULL$	double	0.0	Constant Bx field to add inside the hard-edge boundaries.
BYINSIDE	$NULL$	double	0.0	Constant By field to add inside the hard-edge boundaries.
BZINSIDE	$NULL$	double	0.0	Constant Bz field to add inside the hard-edge boundaries.
BINSIDE_XMIN	$NULL$	double	0.0	Minimum x value at which BInside is applied.
BINSIDE_XMAX	$NULL$	double	0.0	Maximum x value at which BInside is applied.
ACCURACY	$NULL$	double	0.0	integration accuracy
METHOD	$NULL$	STRING	NULL	integration method (runge-kutta, bulirsch-stoer, modified-midpoint, two-pass modified-midpoint, leap-frog, non-adaptive runge-kutta

BMXYZ continued

A map of (Bx, By, Bz) vs (x, y, z), for straight elements only

Parameter Name	Units	Type	Default	Description
FILENAME	*NULL*	STRING	NULL	name of file containing columns (x, y, z) and either (Bx, By, Bz) or (Fx, Fy, Fz)
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
CHECK_FIELDS		short	0	check fields by computing divB and curlB errors?
INJECT_AT_Z0		short	0	By default, particles are placed at the entrance to the field map regardless of the z coordinate values. If nonzero, particles start at z=0.
DRIFT_MATRIX		short	0	If non-zero, instead of tracking to determine the matrix, just assume a drift-space matrix.
XY_INTERPOLATION_ORDER		short	1	Order of interpolation in x and y.
XY_GRID_EXCESS		short	0	Number of rows or columns to add in each dimension to the minimum.
SINGLE_PRECISION		short	0	If nonzero, store field data in single precision to reduce memory requirements.
DISCARD_MAP		short	0	If nonzero, field data is discarded after use. Reduces memory use but may result in additional delays for reading data again if needed.
VERBOSITY		short	0	Larger values result in more informational printouts. Will reduce performance.
PARTICLE_OUTPUT_FILE	*NULL*	STRING	NULL	name of file for phase-space output inside element. Use for debugging only in serial version.

A map of (Bx, By, Bz) vs (x, y, z), for straight elements only

Parameter Name	Units	Type	Default	Description
APCONTOUR	*NULL*	STRING	NULL	name of element defining aperture contour inside the field map region.
ZMIN_APCONTOUR	*NULL*	double	-8.98846567431158e+307	Minimum z value at which APCONTOUR apertures are applied.
ZMAX_APCONTOUR	*NULL*	double	8.98846567431158e+307	Maximum z value at which APCONTOUR apertures are applied.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates transport through a 3D magnetic field specified as a field map. It does this by simply integrating the Lorentz force equation in cartesian coordinates. It does not incorporate changes in the design trajectory resulting from the fields. I.e., if you input a dipole field, it is interpreted as a steering element.

The field map file is an SDDS file with the following columns:

- **x**, **y**, **x** — Transverse coordinates in meters (units should be "m").
- **Fx**, **Fy**, **Fx** — Normalized field values (no units). The field is multiplied by the value of the STRENGTH parameter to convert it to a local bending radius. For example, an ideal quadrupole could be simulated by setting (Fx=y, Fy=x, Fz=0), in which case STRENGTH is the K1 quadrupole parameter.
- **Bx**, **By**, **Bz** — Field values in Tesla (units should be "T"). The field is still multiplied by the value of the STRENGTH parameter, which is dimensionless.

The field map file must contain a rectangular grid of points, equispaced (separately) in x, y, and z. There should be no missing values in the grid (this is not checked by `elegant`). In addition, the x values must vary fastest as the values are accessed in row order, then the y values. To ensure that this is the case, use the following command on the field file:

sddssort *fieldFile* -column=z,incr -column=y,incr -column=x,incr

This element is an alternative to `FTABLE` using a more conventional integration method.

The `BXFACTOR`, `BYFACTOR`, and `BZFACTOR` allow multiplying the indicated field components by the given factors. These scaling parameters may result in unphysical fields.

By default, the `BMXYZ` element should be supplied with the full 3D field map of the magnet. To allow saving memory and reducing the file to load data, partial magnetic field maps can be loaded as well, but the user must specify the symmetry of the magnet to ensure that the fields are modeled correctly in the full volume. This is done using three optional parameters in the input file, `xSymmetry`, `ySymmetry`, and `zSymmetry`. If present, these must have one of three `none` (default), `even`, and `odd`.

For example, a normal quadrupole magnet would have `xSymmetry=odd`, `ySymmetry=odd`, and `zSymmetry=even`. A left/right symmetric dipole or sextupole would have `xSymmetry=even`, `ySymmetry=odd`, `zSymmetry=even`. A normal octupole would have the same symmetry codes as a normal quadrupole. Note that when using these symmetries, the user is not required to limit the field map to, say, $x \geq 0$, $y \geq 0$, and $z \geq 0$, though doing so saves the most memory. If possible, it is recommended to provide the fields over $x \geq -2\Delta x$, $y \geq -2\Delta y$, and $z \geq 0$, so that the transverse interpolation as a few points on both sides of the origin. This will ensure better results near the origin.

Internally, `elegant` stores only the partial field map. When field values are required outside this region, the declared symmetries are used to map the coordinates to the covered region and change signs of the various field components if required.

Internal apertures may be specified using four methods, as used in elements like `KQUAD` and `CSBEND`. The methods include upstrea `MAXAMP` elements, upstream `APCONTOUR` elements with `STICKY=1`, s-dependent apertures defined via the `aperture_data` command, and global-coordinate system apertures defined via the `obstruction_data` command.

For `APCONTOUR`-defined apertures, this can be invoked using the `STICKY=1` parameter, which is used to impose the aperture contour on downstream elements. One fine point is that the field map is typically significantly longer than the magnet itself. We don't want to apply the apertures except in some region in the interior of the field map This can be specified using the `ZMIN_APCONTOUR` and `ZMAX_APCONTOUR` parameters, which give the range of application in the coordinate system of the field map. In this case, to prevent application of the `APCONTOUR` apertures at the point of definition, the `APCONTOUR` element should have `HOLDOFF=1` and be inserted in the lattice just before the `BMXYZ` element.

10.7 BOFFAXE—A straight magnetic field element using off-axis expansion from an on-axis derivative.

A straight magnetic field element using off-axis expansion from an on-axis derivative.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	insertion length
LFIELD	M	double	-1	expected length of the field map for verification purposes only.
FILENAME	$NULL$	STRING	NULL	name of file containing derivative data
Z_COLUMN	$NULL$	STRING	z	name of longitunidal coordinate column in the data file
FIELD_COLUMN	$NULL$	STRING	NULL	name of derivative column in the data file
ORDER		short	1	order of transverse derivative
EXPANSION_ORDER		short	0	order of expansion in x and y. If zero, determined by data in file.
STRENGTH	$NULL$	double	1	factor by which to multiply field
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
BX	T	double	0.0	add BX*STRENGTH to Bx field
BY	T	double	0.0	add BY*STRENGTH to By field
Z_INTERVAL		short	1	input z data is sampled at this interval
Z_SUBDIVISIONS		short	1	Number of subdivisions of z interval to use in integration
SYNCH_RAD		short	0	if nonzero, include classical, single-particle synchrotron radiation
ISR		short	0	if nonzero, include incoherent synchrotron radiation (quantum excitation)
PARTICLE_OUTPUT_FILE		STRING	NULL	name of file for phase-space and field output. Use for debugging only in serial version.

A straight magnetic field element using off-axis expansion from an on-axis derivative.

Parameter Name	Units	Type	Default	Description
FIELD_OUTPUT_FILE	*NULL*	STRING	NULL	name of file to which a grid of field values will be written. Disabled for parallel version.
NX_OUTPUT	*NULL*	long	0	Number of x sample points for field output
NY_OUTPUT	*NULL*	long	0	Number of y sample points for field output
X_HALF_SPAN_OUTPUT	*NULL*	double	-1	Half-span of x sample points for field output
Y_HALF_SPAN_OUTPUT	*NULL*	double	-1	Half-span of y sample points for field output
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This experimental element simulates transport through a 3D magnetic field constructed from an off-axis expansion. At present, it is restricted to non-bending elements and in fact to quadrupoles and sextupoles.

This method of expanding the fields is prone to corruption by noise, to a much greater degree than the generalized gradient expansion used by BGGEXP. However, it uses data that can very readily be obtained from magnetic measurements with a Hall probe. Users are cautioned to take care in deciding how far to trust the expansion.

For quadrupoles, we use the on-axis gradient $g(z)$ and its z derivatives $g^{(n)}(z)$ The scalar potential can be written

$$\Phi = \frac{x^5y^5g^{(8)}(z)}{86400} - \frac{g^{(6)}(z)\left(x^5y^3+x^3y^5\right)}{4320} + \frac{1}{720}g^{(4)}(z)\left(x^5y+xy^5\right) + \frac{1}{108}x^3y^3g^{(4)}(z) - \frac{1}{12}\left(x^3y+xy^3\right)g''(z) + xyg(z) \tag{32}$$

From which we find

$$B_x = \frac{x^4y^5g^{(8)}(z)}{17280} - \frac{g^{(6)}(z)\left(5x^4y^3+3x^2y^5\right)}{4320} + \frac{1}{720}g^{(4)}(z)\left(5x^4y+y^5\right) + \frac{1}{36}x^2y^3g^{(4)}(z) - \frac{1}{12}\left(3x^2y+y^3\right)g''(z) + yg(z) \tag{33}$$

$$B_y = \frac{x^5y^4g^{(8)}(z)}{17280} - \frac{g^{(6)}(z)\left(3x^5y^2+5x^3y^4\right)}{4320} + \frac{1}{720}g^{(4)}(z)\left(x^5+5xy^4\right) + \frac{1}{36}x^3y^2g^{(4)}(z) - \frac{1}{12}\left(x^3+3xy^2\right)g''(z) + xg(z) \tag{34}$$

and

$$B_z = \frac{x^5y^5g^{(9)}(z)}{86400} - \frac{g^{(7)}(z)\left(x^5y^3+x^3y^5\right)}{4320} + \frac{1}{720}g^{(5)}(z)\left(x^5y+xy^5\right) + \frac{1}{108}x^3y^3g^{(5)}(z) - \frac{1}{12}g^{(3)}(z)\left(x^3y+xy^3\right) + xyg'(z) \tag{35}$$

These equations satisfy Maxwell's curl equation exactly while satisfying the divergence equation to 10^{th} order. A similar expansion is available in the code for sextupoles.

For quadrupoles, at minimum the z-dependent gradient $B_1(z)$ must be given, while for sextupoles $B_2(z)$ is required. $B_n(z)$ is specified in the column named by the `FIELD_COLUMN` parameter. The names for the columns containing z derivatives of $B_n(z)$ are constructed from the name of the primary column. Assume for concreteness that `FIELD_COLUMN="Gradient"`. `elegant` looks for $B_n^{(1)}(z)$ in column `GradientDeriv` and $B_n^{(m)}(z)$ for $m > 1$ in columns `GradientDeriv2`, `GradientDeriv3`, etc. Even if the expansion is limited by the `ORDER` parameter, all gradients will be used for interpolation with respect to z if the `Z_SUBDIVISIONS` parameter is larger than 1. The expansion is truncated if the needed columns do not exist in the input file.

The needed derivatives can be obtained using the program `sddsderiv`, e.g.,

```
sddsderiv gradient.sdds gradient1.sdds -differ=Gradient -versus=z -savitzky=7,7,7,1
sddsderiv gradient.sdds gradient2.sdds -differ=Gradient -versus=z -savitzky=7,7,7,2
sddsderiv gradient.sdds gradient3.sdds -differ=Gradient -versus=z -savitzky=7,7,7,3
sddsxref gradient.sdds gradient[123].sdds gradients.sdds -take=*Deriv*
```

(In this example, we use a Savitzky-Golay filter to compute the first three z derivatives of $g(z)$ using a 7^{th} order fit with 7 points ahead of and behind the evaluation location.) The file `gradients.sdds` would then be given as the value of `FILENAME`.

High-order numerical derivative are of course prone to corruption by measurement noise. Examining the derivatives is strongly recommended to ensure this is not an issue.

10.8 BRANCH—Conditional branch instruction to jump to another part of the beamline

Conditional branch instruction to jump to another part of the beamline
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
COUNTER		long	0	Counter, which is decremented by 1 for each pass. Set to negative value for unconditional branch.
INTERVAL		long	0	Interval between branching. If non-positive, use COUNTER-based method instead.
OFFSET		long	0	If INTERVAL method used, offset of branch passes.
VERBOSITY		long	0	Larger values result in more output during running.
DEFAULT_TO_ELSE		long	0	If non-zero, defaults to ELSE_TO when performing tracking for closed orbit, twiss_output, etc.
BRANCH_TO		STRING	NULL	Optional name of element to which to jump when counter is non-positive.
ELSE_TO		STRING	NULL	Optional name of element to which to jump when counter is positive.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is experimental and should be used with care. It may not work well with other features, e.g., orbit correction or twiss parameter output. It should work well with tracking.

Use of the `BRANCH` element to change the starting point in the lattice is not ideal. It is better to use the `change_start` command.

The element permits switching tracking between two segments of a beamline. This can be done once per run or periodically. For the former, the `COUNTER` parameter should be used to specify the pass number (which is zero on the first pass) on which to branch. For the latter, the `INTERVAL` i and (optionally) `OFFSET` o parameters should be used; the branch will occur when $(p-o)\%i == 0$.

The application that inspired creation of this element is to switch from tracking using lumped elements to tracking using element-by-element methods. More specifically, imagine we want to track for 10,000 turns to reach an equilibrium, then perform a beam abort. The equilibrium state can be accurately and rapidly modeled using lumped elements, such as `ILMATRIX` and `SREFFECTS`, but the beam abort needs to be modeled using comparatively slow element-by-element tracking.

```
RING1: ILMATRIX,...
SR1: SREFFECTS,...
...
RINGFULL: line=(SECTOR1, SECTOR2, ..., SECTOR40)
M1: MARK
M2: MARK
RF: RFCA,...
BR1: BRANCH,COUNTER=10000,BRANCH_TO="M1"
BR2: BRANCH,COUNTER=-1,BRANCH_TO="M2"
BL: line=(BR1,RING1,SR1,M1,RINGFULL,M2,RF)
```

Another application is to model a periodic bypass, e.g.,

```
RINGA: line=(...)
RINGB: line=(...)
RINGC: line=(...)
BYPASS: line=(...)
M1: MARK
M2: MARK
BR1: BRANCH,INTERVAL=100,BRANCH_TO="M1",ELSE_TO="M2"
BR2: BRANCH,COUNTER=-1,BRANCH_TO="M3"
BL: line=(RINGA,BR1,M1,BYPASS,BR2,M2,RINGB,M3,RINGC)
```

In this example, the full ring is composed of three sections, `RINGA`, `RINGB`, and `RINGC`. Every 100 passes, the `RINGB` portion is bypassed in favor of `BYPASS`.

10.9 BRAT—Bending magnet RAy Tracing using (Bx, By, Bz) vs (x, y, z).

Bending magnet RAy Tracing using (Bx, By, Bz) vs (x, y, z).
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
ANGLE	*RAD*	double	0.0	Nominal bending angle. Will be refined to match geometry specified by input/output and vertex coordinates
FSE	*NULL*	double	0.0	fractional strength error
ACCURACY	*NULL*	double	0.0	integration accuracy
METHOD	*NULL*	STRING	NULL	Ignored. Method defaults to Bulirsch-Stoer.
FILENAME	*NULL*	STRING	NULL	name of file containing columns (x, y, z, Bx, By, Bz) for main field
ADDITIONAL_FILENAME	*NULL*	STRING	NULL	name of a file containing columns (x, y, z, Bx, By, Bz) for adding to the main field
XVERTEX	*M*	double	0.0	x coordinate of vertex point
ZVERTEX	*M*	double	0.0	z coordinate of vertex point
XENTRY	*M*	double	0.0	x coordinate of nominal entry point
ZENTRY	*M*	double	0.0	z coordinate of nominal entry point
XEXIT	*M*	double	0.0	x coordinate of nominal exit point
ZEXIT	*M*	double	0.0	z coordinate of nominal exit point
DXMAP	*M*	double	0.0	x displacement of map
DYMAP	*M*	double	0.0	y displacement of map
DZMAP	*M*	double	0.0	z displacement of map
YAWMAP	*RAD*	double	0.0	yaw of map about x=z=0
MAIN_FACTOR		double	1	factor by which to multiply main fields
ADDITIONAL_FACTOR		double	0.0	factor by which to multiply additional fields prior to adding to main fields

BRAT continued

Bending magnet RAy Tracing using (Bx, By, Bz) vs (x, y, z).

Parameter Name	Units	Type	Default	Description
FACTOR		double	1	factor by which to multiply combined fields
DELTA_BY_INSIDE	T	double	0.0	Vertical magnetic field to add to the map value when inside the hard-edge boundaries
USE_FTABLE		short	0	If nonzero, use FTABLE method for integration. Value gives the number of kicks.
XY_INTERPOLATION_ORDER		short	1	Order of interpolation in x and y.
XY_GRID_EXCESS		short	0	Number of rows or columns to add in each dimension to the minimum.
XY_EXTRAPOLATE		short	0	If nonzero, will extrapolate the field map in (x,y) if particle is outside. Otherwise, field is assumed to be zero.
USE_SBEN_MATRIX		short	0	If nonzero, instead of using tracking to determine the matrix, will just use a sector-bend matrix.
USE_DRIFT_MATRIX		short	0	If nonzero, instead of using tracking to determine the matrix, will just use a drift matrix.
SINGLE_PRECISION		short	0	If nonzero, store field data in single precision to reduce memory requirements. Incompatible with FTABLE mode.
PARTICLE_OUTPUT_FILE	*NULL*	STRING	NULL	Filename template for particle output. Can be very resource intensive!
PARTICLE_OUTPUT_LOST_ONLY	*NULL*	short	0	If non-zero, particle output includes only lost particles.

Bending magnet RAy Tracing using (Bx, By, Bz) vs (x, y, z).

Parameter Name	Units	Type	Default	Description
PARTICLE_OUTPUT_SELECTION_INTERVAL	*NULL*	long	1	Interval between particles selected for output.
PARTICLE_OUTPUT_SAMPLE_INTERVAL	*NULL*	long	1	Interval in integration steps for particle output.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

Bending magnet RAy Tracing using (Bx, By, Bz) vs (x, y, z). This element is a companion to the commandline program `abrat`. It integrates through a 3-D field map for a bending magnet, including coordinate transformations. No synchrotron radiation calculations are included at this time.

Coordinates

The coordinates of the field map are right-handed system (x, y, z), where z is along the length of the magnet, x is to the right as viewed along the direction of beam propagation, and y is up. The user must specify the (x, z) coordinates of three points:

- Nominal entrance point: `XENTRY` and `ZENTRY`. These give the coordinates of reference trajectory at the exit of the previous element. In the limit of a hard-edge model, this would be at the entrance to the magnetic field region.
- Vertex point: `XVERTEX` and `ZVERTEX`. These give the coordinates of vertex point, which is the intersection of the reference lines from the entrance and exit.
- Nominal exit point: `XEXIT` and `ZEXIT`. These give the coordinates of reference trajectory at the exit of the previous element. In the limit of a hard-edge model, this would be at the exit from the magnetic field region.

The bending angle is equal to the angle between two lines: the line from `ENTRY` to `VERTEX` and the line from `VERTEX` to `EXIT`. The `L` and `ANGLE` parameters supplied by the user are used for geometry calculations (e.g., floor coordinates) only.

The `DXMAP`, `DZMAP`, `YAWMAP`, and `FSE` values can be used to optimize the field map to ensure that the horizontal reference trajectory is not displaced at the exit of the element. The optimization feature of the `abrat` program can be used to determine these values.

Matrix generation

`elegant` will use tracking to determine the transport matrix for `BRAT` elements, which is needed for computation of twiss parameters and other operations. This can require some time, so `elegant` will cache the matrices and re-use them for identical elements.

If matrices are not of particular interest, significant time savings can be realized by setting `USE_SBEND_MATRIX=1`. Of course, any matrix-based results (e.g., twiss parameters) are then dubious at best.

Symmetry

By default, the `BRAT` element should be supplied with the full 3D field map of the magnet. To allow saving memory and reducing the time to load data, partial magnetic field maps can be loaded as well, but the user must specify the symmetry of the magnet to ensure that the fields are modeled correctly in the full volume. This is done using the `ySymmetry` and `zSymmetry` parameters in the input file, which may have one of three values: `none` (default), `even`, and `odd`. A normal (upright) multipole magnet would have `ySymmetry=odd`, while a skew multipole would have `ySymmetry=even`.

Note that when using these symmetries, the user is not required to limit the field map to $y \geq 0$, though doing so saves the most memory. If possible, it is recommended to provide the fields over $y \geq -2\Delta y$, so that the transverse interpolation as a few points on both sides of $y = 0$. This will ensure better results near the origin.

Integration methods

The original (and default) integration method is Bulirsch-Stoer integration of the Lorentz force equation. As an alternative, one can use the faster, rotation-based method of the `FTABLE` element. For repeated use, the two methods should be compared and a choice made based on the user's needs.

10.10 BUMPER—A time-dependent kicker magnet with optional spatial dependence of the kick and no fringe effects. The waveform is in SDDS format, with time in seconds and amplitude normalized to 1. The optional spatial dependence is also specified as an SDDS file.

A time-dependent kicker magnet with optional spatial dependence of the kick and no fringe effects. The waveform is in SDDS format, with time in seconds and amplitude normalized to 1. The optional spatial dependence is also specified as an SDDS file.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
ANGLE	RAD	double	0.0	kick angle
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
B2	$1/M^2$	double	0.0	Sextupole term: By=Bo*(1+b2*x2̂)
TIME_OFFSET	S	double	0.0	time offset of waveform
PERIODIC		long	0	is waveform periodic?
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
FIRE_ON_PASS		long	0	pass number to fire on
N_KICKS		long	0	Number of kicks to use for simulation. 0 uses an exact result but ignores b2.
WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving kick factor vs time
DEFLECTION_MAP		STRING	NULL	optional filename giving the spatial variation of the deflection
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a time-dependent kicker magnet as a rectangular dipole with no fringe field effects. To use this element, you must supply an SDDS file giving the time-dependent waveform. The element is called BUMPER to because HKICK, VKICK, KICKER are used for steering magnets.

The arrival time of the beam is taken to define the reference time, $t = 0$. Hence, if the waveform file has the maximum amplitude at $t = 0$, the beam will get kicked at the peak of the waveform. If the waveform peaks at $t = t_{peak}$, then setting TIME_OFFSET equal to $-t_{peak}$ will ensure that the beam is kicked at the peak amplitude.

By default, the kicker fires on the first beam passage. However, if FIRE_ON_PASS is used, then the kicker is treated like a drift space until the specified pass. Note that the first pass is 0, not 1.

If PHASE_REFERENCE is non-zero, then the initial timing is taken from the first time-dependent element that has the same PHASE_REFERENCE value. This would allow, for example, simulating several kickers firing at the same time. Delays relative to this reference time can then be given with positive adjustments to TIME_OFFSET.

The waveform input file need not have equispaced points in time. However, the time values should increase monotonically.

The deflection map, if provided, should have four floating-point columns

1. Transverse coordinates x and y, with units of m.

2. Kick multipliers xpFactor and ypFactor, which are dimensionless quantities.

The resulting kick in each plane for a particle with coordinates (x, y, t, δ) is

$$\Delta q'(x, y, t, \delta) = \frac{\theta A(t - t_{offset}) f_q(x, y)}{1 + \delta}, \tag{36}$$

where q stands for x or y, θ is the specfied deflection angle, $A(t)$ is the time-dependent amplitude waveform, and $f_q(x, y)$ is the deflection map factor for the q plane at the particle's location.

The data in the deflection map file must be sorted so that x changes fastest, which can be accomplished using the command

```
sddssort input.sdds -column=y,incr -column=x,incr
```

This element simulates a dipole kicker only. For multipole kickers, see the MBUMPER element.

Explanation of <filename>=<x>+<y> format: Several elements in elegant make use of data from external files to provide input waveforms. The external files are SDDS files, which may have many columns. In order to provide a convenient way to specify both the filename and the columns to use, we frequently employ <filename>=<x>+<y> format for the parameter value. For example, if the parameter value is waveform.sdds=t+A, then it means that columns t and A will be taken from file waveform.sdds. The first column is always the independent variable (e.g., time, position, or frequency), while the second column is the dependent quantity.

10.11 CCBEND—A canonically-integrated straight dipole magnet, assumed to have multipoles defined in Cartesian coordinates.

A canonically-integrated straight dipole magnet, assumed to have multipoles defined in Cartesian coordinates.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	arc length (not chord length!)
ANGLE	RAD	double	0.0	bend angle
K1	$1/M^2$	double	0.0	geometric quadrupole strength
K2	$1/M^3$	double	0.0	geometric sextupole strength
K3	$1/M^4$	double	0.0	geometric octupole strength
K4	$1/M^5$	double	0.0	geometric decapole strength
K5	$1/M^6$	double	0.0	geometric 12-pole strength
K6	$1/M^7$	double	0.0	geometric 14-pole strength
K7	$1/M^8$	double	0.0	geometric 16-pole strength
K8	$1/M^9$	double	0.0	geometric 18-pole strength
TILT	RAD	double	0.0	rotation about incoming longitudinal axis
YAW	RAD	double	0.0	rotation about vertical axis through entrance point
FRINGEMODEL		long	0	fringe model to use
HGAP	M	double	0.0	half-gap between poles
FINT1		double	0.0	edge integral for entrance
FINT2		double	0.0	edge integral for exit
FRINGE1K0		double	0.0	Lindberg's K0 edge integral for entrance
FRINGE1I0		double	0.0	Lindberg's I0 edge integral for entrance
FRINGE1K2		double	0.0	Lindberg's K2 edge integral for entrance
FRINGE1I1		double	0.0	Lindberg's I1 edge integral for entrance
FRINGE1K4		double	0.0	Lindberg's K4 edge integral for entrance
FRINGE1K5		double	0.0	Lindberg's K5 edge integral for entrance
FRINGE1K6		double	0.0	Lindberg's K6 edge integral for entrance
FRINGE1K7		double	0.0	Lindberg's K7 edge integral for entrance

A canonically-integrated straight dipole magnet, assumed to have multipoles defined in Cartesian coordinates.

Parameter Name	Units	Type	Default	Description
FRINGE2K0		double	0.0	Lindberg's K0 edge integral for entrance
FRINGE2I0		double	0.0	Lindberg's I0 edge integral for exit
FRINGE2K2		double	0.0	Lindberg's K2 edge integral for exit
FRINGE2I1		double	0.0	Lindberg's I1 edge integral for exit
FRINGE2K4		double	0.0	Lindberg's K4 edge integral for exit
FRINGE2K5		double	0.0	Lindberg's K5 edge integral for exit
FRINGE2K6		double	0.0	Lindberg's K6 edge integral for exit
FRINGE2K7		double	0.0	Lindberg's K7 edge integral for exit
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
DZ	*M*	double	0.0	misalignment
ETILT	*RAD*	double	0.0	misalignment rotation about longitudinal axis
EPITCH	*RAD*	double	0.0	misalignment rotation about vertical axis. Ignored if MALIGN_METHOD=0
EYAW	*RAD*	double	0.0	misalignment rotation about horizontal axis. Ignored if MALIGN_METHOD=0
MALIGN_METHOD		short	0	0=original, 1=new entrace-centered, 2=new body-centered
FSE		double	0.0	fractional strength error
FSE_DIPOLE		double	0.0	fractional strength error of dipole component
FSE_QUADRUPOLE		double	0.0	fractional strength error of quadrupole component
XKICK	*RAD*	double	0.0	horizontal steering angle (approximate)

A canonically-integrated straight dipole magnet, assumed to have multipoles defined in Cartesian coordinates.

Parameter Name	Units	Type	Default	Description
N_SLICES		long	4	Number of slices (full integrator steps).
N_KICKS		long	4	number of kicks. Deprecated. Use N_SLICES.
INTEGRATION_ORDER		short	4	integration order (2, 4, or 6)
SYSTEMATIC_MULTIPOLES		STRING	NULL	input file for systematic multipoles
EDGE_MULTIPOLES		STRING	NULL	input file for systematic entrance/exit edge multipoles
EDGE1_MULTIPOLES		STRING	NULL	input file for systematic entrance edge multipoles. Overrides EDGE_MULTIPOLES.
EDGE2_MULTIPOLES		STRING	NULL	input file for systematic exit edge multipoles. Overrides EDGE_MULTIPOLES.
RANDOM_MULTIPOLES		STRING	NULL	input file for random multipoles
SYSTEMATIC_MULTIPOLE_FACTOR		double	1	Factor by which to multiply systematic and edge multipoles
RANDOM_MULTIPOLE_FACTOR		double	1	Factor by which to multiply random multipoles
REFERENCE_ORDER		short	0	Reference order for multipole errors. Overridden by value in multipole files, if those are given.
MIN_NORMAL_ORDER		short	-1	If nonnegative, minimum order of systematic and random normal multipoles to use from data files.
MIN_SKEW_ORDER		short	-1	If nonnegative, minimum order of systematic and random skew multipoles to use from data files.
MAX_NORMAL_ORDER		short	-1	If nonnegative, maximum order of systematic and random normal multipoles to use from data files.

A canonically-integrated straight dipole magnet, assumed to have multipoles defined in Cartesian coordinates.

Parameter Name	Units	Type	Default	Description
MAX_SKEW_ORDER		short	-1	If nonnegative, maximum order of systematic and random skew multipoles to use from data files.
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?
ISR1PART		short	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
USE_RAD_DIST		short	0	If nonzero, overrides SYNCH_RAD and ISR, causing simulation of radiation from distributions, optionally including opening angle.
ADD_OPENING_ANGLE		short	1	If nonzero, radiation opening angle effects are added if USE_RAD_DIST is nonzero.
SR_IN_ORDINARY_MATRIX		short	0	If nonzero, the (tracking-based) matrix used for routine computations includes classical synchrotron radiation if SYNCH_RAD=1.
OPTIMIZE_FSE		short	1	Optimize strength (FSE) to obtain the ideal deflection angle.
OPTIMIZE_DX		short	1	Optimize x offset to obtain centered trajectory.
OPTIMIZE_FSE_ONCE		short	0	If nonzero, the FSE offset is optimized only once, even if relevant parameters are changed.
OPTIMIZE_DX_ONCE		short	0	If nonzero, the x offset is optimized only once, even if relevant parameters are changed.

A canonically-integrated straight dipole magnet, assumed to have multipoles defined in Cartesian coordinates.

Parameter Name	Units	Type	Default	Description
COMPENSATE_KN		short	0	If nonzero, K1 and K2 strengths are adjusted to compensate for the changes in FSE needed to center the trajectory.
REFERENCE_CORRECTION		short	1	1: correct pathlength, 2: correct trajectory, 3: correct both.
EDGE_ORDER		short	3	Gives order of edge effects. Does not affect edge multipoles.
DX_DY_SIGN		short	1	Prior to 2020.4, the sign of DX and DY was reversed for ANGLE<0. For backward compatibility, this is retained. Set this field to a positive value to use a consistent convention.
VERBOSE		short	0	If nonzero, print messages showing optimized FSE and x offset.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element provides a symplectic straight-pole, bending magnet with the exact Hamiltonian in Cartesian coordinates [61]. The quadrupole, sextupole, and other multipole terms are defined in Cartesian coordinates. The magnet at present is restricted to having rectangular ends. This is quite different from `CSBEND`, where the edge angles are user-defined and where the field expansion is in curvilinear coordinates. Strictly speaking, `CSBEND` is only valid when the dipole is built with curved, beam-following poles.

Integration of particles in `CCBEND` is very similar to what's done for `KQUAD`, `KSEXT`, and `KOCT`. The only real difference is that coordinate transformations are performed at the entrance and exit to orient the incoming central trajectory to the straight magnet axis. In addition, the fractional strength error is adjusted to ensure that the outgoing central trajectory is correct.

By default, two adjustments are made at start-up and whenever the length, angle, gradient, or sextupole term change:

1. The fractional strength error is altered to ensure the correct deflecting angle. This is required because the bending field varies along the trajectory. By default, this affects all field compo-

nents together, per the usual convention in elegant. To restrict the strength change to the dipole term, set COMPENSATE_KN=1. To turn off this optimization, set OPTIMIZE_FSE=0.

2. The transverse position is adjusted to center the trajectory in the magnet. If the sagitta is σ and ANGLE is positive, the initial and final x coordinates are $x = -\sigma/2$, while the center coordinate is $x = \sigma/2$. To turn off this optimization, set OPTIMIZE_DX=0.

One can block the re-optimization of these parameters by setting OPTIMIZE_FSE_ONCE and OPTIMIZE_DX_ONCE to 1. Note also that the optimization is performed with all error-defining parameters (DX, DY, DZ, FSE, ETILT, etc.) set to zero. However, errors that are assigned to, say, the K1 value directly would not be recognized as such. For this reason, assigning errors to K1 is not recommended; instead, use the FSE_QUADRUPOLE parameter.

Having computed the ideal trajectory through a CCBEND element, elegant can suppress any errors in the trajectory. Such errors may occur due to limited accuracy in numerical integration. It is recommended to set REFERENCE_CORRECTION=1 to ensure that the path length is corrected. Optionally, setting REFERENCE_CORRECTION=2 would instead correct residual transverse trajectory errors. Using REFERENCE_CORRECTION=3 corrects both types of error.

Edge angles and edge effects

The user may specify edge multipoles using the EDGE_MULTIPOLE parameter. In addition, the CCBEND element supports two fringe models, selected via the FRINGEMODEL parameter, which may have a value of 0 (default) or 1.

0 — The default edge angle treatment in CCBEND is relatively simple, consisting of a vertical focusing effect with momentum dependence to all orders. Also included are edge pseudo-sextupoles (due to the body K_1 term) and pseudo-octupoles (due to the body K_2 term).

1 — This model is based on theoretical work and code by R. Lindberg, and includes soft-fringe effects via a series of fringe integrals. The integrals can be computed with the companion program straightDipoleFringeCalc from a generalized gradient expansion (GGE). The GGE can be created using either computeCBGGE (for cylindrical-boundary data) or computeRBGGE for (rectangular-boundary data). There is an example in the elegant examples collection.

Multipole errors

Multipole errors are specified for the body and edge in the same fashion as for the KQUAD element. The reference is the dipole field by default, but this may be changed using the REFERENCE_ORDER parameter.

Radiation effects

If SYNCH_RAD is non-zero, classical synchrotron radiation is included in tracking. Incoherent synchrotron radiation, when requested with ISR=1, normally uses gaussian distributions for the excitation of the electrons. (To exclude ISR for single-particle tracking, set ISR1PART=0.) Setting USE_RAD_DIST=1 invokes a more sophisticated algorithm that uses correct statistics for the photon energy and number distributions. In addition, if USE_RAD_DIST=1 one may also set ADD_OPENING_ANGLE=1, which includes the photon angular distribution when computing the effect on the emitting electron.

If SYNCH_RAD and SR_IN_ORDINARY_MATRIX are non-zero, classical synchrotron radiation will be included in the ordinary matrix (e.g., for twiss_output and matrix_output). Symplecticity is not assured, but the results may be interesting nonetheless. A more rigorous approach is to use moments_output. SR_IN_ORDINARY_MATRIX does not affect tracking.

Adding errors

When adding errors, care should be taken to choose the right parameters. The FSE, FSE_DIPOLE, FSE_QUADRUPOLE, ETILT, EPITCH, and YAW, DX, DY, and DZ parameters are used for assigning errors to the strength and alignment relative to the ideal values given by ANGLE and TILT. One can also assign errors to ANGLE and TILT, but this has a different meaning: in this case, one is assigning errors to the survey itself. The reference beam path changes, so there is no orbit/trajectory error. Note that when adding errors to FSE, the error is assumed to come from the power supply, which means that multipole strengths also change.

Assigning errors to K1 is also possible, but is not the best approach, since it changes the internal reference trajectory calculation for the element.

Splitting dipoles

The CCBEND element does not support splitting. **Important**: Users *should not* attempt to split CCBEND elements by hand, since this will not result in the correct geometry entering and exiting the various parts.

Matrix generation

elegant will use tracking to determine the transport matrix for CCBEND elements, which is needed for computation of twiss parameters and other operations. This can require some time, so elegant will cache the matrices and re-use them for identical elements.

10.12 CENTER—An element that centers the beam transversely on the ideal trajectory.

An element that centers the beam transversely on the ideal trajectory.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
X		long	1	center x coordinates?
XP		long	1	center x' coordinates?
Y		long	1	center y coordinates?
YP		long	1	center y' coordinates?
S		long	0	center s coordinates?
DELTA		long	0	center delta coordinates?
T		long	0	center t coordinates?
ONCE_ONLY		long	0	compute centering offsets for first beam only, apply to all?
ON_PASS		long	-1	If nonnegative, do centering on the nth pass only.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.13 CEPL—A numerically-integrated linearly-ramped electric field deflector.

A numerically-integrated linearly-ramped electric field deflector.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
RAMP_TIME	S	double	1e-09	time to ramp to full strenth
TIME_OFFSET	S	double	0.0	offset of ramp-start time
VOLTAGE	V	double	0.0	maximum voltage between plates due to ramp
GAP	M	double	0.01	gap between plates
STATIC_VOLTAGE	V	double	0.0	static component of voltage
TILT	RAD	double	0.0	rotation about longitudinal axis
ACCURACY		double	0.0001	integration accuracy
X_MAX	M	double	0.0	x half-aperture
Y_MAX	M	double	0.0	y half-aperture
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
N_STEPS		long	100	number of steps (for nonadaptive integration)
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, non-adaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.14 CHARGE—An element to establish the total charge of a beam. Active on first pass only. If given, overrides all charge specifications on other elements.

An element to establish the total charge of a beam. Active on first pass only. If given, overrides all charge specifications on other elements.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
TOTAL	C	double	0.0	total charge in beam
PER_PARTICLE	C	double	0.0	charge per macroparticle
ALLOW_TOTAL_CHANGE	$NULL$	long	0	If nonzero, allow total charge to change while tracking even if number of particles does not change. Useful for ramping of charge.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This is the preferred way to assign charge to a beam, which is needed for the use of CSR simulation (CSRCSBEND, CSRDRIFT), wake simulation (WAKE, TRWAKE, LRWAKE, ZLONGIT, ZTRANSVERSE), rf mode simulation (RFMODE, TRFMODE, FRFMODE, RTRFMODE), space charge simulation (LSCDRIFT, RFCW, SCMULT), and intrabeam scattering simulation (IBSCATTER).

10.15 CKICKER—Optical stochastic cooling kicker element—applies a kick in particle momentum

Optical stochastic cooling kicker element—applies a kick in particle momentum
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
ID		STRING	NULL	System identifier
STRENGTH		double	0.0	Strength factor
KICK_LIMIT		double	0.0	Limit on applied kick, nominally in radians.
PHASE	$1/(2*Pi)$	double	0.0	Phase of the applied voltage relative to the bunch center, with 0 being on-crest.
UPDATE_INTERVAL		long	0	Interval in units of pickup update interval for sampling pickup data and upda
START_PASS		long	-1	If positive, first pass on which to drive beam.
END_PASS		long	-1	If positive, last pass on which to drive beam.
BUNCHED_BEAM_MODE		short	1	If non-zero, run in bunched beam mode.
LAMBDA_RAD	M	double	0.0	Wavelength of radiation in meters.
TRANSVERSE_MODE		short	1	If non-zero, use transverse effects.
INCOHERENT_MODE		short	1	If non-zero, include incoherent contributions due to individual particles.
NUMERICAL_MODE		short	0	If non-zero, evaluate kicks from a numerical wave packet determined once per turn, rather than calculating kicks on a particle-by-particle basis.
DT_CLOSED_ORBIT		short	0	If non-zero, take the CPICKUP->CKICKER dt from the closed orbit instead of the bunch average times.
ANGLE_RAD	$MRAD$	double	0.0	Angle, in mrad, of pickup radiation.
MAGNIFICATION		double	0.0	Magnification of pickup radiation in the kicker.

CKICKER continued

Optical stochastic cooling kicker element—applies a kick in particle momentum

Parameter Name	Units	Type	Default	Description
MODULATION_FREQ	$1/Pass$	double	0.0	Frequency of cos$\hat{2}$ modulation function to apply to the coherent kicks.
NU		long	0	Number of undulator periods.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.16 CLEAN—Cleans the beam by removing outlier particles.

Cleans the beam by removing outlier particles.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
MODE		STRING	stdeviation	stdeviation, absdeviation, or absvalue
XLIMIT		double	0.0	Limit for x
XPLIMIT		double	0.0	Limit for x'
YLIMIT		double	0.0	Limit for y
YPLIMIT		double	0.0	Limit for y'
TLIMIT		double	0.0	Limit for t
DELTALIMIT		double	0.0	Limit for (p-p0)/p0
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The three options and their mode of operation are as follows:

- `ABSDEV` : compute the mean of the coordinate values, then compute absolute value of difference between the mean and the coordinate value for each particle. If this absolute deviation exceeds the user-specified limit, then the particle is removed. This could be used, for example, to remove particles outside of 100ps of the mean arrival time.
- `STDEVIATION`: compute the mean and standard deviation of the coordinate values, then compute the absolute value of difference between the mean and the coordinate value for each particle, normalized by the standard deviation. If this value exceeds the user-specified limit, then the particle is removed. This could be used, for example, to remove particles outside of five sigma of the horizontal beam size from the centroid.
- `ABSVALUE`: compare the absolute value of the particle coordinate value to the user-specified limit. If it exceeds this limit, then the particle is removed. This could be used, for example, to remove particles with slopes that exceed 100 mrad.

10.17 CORGPIPE—A corrugated round pipe, commonly used as a dechirper in linacs.

A corrugated round pipe, commonly used as a dechirper in linacs.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
RADIUS	M	double	0.0	pipe radius
PERIOD	M	double	0.0	period of corrugations (<< radius recommended)
GAP	M	double	0.0	gap in corrugations (< period required)
DEPTH	M	double	0.0	depth of corrugations (<< radius, > period recommended)
DT	S	double	0.0	maximum time duration of wake (0 for autoscale)
TMAX	S	double	0.0	maximum time duration of wake (0 for autoscale)
N_BINS		long	0	number of bins for charge histogram (0 for autoscale)
INTERPOLATE		long	0	interpolate wake?
SMOOTHING		long	0	Use Savitzky-Golay filter to smooth current histogram?
SG_HALFWIDTH		long	4	Savitzky-Golay filter halfwidth for smoothing
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing
CHANGE_P0		long	0	change central momentum?
ALLOW_LONG_BEAM		long	0	allow beam longer than wake data?
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the wake to full strength.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element implements the longitudinal wake for a corrugated pipe using a model by K. Bane [38]. The method used is identical to that for the `WAKE` element. The only difference is that instead of providing a file to specify the wake, one specifies the parameters of Bane's model, as described above.

Setting the N_BINS and TMAX paramaters to 0 is recommended. This results in auto-scaling of the number of bins and the time spacing of the wake to ensure sufficient length to cover the beam and a sufficiently fine time step to resolve the oscillations in the wake.

As with WAKE, the default degree of smoothing (SG_HALFWIDTH=4) may be excessive. It is suggested that users vary this parameter to verify that results are reliable if smoothing is employed (SMOOTHING=1).

10.18 CORGPLATES—A pair of corrugated plates, commonly used as a dechirper in linacs.

A pair of corrugated plates, commonly used as a dechirper in linacs.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
HALFGAP	M	double	0.0	Half the distance between the top of the corrugations.
PERIOD	M	double	0.0	period of corrugations
DEPTH	M	double	0.0	depth of corrugations
DT	S	double	0.0	maximum time duration of wake (0 for autoscale)
TMAX	S	double	0.0	maximum time duration of wake (0 for autoscale)
INTERPOLATE		long	0	interpolate wake?
SMOOTHING		long	0	Use Savitzky-Golay filter to smooth current histogram?
SG_HALFWIDTH		long	4	Savitzky-Golay filter half-width for smoothing
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing
CHANGE_P0		long	0	change central momentum?
ALLOW_LONG_BEAM		long	0	allow beam longer than wake data?
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the wake to full strength.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element implements the longitudinal wake for a pair of parallel corrugated plates, using the model described in [63]. The method used is identical to that for the `WAKE` element. The only difference is that instead of providing a file to specify the wake, one specifies the parameters of the model.

Setting the `DT` and `TMAX` paramaters to 0 is recommended. This results in auto-scaling of the bin size, the number of bins, and the time spacing of the wake to ensure sufficient length to cover the beam and a sufficiently fine time step to resolve the oscillations in the wake.

As with `WAKE`, the default degree of smoothing (`SG_HALFWIDTH=4`) may be excessive. It is suggested that users vary this parameter to verify that results are reliable if smoothing is employed (`SMOOTHING=1`).

10.19 CPICKUP—Optical stochastic cooling pickup element—records particle position

Optical stochastic cooling pickup element—records particle position
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
ID		STRING	NULL	System identifier
UPDATE_INTERVAL		long	0	Interval in turns for sampling data and updating filter output.
START_PASS		long	-1	If positive, first pass on which to perform computations.
END_PASS		long	-1	If positive, last pass on which to perform computations.
DX	M	double	0.0	Horizontal offset (subtracted from pickup signal).
DY	M	double	0.0	Vertical offset (subtracted from pickup signal)
BUNCHED_BEAM_MODE		short	1	If non-zero, run in bunched beam mode.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.20 CSBEND—A canonical kick sector dipole magnet.

A canonical kick sector dipole magnet.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	arc length
ANGLE	RAD	double	0.0	bend angle
K1	$1/M^2$	double	0.0	geometric quadrupole strength
K2	$1/M^3$	double	0.0	geometric sextupole strength
K3	$1/M^4$	double	0.0	geometric octupole strength
K4	$1/M^5$	double	0.0	geometric decapole strength
K5	$1/M^6$	double	0.0	geometric 12-pole strength
K6	$1/M^7$	double	0.0	geometric 14-pole strength
K7	$1/M^8$	double	0.0	geometric 16-pole strength
K8	$1/M^9$	double	0.0	geometric 18-pole strength
E1	RAD	double	0.0	entrance edge angle
E2	RAD	double	0.0	exit edge angle
TILT	RAD	double	0.0	rotation about incoming longitudinal axis
H1	$1/M$	double	0.0	entrance pole-face curvature
H2	$1/M$	double	0.0	exit pole-face curvature
HGAP	M	double	0.0	half-gap between poles
FINT		double	0.5	edge-field integral
FINT1		double	-1	edge-field integral. If negative, use FINT.
FINT2		double	-1	edge-field integral. If negative, use FINT.
FRINGE1K0		double	0.0	Lindberg's K0 edge integral for entrance
FRINGE1K1		double	0.0	Lindberg's K1 edge integral for entrance
FRINGE1K2		double	0.0	Lindberg's K2 edge integral for entrance
FRINGE1K3		double	0.0	Lindberg's K3 edge integral for entrance
FRINGE1K4		double	0.0	Lindberg's K4 edge integral for entrance
FRINGE1K5		double	0.0	Lindberg's K5 edge integral for entrance

A canonical kick sector dipole magnet.

Parameter Name	Units	Type	Default	Description
FRINGE1K6		double	0.0	Lindberg's K6 edge integral for entrance
FRINGE2K0		double	0.0	Lindberg's K0 edge integral for entrance
FRINGE2K1		double	0.0	Lindberg's K1 edge integral for exit
FRINGE2K2		double	0.0	Lindberg's K2 edge integral for exit
FRINGE2K3		double	0.0	Lindberg's K3 edge integral for exit
FRINGE2K4		double	0.0	Lindberg's K4 edge integral for exit
FRINGE2K5		double	0.0	Lindberg's K5 edge integral for exit
FRINGE2K6		double	0.0	Lindberg's K6 edge integral for exit
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
XKICK	RAD	double	0.0	bend-plane steering angle (approximate)
YKICK	RAD	double	0.0	non-bend-plane steering angle (approximate)
FSE		double	0.0	fractional strength error of all components
FSE_DIPOLE		double	0.0	fractional strength error of dipole component
FSE_QUADRUPOLE		double	0.0	fractional strength error of quadrupole component
ETILT	RAD	double	0.0	error rotation about incoming longitudinal axis
EPITCH	RAD	double	0.0	error rotation about horizontal axis. Ignored if MALIGN_METHOD=0
EYAW	RAD	double	0.0	error rotation about vertical axis. Ignored if MALIGN_METHOD=0

A canonical kick sector dipole magnet.

Parameter Name	Units	Type	Default	Description
N_SLICES		long	4	Number of slices (full integrator steps).
N_KICKS		long	4	number of kicks. Deprecated. Use N_SLICES.
ETILT_SIGN		short	1	Sign of ETILT relative to TILT. -1 is the old convention prior to 2020.5
NONLINEAR		short	1	include nonlinear field components?
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
SR_IN_ORDINARY_MATRIX		short	0	If nonzero, the tracking-based matrix used for routine computations includes classical synchrotron radiation if SYNCH_RAD=1.
EDGE1_EFFECTS		short	1	If nonzero, determines the method used to include entrance edge effects.
EDGE2_EFFECTS		short	1	If nonzero, determines the method used to include exit edge effects.
EDGE_ORDER		short	1	order to which to include edge effects for both edges
INTEGRATION_ORDER		short	4	integration order (2, 4, or 6)
EXPAND_HAMILTONIAN		short	0	If 1, Hamiltonian is expanded to leading order.
EDGE1_KICK_LIMIT		double	-1	maximum kick entrance edge can deliver
EDGE2_KICK_LIMIT		double	-1	maximum kick exit edge can deliver
KICK_LIMIT_SCALING		short	0	scale maximum edge kick with FSE?
USE_BN		short	0	use b<n> instead of K<n>?

`CSBEND` continued

A canonical kick sector dipole magnet.

Parameter Name	Units	Type	Default	Description
EXPANSION_ORDER		short	`0`	Order of field expansion. (0=auto)
B1	$1/M$	double	0.0	K1 = b1/rho, where rho is bend radius
B2	$1/M^2$	double	0.0	K2 = b2/rho
B3	$1/M^3$	double	0.0	K3 = b3/rho
B4	$1/M^4$	double	0.0	K4 = b4/rho
B5	$1/M^5$	double	0.0	K5 = b5/rho
B6	$1/M^6$	double	0.0	K6 = b6/rho
B7	$1/M^7$	double	0.0	K7 = b7/rho
B8	$1/M^8$	double	0.0	K8 = b8/rho
XREFERENCE	M	double	0.0	reference x for interpretation of fn values
F1		double	0.0	Fractional normal field error fn=bn*xrn̂/n!, adds to K1 or b1.
F2		double	0.0	Fractional normal field error fn=bn*xrn̂/n!, adds to K2 or b2.
F3		double	0.0	Fractional normal field error fn=bn*xrn̂/n!, additive.
F4		double	0.0	Fractional normal field error fn=bn*xrn̂/n!, additive.
F5		double	0.0	Fractional normal field error fn=bn*xrn̂/n!, additive.
F6		double	0.0	Fractional normal field error fn=bn*xrn̂/n!, additive.
F7		double	0.0	Fractional normal field error fn=bn*xrn̂/n!, additive.
F8		double	0.0	Fractional normal field error fn=bn*xrn̂/n!, additive.
G1		double	0.0	Fractional skew field error.
G2		double	0.0	Fractional skew field error.
G3		double	0.0	Fractional skew field error.

CSBEND continued

A canonical kick sector dipole magnet.

Parameter Name	Units	Type	Default	Description
G4		double	0.0	Fractional skew field error.
G5		double	0.0	Fractional skew field error.
G6		double	0.0	Fractional skew field error.
G7		double	0.0	Fractional skew field error.
G8		double	0.0	Fractional skew field error.
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?
ISR1PART		short	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
SQRT_ORDER		short	0	Ignored, kept for backward compatibility only.
USE_RAD_DIST		short	0	If nonzero, overrides SYNCH_RAD and ISR, causing simulation of radiation from distributions, optionally including opening angle.
ADD_OPENING_ANGLE		short	1	If nonzero, radiation opening angle effects are added if USE_RAD_DIST is nonzero.
PHOTON_OUTPUT_FILE		STRING	NULL	output file for photons, if USE_RAD_DIST=1
PHOTON_LOW_ENERGY_CUTOFF	eV	double	0.0	Lower limit of photon energy to output.
REFERENCE_CORRECTION		short	0	If nonzero, reference trajectory is subtracted from particle trajectories to compensate for inaccuracy in integration.
TRACKING_MATRIX		short	0	If nonzero, gives order of tracking-based matrix up to third order to be used for twiss parameters etc. If zero, 2nd-order analytical matrix is used.

A canonical kick sector dipole magnet.

Parameter Name	Units	Type	Default	Description
FSE_CORRECTION		short	0	If nonzero, FSE is adjusted to compensate for edge effects when EDGE1_EFFECTS or EDGE2_EFFECTS = 2
MALIGN_METHOD		short	0	0=original, 1=new entrace-centered, 2=new body-centered
XSTEERING		short	0	use for x steering (bend plane)?
YSTEERING		short	0	use for y steering (non-bend plane)?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element provides a symplectic bending magnet with the exact Hamiltonian. For example, it retains all orders in the momentum offset and curvature. The field expansion is available to eighth order.

One pitfall of symplectic integration is the possibility of orbit and path-length errors for the reference orbit if too few kicks are used. This may be an issue for rings. Hence, one must verify that a sufficient number of kicks are being used by looking at the trajectory closure and length of an on-axis particle by tracking. Using `INTEGRATION_ORDER=4` is recommended to reduce the number of required kicks.

As of version 28.0 and later, the `REFERENCE_CORRECTION` feature is available to compensate for errors inherent in the numerical integration of the trajectories. In particular, depending on the number of kicks used, as well as the bending radius and angle, an on-axis particle may emerge from the element with a non-zero trajectory and a path-length error. With `REFERENCE_CORRECTION` set to a non-zero value, these errors are subtracted from the coordinates of all particles. There are some pitfalls to using this feature: first, one may not realize that the number of kicks is too small to provide good results, since the output trajectory of the central particle will always be (nearly) identically zero. Second, in a magnet with a gradient or other field nonuniformities, a particle may emerge centered on the ideal trajectory yet still see the impact of the gradient, sextupole, etc. For these reasons, this feature should be used with caution and only when the residual trajectory is large enough to cause problems.

Higher-order field components

Normally, one specifies the higher-order components of the field with the `K`*n*, with $n = 1$ through 8. The field expansion in the midplane is $B_y(x) = B_o * (1 + \sum_{n=1}^{8} \frac{K_n \rho_o}{n!} x^n)$. By setting the `USE_bN` flag to a nonzero value, one may instead specify the `b`*n* parameters, which are defined by the expansion $B_y(x) = B_o * (1 + \sum_{n=1}^{8} \frac{b_n}{n!} x^n)$. This is convenient if one is varying the dipole radius but wants to work in terms of constant field quality.

Setting NONLINEAR=0 turns off all the terms above K_1 (or b_1) and also turns off effects due to curvature that would normally result in a gradient producing terms of higher order.

The EXPANSION_ORDER parameter controls the order of the expansion of the nonlinear fields, so that terms are limited to $x^i y^j$ with $i+j \leq$EXPANSION_ORDER. By default, when EXPANSION_ORDER=0, the expansion order is set automatically, as follows: If the highest non-zero multipole order (specified by Kn, Bn, Fn, or Gn) is n (with $n = 1$ being quadruople), then the expansion order is set to $n + 3$. However, the expansion order is never automatically set to less than 4, unless all the multipole terms are zero, in which case the expansion always yields a constant. Since the number of polynomial terms increases like the square of the expansion order, using many multipole terms can significantly increase run time. The maximum value for the expansion order is 10.

Edge angles and edge effects

Some confusion may exist about the edge angles, particularly the signs. For a sector magnet, we have of course E1=E2=0. For a symmetric rectangular magnet, E1=E2=ANGLE/2. If ANGLE is negative, then so are E1 and E2. To understand this, imagine a rectangular magnet with positive ANGLE. If the magnet is flipped over, then ANGLE becomes negative, as does the bending radius ρ. Hence, to keep the focal length of the edge $1/f = -\tan E_i/\rho$ constant, we must also change the sign of E_i.

Several models are available for edge (or fringe) effects. Which is used depends on the settings of the EDGE_ORDER, EDGE1_EFFECTS, and EDGE2_EFFECTS parameters EDGE1_EFFECTS controls entrance edge effects while EDGE2_EFFECTS controls exit edge effects, as follows:

- 1: — Edge effects using non-symplectic method [3].
  - EDGE_ORDER<2 — linear edge focusing with δ-dependence to all orders. Generally not recommended if symplecticity is important, though when the edge effects are weak it appears acceptable.
  - EDGE_ORDER>=2 — second-order matrix edge focusing with δ-dependence to all orders. Use of this model is strongly discouraged when symplecticity matters.
- 2: — Edge effects using K. Hwang's method [45]. Note that there will be a trajectory offset when using this method that is particularly evident for small bending radii, due to extension of the fringe field outside the body of the magnet. To suppress this, adjustment of the FSE parameter can be performed automatically if FSE_CORRECTION is set to a non-zero value. If FSE_CORRECTION=1, the path-length is adjusted to match the nominal length, which is not physical; this behavior can be suppressed by setting FSE_CORRECTION=2.
  - EDGE_ORDER<2 — include only terms linear in transverse coordinates, but δ-dependence to all orders. Recommended for applications where symplecticity matters. This method is symplectic.
  - EDGE_ORDER>=2 — include all terms. This settings has been observed to produce emittance damping in some cases (particularly with large emittance and small bending radii), so users are advised to be cautious. This method is non-symplectic.
- 3: — Edge effects using symplectic method similar to [3]. The value of EDGE_ORDER is ignored. Recommended for applications where symplecticity matters.
- 4: —- Edge effects using symplectic method developed by R. Lindberg based on K. Hwang's method [45]. Note that there will be a trajectory offset when using this method that is

particularly evident for small bending radii, due to extension of the fringe field outside the body of the magnet. To suppress this, adjustment of the FSE parameter can be performed automatically if `FSE_CORRECTION` is set to a non-zero value. If `FSE_CORRECTION=1`, the path-length is adjusted to match the nominal length, which is not physical; this behavior can be suppressed by setting `FSE_CORRECTION=2`. The `EDGE_ORDER` parameter is ignored.

- 5: Edge effects using symplectic method developed by R. Lindberg, similar to method 4, but with a more detailed treatment of the curved geometry and the presence of a transverse gradients. The `EDGE_ORDER` parameter may be used to include linear terms only (value of `1`) or linear and higher-order terms (value of `2`). The program `curvedDipoleFringeCalc` may be used to prepare the data needed for this method, which involves seven fringe integrals for each edge. These are input using the FRINGEnKm parameters, where n is `1` or `2` for the entrance or exit and m is `0` through `6`.
- Other: — No edge effects.

Radiation effects

If `SYNCH_RAD` is non-zero, classical synchrotron radiation is included in tracking. Incoherent synchrotron radiation, when requested with `ISR=1`, normally uses gaussian distributions for the excitation of the electrons. (To exclude ISR for single-particle tracking, set `ISR1PART=0`.) Setting `USE_RAD_DIST=1` invokes a more sophisticated algorithm that uses correct statistics for the photon energy and number distributions. In addition, if `USE_RAD_DIST=1` one may also set `ADD_OPENING_ANGLE=1`, which includes the photon angular distribution when computing the effect on the emitting electron.

If `TRACKING_MATRIX` and `SYNCH_RAD` are non-zero, classical synchrotron radiation can be included in the ordinary matrix (e.g., for `twiss_output` and `matrix_output`) by setting `SR_IN_ORDINARY_MATRIX` to a non-zero value. Symplecticity is not assured, but the results may be interesting nonetheless. A more rigorous approach is to use `moments_output`. `SR_IN_ORDINARY_MATRIX` does not affect tracking.

Adding errors

When adding errors, care should be taken to choose the right parameters. The `FSE` and `ETILT` parameters are used for assigning errors to the strength and alignment relative to the ideal values given by `ANGLE` and `TILT`. One can also assign errors to `ANGLE` and `TILT`, but this has a different meaning: in this case, one is assigning errors to the survey itself. The reference beam path changes, so there is no orbit/trajectory error. The most common thing is to assign errors to `FSE` and `ETILT`. Note that when adding errors to `FSE`, the error is assumed to come from the power supply, which means that multipole strengths also change.

There are three modes for implementing alignment errors. Which is used is controlled by the value of the `MALIGN_METHOD` parameter:

- `MALIGN_METHOD=0` — This selects the original method, which was the only one available before version 2021.1. The misalignment is referenced to the entrance face. The `EYAW` and `EPITCH` parameters are ignored.
- `MALIGN_METHOD=1` — This selects a linearized method based on M. Venturini's work [58], with misalignment referenced to the entrance face. The `EYAW` and `EPITCH` parameters are implemented.
- `MALIGN_METHOD=2` — This selects a linearized method based on M. Venturini's work [58], with misalignment referenced to the magnet center. The `EYAW` and `EPITCH` parameters are implemented.

- MALIGN_METHOD=3 — This selects an exact method based on M. Venturini's work [58], with misalignment referenced to the entrance face. The EYAW and EPITCH parameters are implemented.

- MALIGN_METHOD=4 — This selects an exact method based on M. Venturini's work [58], with misalignment referenced to the magnet center. The EYAW and EPITCH parameters are implemented.

For elements with non-zero TILT, error displacements and rotations are performed in the lab frame. An exception is the CCBEND, where error displacements and rotations are performed in the tilted frame.

Splitting dipoles

When dipoles are long, it is common to want to split them into several pieces, to get a better look at the interior optics. When doing this, care must be exercised not to change the optics. elegant has some special features that are designed to reduce or manage potential problems. At issue is the need to turn off edge effects between the portions of the same dipole.

First, one can simply use the divide_elements command to set up the splitting. Using this command, elegant takes care of everything.

Second, one can use a series of dipoles *with the same name*. In this case, elegant automatically turns off interior edge effects. This is true when the dipole elements directly follow one another or are separated by a MARK element.

Third, one can use a series of dipoles with different names. In this case, one must also use the EDGE1_EFFECTS and EDGE2_EFFECTS parameters to turn off interior edge effects.

10.21 CSRCSBEND—Like CSBEND, but incorporates a simulation of Coherent Synchrotron radiation.

Like CSBEND, but incorporates a simulation of Coherent Synchrotron radiation.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	arc length
ANGLE	RAD	double	0.0	bend angle
K1	$1/M^2$	double	0.0	geometric quadrupole strength
K2	$1/M^3$	double	0.0	geometric sextupole strength
K3	$1/M^4$	double	0.0	geometric octupole strength
K4	$1/M^5$	double	0.0	geometric decapole strength
K5	$1/M^6$	double	0.0	geometric 12-pole strength
K6	$1/M^7$	double	0.0	geometric 14-pole strength
K7	$1/M^8$	double	0.0	geometric 16-pole strength
K8	$1/M^9$	double	0.0	geometric 18-pole strength
E1	RAD	double	0.0	entrance edge angle
E2	RAD	double	0.0	exit edge angle
TILT	RAD	double	0.0	rotation about incoming longitudinal axis
H1	$1/M$	double	0.0	entrance pole-face curvature
H2	$1/M$	double	0.0	exit pole-face curvature
HGAP	M	double	0.0	half-gap between poles
FINT		double	0.5	edge-field integral
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE		double	0.0	fractional strength error
ETILT	RAD	double	0.0	error rotation about incoming longitudinal axis
N_SLICES		long	4	Number of slices (full integrator steps).
N_KICKS		long	4	number of kicks. Deprecated. Use N_SLICES
ETILT_SIGN		short	1	Sign of ETILT relative to TILT. -1 is the old convention prior to 2020.5
NONLINEAR		short	1	include nonlinear field components?

CSRCSBEND continued

Like CSBEND, but incorporates a simulation of Coherent Synchrotron radiation.

Parameter Name	Units	Type	Default	Description
LINEARIZE		short	0	use linear matrix instead of symplectic integrator?
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
EDGE1_EFFECTS		short	1	include entrance edge effects?
EDGE2_EFFECTS		short	1	include exit edge effects?
EDGE_ORDER		short	1	order to which to include edge effects
INTEGRATION_ORDER		short	4	integration order (2, 4, or 6)
BINS		long	0	number of bins for CSR wake
BIN_ONCE		short	0	bin only at the start of the dipole?
BIN_RANGE_FACTOR		double	1.2	Factor by which to increase the range of histogram compared to total bunch length. Large value eliminates binning problems in CSRDRIFTs.
SG_HALFWIDTH		short	0	Savitzky-Golay filter half-width for smoothing current histogram. If less than 1, no SG smoothing is performed.
SG_ORDER		short	1	Savitzky-Golay filter order for smoothing current histogram
SGDERIV_HALFWIDTH		short	0	Savitzky-Golay filter half-width for taking derivative of current histogram. Defaults to SG_HALFWIDTH (if positive) or else 1.
SGDERIV_ORDER		short	1	Savitzky-Golay filter order for taking derivative of current histogram

Like CSBEND, but incorporates a simulation of Coherent Synchrotron radiation.

Parameter Name	Units	Type	Default	Description
TRAPAZOID_INTEGRATION		short	1	Select whether to use trapazoid-rule integration (default) or a simple sum.
OUTPUT_FILE		STRING	NULL	output file for CSR wakes
OUTPUT_INTERVAL		long	1	interval (in kicks) of output to OUTPUT_FILE
OUTPUT_LAST_WAKE_ONLY		short	0	output final wake only?
STEADY_STATE		short	0	use steady-state wake equations?
IGF		short	0	use integrated Greens function (requires STEADY_STATE=1)?
USE_BN		short	0	use b<n> instead of K<n>?
EXPANSION_ORDER		short	0	Order of field expansion. (0=auto)
B1	$1/M$	double	0.0	K1 = b1/rho, where rho is bend radius
B2	$1/M^2$	double	0.0	K2 = B2/rho
B3	$1/M^3$	double	0.0	K3 = B3/rho
B4	$1/M^4$	double	0.0	K4 = B4/rho
B5	$1/M^5$	double	0.0	K5 = B5/rho
B6	$1/M^6$	double	0.0	K6 = B6/rho
B7	$1/M^7$	double	0.0	K7 = B7/rho
B8	$1/M^8$	double	0.0	K8 = B8/rho
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?
ISR1PART		short	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
CSR		short	1	enable CSR computations?
BLOCK_CSR		short	0	block CSR from entering CSRDRIFT?
DERBENEV_CRITERION_MODE		STRING	disable	disable, evaluate, or enforce
PARTICLE_OUTPUT_FILE		STRING	NULL	name of file for phase-space output

CSRCSBEND continued

Like CSBEND, but incorporates a simulation of Coherent Synchrotron radiation.

Parameter Name	Units	Type	Default	Description
PARTICLE_OUTPUT_INTERVAL		long	1	interval (in kicks) of output to PARTICLE_OUTPUT_FILE
SLICE_ANALYSIS_INTERVAL		long	0	interval (in kicks) of output to slice analysis file (from slice_analysis command)
LOW_FREQUENCY_CUTOFF0		double	-1	Highest spatial frequency at which low-frequency cutoff filter is zero. If not positive, no low-frequency cutoff filter is applied. Frequency is in units of Nyquist (0.5/binsize).
LOW_FREQUENCY_CUTOFF1		double	-1	Lowest spatial frequency at which low-frequency cutoff filter is 1. If not given, defaults to LOW_FREQUENCY_CUTOFF1.
HIGH_FREQUENCY_CUTOFF0		double	-1	Spatial frequency at which smoothing (high-frequency cutoff) filter begins. If not positive, no frequency filter smoothing is done. Frequency is in units of Nyquist (0.5/binsize).
HIGH_FREQUENCY_CUTOFF1		double	-1	Spatial frequency at which smoothing (high-frequency cutoff) filter is 0. If not given, defaults to HIGH_FREQUENCY_CUTOFF0.
CLIP_NEGATIVE_BINS		short	1	If non-zero, then any bins with negative counts after the filters are applied have the counts set to zero.
WAKE_FILTER_FILE		STRING	NULL	Name of file supplying wakefield filtering data.
WFF_FREQ_COLUMN		STRING	NULL	Name of column supplying frequency values for wakefield filtering data.

CSRCSBEND continued

Like CSBEND, but incorporates a simulation of Coherent Synchrotron radiation.

Parameter Name	Units	Type	Default	Description
WFF_REAL_COLUMN		STRING	NULL	Name of column supplying real values for wakefield filtering data.
WFF_IMAG_COLUMN		STRING	NULL	Name of column supplying imaginary values for wakefield filtering data.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

For a discussion of the method behind this element, see M. Borland, "Simple method for particle tracking with coherent synchrotron radiation," Phys. Rev. ST Accel. Beams 4, 070701 (2001) and G. Stupakov and P. Emma, SLAC LCLS-TN-01-12 (2001).

Recommendations for using this element. The default values for this element are not the best ones to use. They are retained only for consistency through upgrades. In using this element, it is recommended to have 50 to 100 k particle in the simulation. Setting `BINS=600` and `SG_HALFWIDTH=1` is also recommended to allow resolution of fine structure in the beam and to avoid excessive smoothing. It is strongly suggested that the user vary these parameters and view the histogram output to verify that the longitudinal distribution is well represented by the histograms (use `OUTPUT_FILE` to obtain the histograms). For LCLS simulations, we find that the above parameters give essentially the same results as obtained with 500 k particles and up to 3000 bins.

In order to verify that the 1D approximation is valid, the user should also set `DERBENEV_CRITERION_MODE = ''evaluate''` and view the data in `OUTPUT_FILE`. Generally, the criterion should be much less than 1. See equation 11 of [20].

In order respects, this element is just like the `CSBEND` element, which provides a symplectic bending magnet that is accurate to all orders in momentum offset. Please see the manual page for `CSBEND` for more details about features not related to CSR.

Splitting dipoles: Splitting dipoles with continuation of CSR effects is possible provided the dipole sections (all of which must have the same name) are either consecutive or separated only by `MARK`, `WATCH`, or `LSCDRIFT` elements. The `LSCDRIFT` elements must have `L=0` and should have `LEFFECTIVE` set to the length of the upstream dipole segment. This allows simulating LSC and CSR within a single dipole.

10.22 CSRDRIFT—A follow-on element for CSRCSBEND that applies the CSR wake over a drift.

A follow-on element for CSRCSBEND that applies the CSR wake over a drift.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
ATTENUATION_LENGTH	M	double	0.0	exponential attenuation length for wake
DZ		double	0.0	interval between kicks
N_KICKS		long	1	number of kicks (if DZ is zero)
SPREAD		short	0	use spreading function?
USE_OVERTAKING_LENGTH		short	0	use overtaking length for ATTENUATION_LENGTH?
OL_MULTIPLIER		double	1	factor by which to multiply the overtaking length to get the attenuation length
CSR		short	1	do CSR calcuations
USE_SALDIN54		short	0	Use Saldin et al eq. 54 (NIM A 398 (1997) 373-394 for decay vs z?
SALDIN54POINTS		long	1000	Number of values of position inside bunch to average for Saldin eq 54.
SALDIN54NORM_MODE		STRING	peak	peak or first
SPREAD_MODE		STRING	full	full, simple, or radiation-only
WAVELENGTH_MODE		STRING	sigmaz	sigmaz or peak-to-peak
BUNCHLENGTH_MODE		STRING	68-percentile	rms, 68-percentile, or 90-percentile
SALDIN54_OUTPUT		STRING	NULL	Filename for output of CSR intensity vs. z as computed using Saldin eq 54.
USE_STUPAKOV		short	0	Use treatment from G. Stupakov's note of 9/12/2001?
STUPAKOV_OUTPUT		STRING	NULL	Filename for output of CSR wake vs. s as computed using Stupakov's equations.
STUPAKOV_OUTPUT_INTERVAL		long	1	Interval (in kicks) between output of Stupakov wakes.
SLICE_ANALYSIS_INTERVAL		long	0	interval (in kicks) of output to slice analysis file (from slice_analysis command)

A follow-on element for CSRCSBEND that applies the CSR wake over a drift.

Parameter Name	Units	Type	Default	Description
LINEARIZE		short	0	use linear optics for drift pieces?
LSC_INTERPOLATE		short	1	Interpolate computed LSC wake?
LSC_BINS		long	0	If non-zero, include LSC with given number of bins.
LSC_LOW_FREQUENCY_CUTOFF0		double	-1	Highest spatial frequency at which low-frequency cutoff filter is zero. If not positive, no low-frequency cutoff filter is applied. Frequency is in units of Nyquist (0.5/binsize).
LSC_LOW_FREQUENCY_CUTOFF1		double	-1	Lowest spatial frequency at which low-frequency cutoff filter is 1. If not given, defaults to LOW_FREQUENCY_CUTOFF1.
LSC_HIGH_FREQUENCY_CUTOFF0		double	-1	Spatial frequency at which smoothing filter begins for LSC. If not positive, no frequency filter smoothing is done. Frequency is in units of Nyquist (0.5/binsize).
LSC_HIGH_FREQUENCY_CUTOFF1		double	-1	Spatial frequency at which smoothing filter is 0 for LSC. If not given, defaults to HIGH_FREQUENCY_CUTOFF0.
LSC_RADIUS_FACTOR		double	1.7	Radius factor for LSC computation.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element has a number of models for simulation of CSR in drift spaces following CSRCSBEND elements. Note that all models allow support splitting the drift into multiple CSRDRIFT elements. One can also have intervening elements like quadrupoles, as often happens in chicanes. The CSR effects inside such intervening elements are applied in the CSRDRIFT downstream of the element.

For a discussion of some of the methods behind this element, see M. Borland, "Simple method for particle tracking with coherent synchrotron radiation," Phys. Rev. ST Accel. Beams 4, 070701 (2001).

N.B.: by default, this element uses 1 CSR kick (N_KICKS=1) at the center of the drift. This is usually not a good choice. I usually use the DZ parameter instead of N_KICKS, and set it to something like 0.01 (meters). The user should vary this parameter to assess how small it needs to be.

The models are as following, in order of decreasing sophistication and accuracy:

- G. Stupakov's extension of Saldin et al. Set USE_STUPAKOV=1. The most advanced model at present is based on a private communication from G. Stupakov (SLAC), which extends equation 87 of the one-dimensional treatment of Saldin et al. (NIM A 398 (1997) 373-394) to include the post-dipole region. This model includes not only the attenuation of the CSR as one proceeds along the drift, but also the change in the shape of the "wake."

  This model has the most sophisticated treatment for intervening elements of any of the models. For example, if you have a sequence `CSRCSBEND-CSRDRIFT-CSRDRIFT` and compare it with the sequence `CSRCSBEND-CSRDRIFT-DRIFT -CSRDRIFT`, keeping the total drift length constant, you'll find no change in the CSR-induced energy modulation. The model back-propagates to the beginning of the intervening element and performs the CSR computations starting from there.

  This is the slowest model to run. It uses the same binning and smoothing parameters as the upstream CSRCSBEND. If run time is a problem, compare it to the other models and use only if you get different answers.

- M. Borland's model based on Saldin et al. equations 53 and 54. Set USE_SALDIN54=1. This model computes the fall-off of the CSR wake from the work of Saldin and coworkers, as described in the reference above. It does not compute the change in the shape of the wake. The fall-off is computed approximately as well, based on the fall-off for a rectangular current distribution. The length of this rectangular bunch is taken to be twice the bunch length computed according to the BUNCHLENGTH_MODE parameter (see below). If your bunch is nearly rectangular, then you probably want BUNCHLENGTH_MODE of "90-percentile".

- Exponential attenuation of a CSR wake with unchanging shape. There are two options here. First, you can provide the attenuation length yourself, using the ATTENUATION_LENGTH parameter. Second, you can set USE_OVERTAKING_LENGTH=1 and let `elegant` compute the overtaking length for use as the attenuation length. In addition, you can multiply this result by a factor if you wish, using the OL_MULTIPLIER parameter.

- Beam-spreading model. This model is not recommended. It is based on the seemingly plausible idea that CSR spreads out just like any synchrotron radiation, thus decreasing the intensity. The model doesn't reproduce experiments.

The "Saldin 54" and "overtaking-length" models rely on computation of the bunch length, which is controlled with the BUNCHLENGTH_MODE parameter. Nominally, one should use the true RMS, but when the beam has temporal spikes, it isn't always clear that this is the best choice. The choices are "rms", "68-percentile", and "90-percentile". The last two imply using half the length determined from the given percentile in place of the rms bunch length. I usually use 68-percentile, which is the default.

10.23 CWIGGLER—Tracks through a wiggler using canonical integration routines of Y. Wu (Duke University).

Tracks through a wiggler using canonical integration routines of Y. Wu (Duke University).
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	Total length
B_MAX	T	double	0.0	Maximum on-axis magnetic field.
BX_MAX	T	double	0.0	Maximum on-axis magnetic field. Ignored if B_MAX is nonzero.
BY_MAX	T	double	0.0	Maximum on-axis magnetic field. Ignored if B_MAX is nonzero.
TGU_GRADIENT	$1/M$	double	0.0	Transverse gradient divided by maximum on-axis field, used if TGU=1.
TGU_COMP_FACTOR	$NULL$	double	1	Use to adjust constant field component to reduce trajectory error.
POLE1_FACTOR	$NULL$	double	1	Use to adjust first and last pole strength, e.g., to reduce trajectory error.
POLE2_FACTOR	$NULL$	double	1	Use to adjust second and penultimate pole strength, e.g., to reduce trajectory error.
POLE3_FACTOR	$NULL$	double	1	Use to adjust third and third-from=last pole strength, e.g., to reduce trajectory error.
DX	M	double	0.0	Misaligment.
DY	M	double	0.0	Misaligment.
DZ	M	double	0.0	Misaligment.
TILT	RAD	double	0.0	Rotation about beam axis.
PERIODS		long	0	Number of wiggler periods.
STEPS_PER_PERIOD		long	12	Integration steps per period. Must be 4*integer
INTEGRATION_ORDER		short	4	Integration order (2 or 4).
BY_FILE		STRING	NULL	Name of SDDS file with By harmonic data.

Tracks through a wiggler using canonical integration routines of Y. Wu (Duke University).

Parameter Name	Units	Type	Default	Description
BX_FILE		STRING	NULL	Name of SDDS file with Bx harmonic data.
BY_SPLIT_POLE		short	0	Use "split-pole" expansion for By?
BX_SPLIT_POLE		short	0	Use "split-pole" expansion for Bx?
SYNCH_RAD		short	0	Include classical, single-particle synchrotron radiation?
ISR		short	0	Include incoherent synchrotron radiation (quantum excitation)?
ISR1PART		short	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
SINUSOIDAL		short	0	Ideal sinusoidal wiggler? If non-zero, BX_FILE and BY_FILE are not used.
VERTICAL		short	0	If SINUSOIDAL is non-zero, then setting this to non-zero gives a vertical wiggler. Default is horizontal.
HELICAL		short	0	Ideal helical wiggler? If non-zero and SINUSOIDAL is also non-zero, BX_FILE and BY_FILE are not used.
TGU		short	0	Ideal transverse gradient undulator? If non-zero and SINUSOIDAL is also non-zero, BX_FILE and BY_FILE are not used. Give gradient in TGU_GRADIENT.
FORCE_MATCHED		short	1	Force matched dispersion for first harmonics? If non-zero, start and end of magnetic field will be inset from the ends of the device if phase is not 0 or π.

Tracks through a wiggler using canonical integration routines of Y. Wu (Duke University).

Parameter Name	Units	Type	Default	Description
FIELD_OUTPUT		STRING	NULL	Name of file to which field samples will be written. Slow, so use only for debugging.
VERBOSITY		short	0	A higher value requires more detailed printouts related to computations.
BX_CONSTANT	T	double	0.0	Constant horizontal magnetic field.
BY_CONSTANT	T	double	0.0	Constant vertical magnetic field.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a wiggler or undulator using a modified version of Ying Wu's canonical integration code for wigglers. To use the element, one must supply an SDDS file giving harmonic analysis of the wiggler field. The field expansion used by the code for a horizontally-deflecting wiggler is (Y. Wu, Duke University, private communication).

$$B_y = -|B_0| \sum_{m,n} C_{mn} \cos(k_{xl}x) \cosh(k_{ym}y) \cos(k_{zn}z + \theta_{zn}), \tag{37}$$

where $|B_0|$ is the peak value of the on-axis magnetic field, the C_{mn} give the relative amplitudes of the harmonics, the wavenumbers satisfy $k_{ym}^2 = k_{xl}^2 + k_{zn}^2$, and θ_{zn} is the phase.

The file must contain the following columns:

- The harmonic amplitude, C_{mn}, in column `Cmn`.
- The phase, in radians, in column `Phase`. The phase of the first harmonic should be 0 or π in order to have matched dispersion.
- The three wave numbers, normalized to $k_w = 2\pi/\lambda_w$, where λ_w is the wiggler period. These are given in columns `KxOverKw`, `KyOverKw`, and `KzOverKw`.

In Version 17.3 and later, for matrix computations `elegant` uses a first-order matrix derived from particle tracking when it encounterse a CWIGGLER. Tests show that this gives good agreement in the tunes from tracking and Twiss parameter calculations. For radiation integrals, an idealized sinusoidal wiggler model is used with bending radius equal to $B\rho/(B_0 \sum C_{mn})$ for each plane. Energy loss, energy spread, and horizontal emittance should be estimated accurately.

`elegant` allows specifying field expansions for on-axis B_y and B_x components, so one can model a helical wiggler. However, in this case one set of components should have $\theta_{zn} = 0$ or $\theta_{zn} = \pi$, while the other should have $\theta_{zn} = \pm\pi/2$. Using Wu's code, the latter set will not have matched dispersion. Our modified version solves this by delaying the beginning of the field components in question by $\lambda/4$ and ending the field prematurely by $3\lambda/4$. This causes all the fields to start and end at the crest, which ensures matched dispersion. The downside is that the (typically) vertical wiggler component is missing a full period of field. One can turn off this behavior by setting `FORCE_MATCHED=0`.

Additional field expansions

Y. Wu's code included field expansions for a vertically-deflecting wiggler as well as the horizontally-deflecting wiggler given above. In both cases, these expansions are suitable for a wiggler with two poles that are above/below or left/right of the beam axis. They are not always suitable for devices with more complex pole geometries.

Another geometry that is important is a "split pole" wiggler, in which each pole is made from two pieces. Such configurations are seen, for example, in devices used to produce variable polarization. In such cases, the expansion given above may not be appropriate. Here, we summarize the form of the various expansions that `elegant` supports. For brevity, we show the form of a single harmonic component.

Horizontal wiggler, normal poles, produces B_y only on-axis. Specified by setting `BY_SPLIT_POLE=0`, and giving `BY_FILE` or `SINUSOIDAL=1` with `VERTICAL=0`.

$$B_x = |B_0| \frac{k_x \cos(k_z z + \phi) \sin(k_x x) \sinh(k_y y)}{k_y} \tag{38}$$

$$B_y = -|B_0| \cos(k_x x) \cos(k_z z + \phi) \cosh(k_y y) \tag{39}$$

where $k_y^2 = k_x^2 + k_z^2$.

Experimental feature: Horizontal wiggler, normal poles, with transverse gradient, produces B_y only on-axis. Specified by setting `BY_SPLIT_POLE=0`, `SINUSOIDAL=1`, `TGU=1`, `VERTICAL=0`. The TGU normalized gradient is given using the `TGU_GRADIENT` parameter. Taking a as the normalized gradient, the fields are[54]

$$B_x = \frac{a\,|B_0| \sinh k_u y \cos k_u z}{k_u} \tag{40}$$

$$B_y = |B_0| \left((1 + ax) \cosh k_u y \cos k_u z + \frac{aC}{2k_u^2} \frac{e\,|B_0|}{\gamma m_e c} \right) \tag{41}$$

$$B_z = -|B_0|\,(1 + ax) \sinh k_u y \sin k_u z, \tag{42}$$

where $k_u = k_y = k_z$, $k_x = 0$ is assumed, γ is the central relativistic factor for the beam and C is given by the `TGU_COMP_FACTOR` parameter. This factor, and the term it multiplies, is present in order to help suppress the trajectory error at the end of the device. It may require adjustment in order to achieve the desired level of correction. In addition, the user may need to adjust the pole-strength factors and include external misalignments and steering magnets in order to supress not only the trajectory error, but also dispersion errors.

Horizontal wiggler, split poles, produces B_y only on-axis. Specified by setting `BY_SPLIT_POLE=1`, and giving `BY_FILE` or `SINUSOIDAL=1` with `VERTICAL=0`.

$$B_x = -|B_0| \frac{k_x \cos(k_z z + \phi) \sin(k_y y) \sinh(k_x x)}{k_y} \tag{43}$$

$$B_y = -|B_0| \cos(k_y y) \cos(k_z z + \phi) \cosh(k_x x) \tag{44}$$

where $k_x^2 = k_y^2 + k_z^2$.

Vertical wiggler, normal poles, produces B_x only on-axis. Specified by setting BX_SPLIT_POLE=0, and giving BX_FILE or SINUSOIDAL=1 with either VERTICAL=1 or HELICAL=1.

$$
\begin{aligned}
B_x &= |B_0| \cos(k_y y) \cos(k_z z + \phi) \cosh(k_x x) && (45) \\
B_y &= -|B_0| \frac{k_y \cos(k_z z + \phi) \sin(k_y y) \sinh(k_x x)}{k_x} && (46)
\end{aligned}
$$

where $k_x^2 = k_y^2 + k_z^2$.

Vertical wiggler, split poles, produces B_x only on-axis. Specified by setting BX_SPLIT_POLE=1, and giving BX_FILE or SINUSOIDAL=1 with either VERTICAL=1 or HELICAL=1.

$$
\begin{aligned}
B_x &= |B_0| \cos(k_x x) \cos(k_z z + \phi) \cosh(k_y y) && (47) \\
B_y &= |B_0| \frac{k_y \cos(k_z z + \phi) \sin(k_x x) \sinh(k_y y)}{k_x} && (48)
\end{aligned}
$$

where $k_y^2 = k_x^2 + k_z^2$.

Splitting wigglers

The CWIGGLER element supports a limited ability to split a long element into parts using the element_divisions command or the divide_elements pararameter of the run_setup command.

In addition, if contiguous CWIGGLER elements are seen in a beamline, they will be treated as part of the same element. That means that the pole factors will be ignored except at the ends of the sequence. For this purpose, CWIGGLER elements separated only by MARK or WATCH elements are considered to be contiguous.

10.24 DRIF—A drift space implemented as a matrix, up to 2nd order. Use EDRIFT for symplectic tracking.

A drift space implemented as a matrix, up to 2nd order. Use EDRIFT for symplectic tracking.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
ORDER		short	0	matrix order
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.25 DSCATTER—A scattering element to add random changes to particle coordinates according to a user-supplied distribution function

A scattering element to add random changes to particle coordinates according to a user-supplied distribution function
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
PLANE		STRING	NULL	Plane to scatter: xp, yp, dp (dp is deltaP/P)
FILENAME		STRING	NULL	Name of SDDS file containing distribution function.
VALUENAME		STRING	NULL	Name of column containing the independent variable for the distribution function data.
CDFNAME		STRING	NULL	Name of column containing the cumulative distribution function data.
PDFNAME		STRING	NULL	Name of column containing the probability distribution function data.
ONCEPERPARTICLE		long	0	If nonzero, each particle can only get scattered once by this element.
FACTOR		double	1	Factor by which to multiply the independent variable values.
PROBABILITY		double	1	Probability that any particle will be selected for scattering.
GROUPID		long	-1	Group ID number (nonnegative integer) for linking once-per-particle behavior of multiple elements.
RANDOMSIGN		long	0	If non-zero, then the scatter is given a random sign. Useful if distribution data is one-sided.
LIMITPERPASS		long	-1	Maximum number of particles that will be scattered on each pass.
LIMITTOTAL		long	-1	Maximum number of particles that will be scatter for each step.

DSCATTER continued

A scattering element to add random changes to particle coordinates according to a user-supplied distribution function

Parameter Name	Units	Type	Default	Description
STARTONPASS		long	0	Pass number to start on.
ENDONPASS		long	-1	Pass number to end on (inclusive). Ignored if negative.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.26 ECOL—An elliptical collimator.

An elliptical collimator.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
X_MAX	M	double	0.0	half-axis in x
Y_MAX	M	double	0.0	half-axis in y
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
OPEN_SIDE		STRING	NULL	which side, if any, is open (+x, -x, +y, -y)
EXPONENT		short	2	Exponent for boundary equation. 2 is ellipse.
YEXPONENT		short	0	y exponent for boundary equation. 2 is ellipse. If 0, defaults to EXPONENT
INVERT		short	0	If non-zero, particles inside the aperture are lost while those outside are transmitted.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.27 EDRIFT—Tracks through a drift with no approximations (Exact DRIFT).

Tracks through a drift with no approximations (Exact DRIFT).
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.28 EHKICK—A horizontal steering dipole implemented using an exact hard-edge model

A horizontal steering dipole implemented using an exact hard-edge model
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
KICK	RAD	double	0.0	kick angle
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
CALIBRATION		double	1	factor applied to obtain kick
LERAD		double	0.0	if L=0, use this length for radiation computations
STEERING		short	1	use for steering?
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?
STEERING_MULTIPOLES		STRING	NULL	input file for systematic multipole content of steering kicks
RANDOM_MULTIPOLES		STRING	NULL	input file for random multipoles content of steering kicks
RANDOM_MULTIPOLE_FACTOR		double	1	Factor by which to multiply random multipoles
STEERING_MULTIPOLE_FACTOR		double	1	Factor by which to multiply steering multipoles
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

Note that `closed_orbit` and `correct` command may report orbit convergence problems when using `EHKICK` in place of `HKICK`. This may be resolved by increasing the `closed_orbit_accuracy` parameter.

If requested, synchrotron radiation effects are imposed as a kick at the end of the element.

10.29 EKICKER—A combined horizontal/vertical steering dipole implemented using an exact hard-edge model

A combined horizontal/vertical steering dipole implemented using an exact hard-edge model
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
HKICK	RAD	double	0.0	horizontal kick angle
VKICK	RAD	double	0.0	vertical kick angle
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
HCALIBRATION		double	1	factor applied to obtain horizontal kick
VCALIBRATION		double	1	factor applied to obtain vertical kick
LERAD		double	0.0	if L=0, use this length for radiation computations
STEERING		short	1	use for steering?
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?
STEERING_MULTIPOLES		STRING	NULL	input file for systematic multipole content of steering kicks
RANDOM_MULTIPOLES		STRING	NULL	input file for random multipoles content of steering kicks
RANDOM_MULTIPOLE_FACTOR		double	1	Factor by which to multiply random multipoles
STEERING_MULTIPOLE_FACTOR		double	1	Factor by which to multiply steering multipoles
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

Note that `closed_orbit` and `correct` command may report orbit convergence problems when using `EKICKER` in place of `KICKER`. This may be resolved by increasing the `closed_orbit_accuracy` parameter.

If requested, synchrotron radiation effects are imposed as a kick at the end of the element.

10.30 EMATRIX—Explicit matrix input with data in the element definition, rather than in a file.

Explicit matrix input with data in the element definition, rather than in a file.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	Length (used only for position computation)
ANGLE	RAD	double	0.0	Angle (used only for position computation)
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
TILT	RAD	double	0.0	Tilt angle
YAW	RAD	double	0.0	Yaw angle
PITCH	RAD	double	0.0	Pitch angle
ORDER		short	0	
C1	M	double	0.0	
C2		double	0.0	
C3	M	double	0.0	
C4		double	0.0	
C5	M	double	0.0	
C6		double	0.0	Change in momentum offset
DELTAP		double	0.0	Change in central momentum (beta*gamma)
R11		double	0.0	
R12	M	double	0.0	
R13		double	0.0	
R14	M	double	0.0	
R15		double	0.0	
R16	M	double	0.0	
R21	$1/M$	double	0.0	
R22		double	0.0	
R23	$1/M$	double	0.0	
R24		double	0.0	
R25	$1/M$	double	0.0	
R26		double	0.0	
R31		double	0.0	
R32	M	double	0.0	
R33		double	0.0	
R34	M	double	0.0	
R35		double	0.0	

EMATRIX continued

Explicit matrix input with data in the element definition, rather than in a file.

Parameter Name	Units	Type	Default	Description
R36	M	double	0.0	
R41	$1/M$	double	0.0	
R42		double	0.0	
R43	$1/M$	double	0.0	
R44		double	0.0	
R45	$1/M$	double	0.0	
R46		double	0.0	
R51		double	0.0	
R52	M	double	0.0	
R53		double	0.0	
R54	M	double	0.0	
R55		double	0.0	
R56	M	double	0.0	
R61	$1/M$	double	0.0	
R62		double	0.0	
R63	$1/M$	double	0.0	
R64		double	0.0	
R65	$1/M$	double	0.0	
R66		double	0.0	
T111	$1/M$	double	0.0	
T121		double	0.0	
T122	M	double	0.0	
T131	$1/M$	double	0.0	
T132		double	0.0	
T133	$1/M$	double	0.0	
T141		double	0.0	
T142	M	double	0.0	
T143		double	0.0	
T144	M	double	0.0	
T151	$1/M$	double	0.0	
T152		double	0.0	
T153	$1/M$	double	0.0	
T154		double	0.0	
T155	$1/M$	double	0.0	
T161		double	0.0	
T162	M	double	0.0	

EMATRIX continued

Explicit matrix input with data in the element definition, rather than in a file.

Parameter Name	Units	Type	Default	Description
T163		double	0.0	
T164	M	double	0.0	
T165		double	0.0	
T166	M	double	0.0	
T211	$1/M^2$	double	0.0	
T221	$1/M$	double	0.0	
T222		double	0.0	
T231	$1/M^2$	double	0.0	
T232	$1/M$	double	0.0	
T233	$1/M^2$	double	0.0	
T241	$1/M$	double	0.0	
T242		double	0.0	
T243	$1/M$	double	0.0	
T244		double	0.0	
T251	$1/M^2$	double	0.0	
T252	$1/M$	double	0.0	
T253	$1/M^2$	double	0.0	
T254	$1/M$	double	0.0	
T255	$1/M^2$	double	0.0	
T261	$1/M$	double	0.0	
T262		double	0.0	
T263	$1/M$	double	0.0	
T264	1	double	0.0	
T265	$1/M$	double	0.0	
T266		double	0.0	
T311	$1/M$	double	0.0	
T321		double	0.0	
T322	M	double	0.0	
T331	$1/M$	double	0.0	
T332		double	0.0	
T333	$1/M$	double	0.0	
T341		double	0.0	
T342	M	double	0.0	
T343		double	0.0	
T344	M	double	0.0	
T351	$1/M$	double	0.0	

EMATRIX continued

Explicit matrix input with data in the element definition, rather than in a file.

Parameter Name	Units	Type	Default	Description
T352		double	0.0	
T353	$1/M$	double	0.0	
T354		double	0.0	
T355	$1/M$	double	0.0	
T361		double	0.0	
T362	M	double	0.0	
T363		double	0.0	
T364	M	double	0.0	
T365		double	0.0	
T366	M	double	0.0	
T411	$1/M^2$	double	0.0	
T421	$1/M$	double	0.0	
T422		double	0.0	
T431	$1/M^2$	double	0.0	
T432	$1/M$	double	0.0	
T433	$1/M^2$	double	0.0	
T441	$1/M$	double	0.0	
T442		double	0.0	
T443	$1/M$	double	0.0	
T444		double	0.0	
T451	$1/M^2$	double	0.0	
T452	$1/M$	double	0.0	
T453	$1/M^2$	double	0.0	
T454	$1/M$	double	0.0	
T455	$1/M^2$	double	0.0	
T461	$1/M$	double	0.0	
T462		double	0.0	
T463	$1/M$	double	0.0	
T464	1	double	0.0	
T465	$1/M$	double	0.0	
T466		double	0.0	
T511	$1/M$	double	0.0	
T521		double	0.0	
T522	M	double	0.0	
T531	$1/M$	double	0.0	
T532		double	0.0	

EMATRIX continued

Explicit matrix input with data in the element definition, rather than in a file.

Parameter Name	Units	Type	Default	Description
T533	$1/M$	double	0.0	
T541		double	0.0	
T542	M	double	0.0	
T543		double	0.0	
T544	M	double	0.0	
T551	$1/M$	double	0.0	
T552		double	0.0	
T553	$1/M$	double	0.0	
T554		double	0.0	
T555	$1/M$	double	0.0	
T561		double	0.0	
T562	M	double	0.0	
T563		double	0.0	
T564	M	double	0.0	
T565		double	0.0	
T566	M	double	0.0	
T611	$1/M^2$	double	0.0	
T621	$1/M$	double	0.0	
T622		double	0.0	
T631	$1/M^2$	double	0.0	
T632	$1/M$	double	0.0	
T633	$1/M^2$	double	0.0	
T641	$1/M$	double	0.0	
T642		double	0.0	
T643	$1/M$	double	0.0	
T644		double	0.0	
T651	$1/M^2$	double	0.0	
T652	$1/M$	double	0.0	
T653	$1/M^2$	double	0.0	
T654	$1/M$	double	0.0	
T655	$1/M^2$	double	0.0	
T661	$1/M$	double	0.0	
T662		double	0.0	
T663	$1/M$	double	0.0	
T664	1	double	0.0	
T665	$1/M$	double	0.0	

Explicit matrix input with data in the element definition, rather than in a file.

Parameter Name	Units	Type	Default	Description
T666		double	0.0	
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

Note that the default value of all matrix elements is 0. This can produce unexpected results if one imagines by mistake that the default values give a unit matrix, for example.

10.31 EMITTANCE—Applies a linear transformation to the beam to force the emittance to given values.

Applies a linear transformation to the beam to force the emittance to given values.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
EMITX	M	double	-1	horizontal emittance
EMITY	M	double	-1	vertical emittance
EMITNX	M	double	-1	horizontal normalized emittance
EMITNY	M	double	-1	vertical normalized emittance
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element allows changing the emittance of a beam during tracking. It is intended to be used to modify the emittance "slightly" to agree with, for example, experimental measurements.

The LCLS provides an example application: we track a beam from a photo-injector simulation through a laser/undulator beam heater and then through the entire linac. The beam emittance and twiss parameters are measured at a diagnostic downstream of the laser heater. We can insert an EMITTANCE element and a TWISS element at the location of the diagnostic to force the beam properties to the exact values that are measured. This compensates for imperfect modeling of the photo-injector while allowing us to conveniently model the system between the photo-injector and the point at which the emittance is measured.

10.32 ENERGY—An element that matches the central momentum to the beam momentum, or changes the central momentum or energy to a specified value.

An element that matches the central momentum to the beam momentum, or changes the central momentum or energy to a specified value.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
CENTRAL_ENERGY	MC^2	double	0.0	desired central gamma
CENTRAL_MOMENTUM	MC	double	0.0	desired central beta*gamma
MATCH_BEAMLINE		long	0	if nonzero, beamline reference momentum is set to beam average momentum
MATCH_PARTICLES		long	0	if nonzero, beam average momentum is set to beamline reference momentum
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.33 EVKICK—A vertical steering dipole implemented using an exact hard-edge model

A vertical steering dipole implemented using an exact hard-edge model
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
KICK	RAD	double	0.0	kick angle
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
CALIBRATION		double	1	factor applied to obtain kick
LERAD		double	0.0	if L=0, use this length for radiation computations
STEERING		short	1	use for steering?
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?
STEERING_MULTIPOLES		STRING	NULL	input file for systematic multipole content of steering kicks
RANDOM_MULTIPOLES		STRING	NULL	input file for random multipoles content of steering kicks
RANDOM_MULTIPOLE_FACTOR		double	1	Factor by which to multiply random multipoles
STEERING_MULTIPOLE_FACTOR		double	1	Factor by which to multiply steering multipoles
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

Note that `closed_orbit` and `correct` command may report orbit convergence problems when using `EVKICK` in place of `VKICK`. This may be resolved by increasing the `closed_orbit_accuracy` parameter.

If requested, synchrotron radiation effects are imposed as a kick at the end of the element.

10.34 FLOOR—Sets floor coordinates

Sets floor coordinates
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
X		double	0.0	X coordinate
Y		double	0.0	Y coordinate
Z		double	0.0	Z coordinate
THETA		double	0.0	theta value
PHI		double	0.0	phi value
PSI		double	0.0	psi value
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.35 FMULT—Multipole kick element with coefficient input from an SDDS file.

Multipole kick element with coefficient input from an SDDS file.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE		double	0.0	fractional strength error
FACTOR		double	1	factor by which to multiply multipole strengths
N_KICKS		long	0	Deprecated. Use N_SLICES.
N_SLICES		long	1	Number of slices (full integrator steps).
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
FILENAME		STRING	NULL	name of file containing multipole data
SQRT_ORDER		short	0	Ignored, kept for backward compatibility only.
UNTILTED_MATRIX		short	0	If nonzero, TILT is ignored for purposes of matrix computation.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a multipole element using a 4th-order sympletic integration. Specification of the multipole strength is through an SDDS file. The file is expected to contain a single page of data with the following elements:

1. An integer column named `order` giving the order of the multipole. The order is defined as $(N_{poles} - 2)/2$, so a quadrupole has order 1, a sextupole has order 2, and so on.
2. A floating point column named `KnL` giving the integrated strength of the multipole, K_nL,

where n is the order. The units are $1/m^n$.

3. A floating point column named `JnL` giving the integrated strength of the skew multipole, $J_n L$, where n is the order. The units are $1/m^n$.

The `MULT` element is also available, which allows the same functionality without an external file, for a single component.

The transport matrix for `FMULT` elements is determined by tracking and will affect the tunes, chromaticities, etc. However, in some cases if `TILT` is nonzero, it may happen that the phase advance jumps by nearly 2π; this anomaly can be suppressed by setting `UNTILTED_MATRIX` to a nonzero value.

10.36 FRFMODE—One or more beam-driven TM monopole modes of an RF cavity, with data from a file.

One or more beam-driven TM monopole modes of an RF cavity, with data from a file.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
FILENAME		STRING	NULL	input file
BIN_SIZE	S	double	0.0	bin size for current histogram
N_BINS		long	20	number of bins for current histogram
RIGID_UNTIL_PASS		long	0	don't affect the beam until this pass
USE_SYMM_DATA		long	0	use "Symm" columns from URMEL output file?
FACTOR		double	1	factor by which to multiply shunt impedances
CUTOFF	HZ	double	0.0	If >0, cutoff frequency. Modes above this frequency are ignored.
OUTPUT_FILE		STRING	NULL	Output file for voltage in each mode.
FLUSH_INTERVAL		long	1	Interval in passes at which to flush output data.
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the impedance to full strength.
RESET_FOR_EACH_STEP		long	1	If nonzero, voltage and phase are reset for each simulation step.
LONG_RANGE_ONLY		long	0	If nonzero, induced voltage from present turn does not affect bunch. Short range wake should be included via WAKE or ZLONGIT element.
N_CAVITIES		long	1	effect is multiplied by this number, simulating N identical cavities
BUNCHED_BEAM_MODE		long	1	If non-zero, then do calculations bunch-by-bunch.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a set of beam-driven monopole modes in a cavity using the fundamental theorem of beam loading and phasor rotation. It is similar to `RFMODE`, but it allows faster simulation of more than one mode. Also, the mode data is specified in an SDDS file. This file can be generated using the APS version of URMEL, or by hand. It must have the following columns and units:

1. `Frequency` — The frequency of the mode in Hz. Floating point.
2. `Q` — The quality factor. Floating point.
3. `ShuntImpedance` or `ShuntImpedanceSymm` — The shunt impedance in Ohms, defined as $V^2/(2*P)$ (i.e., the "circuit definition"). Floating point. By default, `ShuntImpedance` is used. However, if the parameter `USE_SYMM_DATA` is non-zero, then `ShuntImpedanceSymm` is used. The latter is the full-cavity shunt impedance that URMEL computes by assuming that the input cavity used is one half of a symmetric cavity.

The file may also have the following column:

1. `beta` — Normalized load impedance (dimensionless). Floating point. If not given, the $\beta = 0$ is assumed for all modes.

In many simulations, a transient effect may occur when using this element because, in the context of the simulation, the impedance is switched on instantaneously. This can give a false indication of the threshold for instability. The `RAMP_PASSES` parameter should be used to prevent this by slowly ramping the impedance to full strength. This idea is from M. Blaskiewicz (BNL).

Normally, the field dumped in the cavity by one particle affects trailing particles in the same turn. However, if one is also using a `WAKE` or `ZLONGIT` element to simulate the short-range wake of the cavity, this would be double-counting. In that case, one can use `LONG_RANGE_ONLY=1` to suppress the same-turn effects of the `RFMODE` element.

10.37 FTABLE—Tracks through a magnetic field which is expressed by a SDDS table.

Tracks through a magnetic field which is expressed by a SDDS table.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	The effective field length measured along a straight line.
ANGLE	*RAD*	double	0.0	The designed bending angle
L1	*M*	double	0.0	The left fringe field length.
L2	*M*	double	0.0	The right fringe field length. L1+L+L2=Total z span in the input field table.
E1	*RAD*	double	0.0	The designed entrance edge angle
E2	*RAD*	double	0.0	The designed exit edge angle
TILT	*RAD*	double	0.0	rotation about incoming longitudinal axis
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
DZ	*M*	double	0.0	misalignment
FACTOR		double	1	Factor by which to multiply field data.
THRESHOLD		double	1e-08	Fields smaller than this are considered 0.
INPUT_FILE		STRING	NULL	Name of SDDS file which contains field data.
N_KICKS		long	1	Number of kicks into which to split the element.
VERBOSE		short	0	Used for debugging code. Not applicable to Pelegant
SIMPLE_INPUT		short	0	If non-zero, use simple input format.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is used for tracking through an arbitrary magnetic field when its values are known at regularly spaced grid points and it is hard to find a suitable model to describe it. The input magnet parameter and coordinate system definition are illustrated in Fig:1.

The `THRESHOLD` parameter sets the magnitude of magnetic field below which the field is considered zero. If this is too small, there may be numerical problems.

The field data is provided in an SDDS file, with two formats available. The recommended format can be used if the `SIMPLE_INPUT` parameter is non-zero.

Simple input format — This format is shared with the `BMXYZ` and `BRAT` elements and is more convenient than the original, default format. The field map file is an SDDS file with the following columns:

- **x**, **y**, **z** — Transverse coordinates in meters (units should be "m").
- **Bx**, **By**, **Bz** — Field values in Tesla (units should be "T").

The field map file must contain a rectangular grid of points, equispaced (separately) in x, y, and z. There should be no missing values in the grid (this is not checked by `elegant`). In addition, the x values must vary fastest as the values are accessed in row order, then the y values. To ensure that this is the case, use the following command on the field file:

sddssort *fieldFile* -column=z,incr -column=y,incr -column=x,incr

N.B.: Particles are injected into the field region with z=0. Hence, one would normally want the minimum value of z to be 0.

Original input format — This format is difficult to understand and set up. Although it is not recommended, it is the default at present for historical reasons.

The field data is saved in a 3 pages (B_x, B_y, B_z) 3D histogram SDDS table (see `MHISTOGRAM` for detail). An example is shown in Fig:2. This SDDS file must have one column `Frequency` to store the field data in Tesla, and following parameters:

- `ND` — Type "long"; Value "3".
- `Variable00Name`, `Variable01Name`,`Variable02Name` — Type "string"; Value "x", "y", "z".
- `Variable00Min`, `Variable01Min`, `Variable02Min` — Type "double"; Value: the minimum boundary coordinates of "x", "y", "z" in meter. `Variable02Min` (z_min) must start from zero.
- `Variable00Max`,`Variable01Max`, `Variable02Max` — Type "double"; Value: the maximum boundary coordinates of "x", "y", "z" in meter.
- `Variable00Interval`, `Variable01Interval`,`Variable02Interval` — Type "double"; Value of the grid size of "x", "y", "z" in meter.
- `Variable00Dimension`,`Variable01Dimension`, `Variable02Dimension` — Type "long"; Value of total number of grid points in "x", "y", "z". For example: Variable00Dimension =(Variable00Max-Variable00Min)/Variable00Interval + 1.

N.B.: Particles are injected into the field region with z=0. Hence, one would normally want `Variable02Min`=0. If `Variable02Min`<0, data ahead of the injection point.

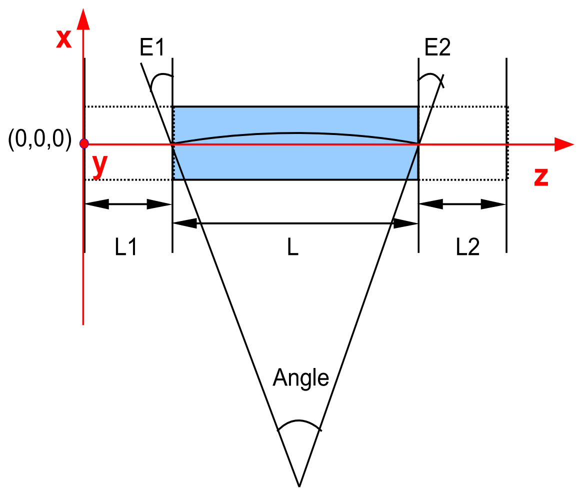

Figure 1: Illustration of coordinate system and magnet definition.

sddsprintout -para=* ftable.input

ND = 3	Variable00Name = x
Variable01Name = y	Variable02Name = z
Variable00Min (m) = -1.700000e-02	Variable01Min (m) = -5.000000e-03
Variable02Min (m) = 0.000000e+00	Variable00Max (m) = 1.700000e-02
Variable01Max (m) = 5.000000e-03	Variable02Max (m) = 1.250000e-01
Variable00Interval (m) = 1.000000e-03	Variable01Interval (m) = 1.000000e-03
Variable02Interval (m) = 1.250000e-0	Variable00Dimension = 35
Variable01Dimension = 11	Variable02Dimension = 101

sddsprintout -col=* ftable.input (page 2, By field)

x_index	y_index	z_index	Frequency T
0	0	0	1
0	0	100	1
…	…	…	…
0	10	100	1
…	…	…	…
1	10	100	1
…	…	…	…
34	10	100	1

Figure 2: Example of SDDS input file. The column x_index, y_index, z_index is not the necessary part, it's shown here just for clarifying how the data is arranged

10.38 FTRFMODE—One or more beam-driven TM dipole modes of an RF cavity, with data from a file.

One or more beam-driven TM dipole modes of an RF cavity, with data from a file.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
FILENAME		STRING	NULL	input file
BIN_SIZE	S	double	0.0	bin size for current histogram (use 0 for autosize)
N_BINS		long	20	number of bins for current histogram
RIGID_UNTIL_PASS		long	0	don't affect the beam until this pass
USE_SYMM_DATA		long	0	use "Symm" columns from URMEL output file?
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
XFACTOR		double	1	factor by which to multiply shunt impedances
YFACTOR		double	1	factor by which to multiply shunt impedances
CUTOFF	HZ	double	0.0	If >0, cutoff frequency. Modes above this frequency are ignored.
OUTPUT_FILE		STRING	NULL	Output file for voltage in each mode.
FLUSH_INTERVAL		long	1	Interval in passes at which to flush output data.
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the impedance to full strength.
RESET_FOR_EACH_STEP		long	1	If nonzero, voltage and phase are reset for each simulation step.
LONG_RANGE_ONLY		long	0	If nonzero, induced voltage from present turn does not affect bunch. Short range wake should be included via WAKE or ZLONGIT element.
N_CAVITIES		long	1	effect is multiplied by this number, simulating N identical cavities

FTRFMODE continued

One or more beam-driven TM dipole modes of an RF cavity, with data from a file.

Parameter Name	Units	Type	Default	Description
BUNCHED_BEAM_MODE		long	1	If non-zero, then do calculations bunch-by-bunch.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a set of beam-driven dipole modes in a cavity using the fundamental theorem of beam loading and phasor rotation. It is similar to `TRFMODE`, but it allows faster simulation of more than one mode. Also, the mode data is specified in an SDDS file. This file can be generated using the APS version of URMEL, or by hand. It must have the following columns and units:

1. `Frequency` — The frequency of the mode in Hz. Floating point.
2. `Q` — The quality factor. Floating point.
3. `ShuntImpedance` or `ShuntImpedanceSymm` — The shunt impedance in Ohms/m, defined as $V^2/(2*P)/x$ or $V^2/(2*P)/y$ (i.e., "circuit definition"). Floating point. By default, `ShuntImpedance` is used. However, if the parameter `USE_SYMM_DATA` is non-zero, then `ShuntImpedanceSymm` is used. The latter is the full-cavity shunt impedance that URMEL computes by assuming that the input cavity used is one half of a symmetric cavity.

The file may also have the following columns:

1. `beta` — Normalized load impedance (dimensionless). Floating point. If not given, the $\beta = 0$ is assumed for all modes.
2. `xMode` — If given, then only modes for which the value is nonzero will produce an x-plane kick. Integer. If not given, all modes affect the x plane.
3. `yMode` — If given, then only modes for which the value is nonzero will produce an y-plane kick. Integer. If not given, all modes affect the y plane.

In many simulations, a transient effect may occur when using this element because, in the context of the simulation, the impedance is switched on instantaneously. This can give a false indication of the threshold for instability. The `RAMP_PASSES` parameter should be used to prevent this by slowly ramping the impedance to full strength. This idea is from M. Blaskiewicz (BNL).

Normally, the field dumped in the cavity by one particle affects trailing particles in the same turn. However, if one is also using a `TRWAKE` or `ZTRANSVSE` element to simulate the short-range wake of the cavity, this would be double-counting. In that case, one can use `LONG_RANGE_ONLY=1` to suppress the same-turn effects of the `RFMODE` element.

10.39 GFWIGGLER—Tracks through a wiggler using generate function method of J. Bahrdt and G. Wuestefeld (BESSY, Berlin, Germany).

Tracks through a wiggler using generate function method of J. Bahrdt and G. Wuestefeld (BESSY, Berlin, Germany).
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	Total length
B_MAX	T	double	0.0	Maximum on-axis magnetic field at gap=GAP0 and equal longitudinal phases of PHASE_1,2,3,4
SHIM_SCALE		double	1	Scaling factor of shim correction field.
DX	M	double	0.0	Misaligment.
DY	M	double	0.0	Misaligment.
DZ	M	double	0.0	Misaligment.
TILT	RAD	double	0.0	Rotation about beam axis.
PERIODS		long	0	Total number of wiggler periods. Include end poles
STEP		long	1	Number of normal periods to track for each step
ORDER		short	0	Order=3 including the 3rd order terms. Otherwise using 2nd order formula.
END_POLE		short	1	The ending poles are treated as 2 half periods at each sides of the wiggler with reducing field strength, such as 0.25, -0.75, ..., 0.75, -0.25. Periods has to >2
SHIM_ON		short	0	Include shim correction
INPUT_FILE		STRING	NULL	Name of SDDS file with By harmonic data given at GAP0 and equal longitudinal phases.
SHIM_INPUT		STRING	NULL	Name of SDDS file with shim field integral harmonic data given at GAP0.
SYNCH_RAD		short	0	Include classical, single-particle synchrotron radiation?
ISR		short	0	Include incoherent synchrotron radiation (quantum excitation)?

Tracks through a wiggler using generate function method of J. Bahrdt and G. Wuestefeld (BESSY, Berlin, Germany).

Parameter Name	Units	Type	Default	Description
ISR1PART		short	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
X0	M	double	0.0	Offset of magnet row center in meter.
GAP0	M	double	0.0	Nominal magnetic gap.
D_GAP	M	double	0.0	Delta gap: actual gap - nominal gap
PHASE_1	RAD	double	0.0	Longitudinal phase of the first row (top right)
PHASE_2	RAD	double	0.0	Longitudinal phase of the second row (top left)
PHASE_3	RAD	double	0.0	Longitudinal phase of the third row (bottom left)
PHASE_4	RAD	double	0.0	Longitudinal phase of the fourth row (bottom right)
VERBOSITY		short	0	A higher value requires more detailed printouts related to computations.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

N.B.: at present this element is *not* included in computations of beam moments (`moments_output` command).

This element simulates a wiggler or undulator using the generate function method given by J. Bahrdt and G. Wüstefeld ("Symplectic tracking and compensation of dynamic field integrals in complex undulator structures," PRSTAB 14, 040703, 2011.).

To use the element, one must supply an SDDS file giving harmonic analysis of the wiggler field. The field expansion used by the code is for a wiggler working at the nominal gap and provide pure horizontal deflecting to the on-axis beam. See `CWIGGLER`, horizontal wiggler with normal poles, for detail explaination of the field expansion and format of the input file. Besides the required columns of `Cmn`, `KxOverKw`, `KyOverKw`, and `KzOverKw` by the `CWIGGLER` elements, two more input columns are needed:

- The longitudinal harmonic number, n, in column `zHarm`.
- The horizontal harmonic number of l, in column `xHarm`.

If a file include all required columns from CWIGGLER and GFWIGGLER then user can use either of the both methods for simulating a horizontal planar wiggler.

An universal wiggler field, which be used for generating an arbitrary polarization, can be derived by given different longitudinal phase parameters: PHASE_1,2,3,4. The photon energy can be varied by a non-zero D_GAP value.

10.40 GKICKMAP—An ordinary kick map (use UKICKMAP for undualtors).

An ordinary kick map (use UKICKMAP for undualtors).
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FACTOR		double	1	Factor by which to multiply the kicks.
XY_FACTOR		double	1	Factor by which to multiply the x and y values in the input file.
YAW		double	0.0	Yaw angle of the device. Meaningful only if N_KICKS is not 1.
INPUT_FILE		STRING	NULL	Name of SDDS file with undulator kickmap data.
N_KICKS		long	1	Number of kicks into which to split the element.
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?
YAW_END		short	0	-1=Entrance, 0=Center, 1=Exit
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element provides simulation a generalized kick map, similar to the `UKICKMAP` but appropriate for maps that do not pertain to undulators or wigglers.

The input file has the following columns:

- `x` — Horizontal position in meters.

- `y` — Vertical position in meters.
- `xpFactor` — Dimensionless horizontal kick factor. The horizontal kick for any particle with a particular momentum deviation δ is the interpolated value of `xpFactor` divided by $1 + \delta$.
- `ypFactor` — Dimensionless horizontal kick factor. The vertical kick for any particle with a particular momentum deviation δ is the interpolated value of `ypFactor` divided by $1 + \delta$.

The values of `x` and `y` must be laid out on a grid of equispaced points. It is assumed that the data is ordered such that `x` varies fastest. This can be accomplished with the command

```
% sddssort -column=y,increasing -column=x,increasing input1.sdds input2.sdds
```

where `input1.sdds` is the original (unordered) file and `input2.sdds` is the new file, which would be used with `KICKMAP`.

This element is included in beam moments computations via the `moments_output` command.

The `YAW` and `YAW_END` parameters can be used in the simulation of canted IDs. Normally, steering magnets are used to create an angle between the devices. The devices are thus oriented in the reference coordinate system, meaning the beam tranverses the IDs at an angle. If it is desirable to align the IDs to the beam, the IDs can be yawed. A positive yaw will tilt the ID so that it is colinear with a beam that has been kicked by a positive horizontal steering angle. The `YAW_END` parameter defines which end of the ID is held fixed when the yaw is applied.

10.41 HISTOGRAM—Request for histograms of particle coordinates to be output to SDDS file.

Request for histograms of particle coordinates to be output to SDDS file.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
FILENAME		STRING		filename for histogram output, possibly incomplete (see below)
INTERVAL		long	1	interval in passes between output
START_PASS		long	0	starting pass for output
BINS		long	50	number of bins
FIXED_BIN_SIZE		short	0	if nonzero, bin size is fixed after the first histogram is made
X_DATA		short	1	histogram x and x'?
Y_DATA		short	1	histogram y and y'?
LONGIT_DATA		short	1	histogram t and p?
BIN_SIZE_FACTOR		double	1	multiply computed bin size by this factor before histogramming
NORMALIZE		short	1	normalize histogram with bin size and number of particles?
DISABLE		short	0	If nonzero, no output will be generated.
SPARSE		short	0	If nonzero, only bins with nonzero counts will be output.
START_PID		long	-1	starting particleID for particles to include
END_PID		long	-1	ending particleID for particles to include
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The output filename may be an incomplete filename. In the case of the `HISTOGRAM` point element, this means it may contain one instance of the string format specification "%s" and one occurence of an integer format specification (e.g., "%ld"). `elegant` will replace the format with the rootname (see `run_setup`) and the latter with the element's occurrence number. For example, suppose you had a repetitive lattice defined as follows:

```
H1: HISTOGRAM,FILENAME=''%s-%03ld.h1''
Q1: QUAD,L=0.1,K1=1
D: DRIFT,L=1
Q2: QUAD,L=0.1,K1=-1
CELL: LINE=(H1,Q1,D,2*Q2,D,Q1)
BL: LINE=(100*CELL)
```

The element `H1` appears 100 times. Each instance will result in a new file being produced. Successive instances have names like "*rootname*-001.h1", "*rootname*-002.h1", "*rootname*-003.h1", and so on up to "*rootname*-100.h1". (If instead of "%03ld" you used "%ld", the names would be "*rootname*-1.h1", "*rootname*-2.h1", etc. up to "*rootname*-100.h1". This is generally not as convenient as the names don't sort into occurrence order.)

The files can easily be plotted together, as in

```
% sddsplot -column=dt,dtFrequency *-???.h1 -separate
```

They may also be combined into a single file, as in

```
% sddscombine *-???.h1 all.h1
```

In passing, note that if `H1` was defined as

```
H1: HISTOGRAM,FILENAME=''%s.h1''
```

or

```
H1: HISTOGRAM,FILENAME=''output.h1''
```

only a single file would be produced, containing output from the last instance only.

10.42 HKICK—A horizontal steering dipole implemented as a matrix, up to 2nd order. Use EHKICK for symplectic tracking.

A horizontal steering dipole implemented as a matrix, up to 2nd order. Use EHKICK for symplectic tracking.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
KICK	RAD	double	0.0	kick strength
TILT	RAD	double	0.0	rotation about longitudinal axis
B2	$1/M^2$	double	0.0	normalized sextupole strength (kick = KICK*(1+B2*x$\hat{2}$) when y=0)
CALIBRATION		double	1	strength multiplier
EDGE_EFFECTS		short	0	include edge effects?
ORDER		short	0	matrix order
STEERING		short	1	use for steering?
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?
LERAD		double	0.0	if L=0, use this length for radiation computations
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.43 HKPOLY—Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length for geometry only, ignored in tracking
K00		double	0.0	Coefficient of polynomial for kicks—ignored
K01		double	0.0	Coefficient of polynomial for kicks
K02		double	0.0	Coefficient of polynomial for kicks
K03		double	0.0	Coefficient of polynomial for kicks
K04		double	0.0	Coefficient of polynomial for kicks
K05		double	0.0	Coefficient of polynomial for kicks
K06		double	0.0	Coefficient of polynomial for kicks
K10		double	0.0	Coefficient of polynomial for kicks
K11		double	0.0	Coefficient of polynomial for kicks
K12		double	0.0	Coefficient of polynomial for kicks
K13		double	0.0	Coefficient of polynomial for kicks
K14		double	0.0	Coefficient of polynomial for kicks
K15		double	0.0	Coefficient of polynomial for kicks
K16		double	0.0	Coefficient of polynomial for kicks
K20		double	0.0	Coefficient of polynomial for kicks
K21		double	0.0	Coefficient of polynomial for kicks
K22		double	0.0	Coefficient of polynomial for kicks

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
K23		double	0.0	Coefficient of polynomial for kicks
K24		double	0.0	Coefficient of polynomial for kicks
K25		double	0.0	Coefficient of polynomial for kicks
K26		double	0.0	Coefficient of polynomial for kicks
K30		double	0.0	Coefficient of polynomial for kicks
K31		double	0.0	Coefficient of polynomial for kicks
K32		double	0.0	Coefficient of polynomial for kicks
K33		double	0.0	Coefficient of polynomial for kicks
K34		double	0.0	Coefficient of polynomial for kicks
K35		double	0.0	Coefficient of polynomial for kicks
K36		double	0.0	Coefficient of polynomial for kicks
K40		double	0.0	Coefficient of polynomial for kicks
K41		double	0.0	Coefficient of polynomial for kicks
K42		double	0.0	Coefficient of polynomial for kicks
K43		double	0.0	Coefficient of polynomial for kicks
K44		double	0.0	Coefficient of polynomial for kicks
K45		double	0.0	Coefficient of polynomial for kicks
K46		double	0.0	Coefficient of polynomial for kicks

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
K50		double	0.0	Coefficient of polynomial for kicks
K51		double	0.0	Coefficient of polynomial for kicks
K52		double	0.0	Coefficient of polynomial for kicks
K53		double	0.0	Coefficient of polynomial for kicks
K54		double	0.0	Coefficient of polynomial for kicks
K55		double	0.0	Coefficient of polynomial for kicks
K56		double	0.0	Coefficient of polynomial for kicks
K60		double	0.0	Coefficient of polynomial for kicks
K61		double	0.0	Coefficient of polynomial for kicks
K62		double	0.0	Coefficient of polynomial for kicks
K63		double	0.0	Coefficient of polynomial for kicks
K64		double	0.0	Coefficient of polynomial for kicks
K65		double	0.0	Coefficient of polynomial for kicks
K66		double	0.0	Coefficient of polynomial for kicks
D00		double	0.0	Coefficient of polynomial for generalized drift—ignored
D01		double	0.0	Coefficient of polynomial for generalized drift
D02		double	0.0	Coefficient of polynomial for generalized drift
D03		double	0.0	Coefficient of polynomial for generalized drift

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
D04		double	0.0	Coefficient of polynomial for generalized drift
D05		double	0.0	Coefficient of polynomial for generalized drift
D06		double	0.0	Coefficient of polynomial for generalized drift
D10		double	0.0	Coefficient of polynomial for generalized drift
D11		double	0.0	Coefficient of polynomial for generalized drift
D12		double	0.0	Coefficient of polynomial for generalized drift
D13		double	0.0	Coefficient of polynomial for generalized drift
D14		double	0.0	Coefficient of polynomial for generalized drift
D15		double	0.0	Coefficient of polynomial for generalized drift
D16		double	0.0	Coefficient of polynomial for generalized drift
D20		double	0.0	Coefficient of polynomial for generalized drift
D21		double	0.0	Coefficient of polynomial for generalized drift
D22		double	0.0	Coefficient of polynomial for generalized drift
D23		double	0.0	Coefficient of polynomial for generalized drift
D24		double	0.0	Coefficient of polynomial for generalized drift
D25		double	0.0	Coefficient of polynomial for generalized drift
D26		double	0.0	Coefficient of polynomial for generalized drift
D30		double	0.0	Coefficient of polynomial for generalized drift

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
D31		double	0.0	Coefficient of polynomial for generalized drift
D32		double	0.0	Coefficient of polynomial for generalized drift
D33		double	0.0	Coefficient of polynomial for generalized drift
D34		double	0.0	Coefficient of polynomial for generalized drift
D35		double	0.0	Coefficient of polynomial for generalized drift
D36		double	0.0	Coefficient of polynomial for generalized drift
D40		double	0.0	Coefficient of polynomial for generalized drift
D41		double	0.0	Coefficient of polynomial for generalized drift
D42		double	0.0	Coefficient of polynomial for generalized drift
D43		double	0.0	Coefficient of polynomial for generalized drift
D44		double	0.0	Coefficient of polynomial for generalized drift
D45		double	0.0	Coefficient of polynomial for generalized drift
D46		double	0.0	Coefficient of polynomial for generalized drift
D50		double	0.0	Coefficient of polynomial for generalized drift
D51		double	0.0	Coefficient of polynomial for generalized drift
D52		double	0.0	Coefficient of polynomial for generalized drift
D53		double	0.0	Coefficient of polynomial for generalized drift
D54		double	0.0	Coefficient of polynomial for generalized drift

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
D55		double	0.0	Coefficient of polynomial for generalized drift
D56		double	0.0	Coefficient of polynomial for generalized drift
D60		double	0.0	Coefficient of polynomial for generalized drift
D61		double	0.0	Coefficient of polynomial for generalized drift
D62		double	0.0	Coefficient of polynomial for generalized drift
D63		double	0.0	Coefficient of polynomial for generalized drift
D64		double	0.0	Coefficient of polynomial for generalized drift
D65		double	0.0	Coefficient of polynomial for generalized drift
D66		double	0.0	Coefficient of polynomial for generalized drift
E000		double	0.0	Coefficient of polynomial for type 2 drifts
E001		double	0.0	Coefficient of polynomial for type 2 drifts
E002		double	0.0	Coefficient of polynomial for type 2 drifts
E003		double	0.0	Coefficient of polynomial for type 2 drifts
E004		double	0.0	Coefficient of polynomial for type 2 drifts
E005		double	0.0	Coefficient of polynomial for type 2 drifts
E006		double	0.0	Coefficient of polynomial for type 2 drifts
E010		double	0.0	Coefficient of polynomial for type 2 drifts
E011		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E012		double	0.0	Coefficient of polynomial for type 2 drifts
E013		double	0.0	Coefficient of polynomial for type 2 drifts
E014		double	0.0	Coefficient of polynomial for type 2 drifts
E015		double	0.0	Coefficient of polynomial for type 2 drifts
E016		double	0.0	Coefficient of polynomial for type 2 drifts
E020		double	0.0	Coefficient of polynomial for type 2 drifts
E021		double	0.0	Coefficient of polynomial for type 2 drifts
E022		double	0.0	Coefficient of polynomial for type 2 drifts
E023		double	0.0	Coefficient of polynomial for type 2 drifts
E024		double	0.0	Coefficient of polynomial for type 2 drifts
E025		double	0.0	Coefficient of polynomial for type 2 drifts
E026		double	0.0	Coefficient of polynomial for type 2 drifts
E030		double	0.0	Coefficient of polynomial for type 2 drifts
E031		double	0.0	Coefficient of polynomial for type 2 drifts
E032		double	0.0	Coefficient of polynomial for type 2 drifts
E033		double	0.0	Coefficient of polynomial for type 2 drifts
E034		double	0.0	Coefficient of polynomial for type 2 drifts
E035		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E036		double	0.0	Coefficient of polynomial for type 2 drifts
E040		double	0.0	Coefficient of polynomial for type 2 drifts
E041		double	0.0	Coefficient of polynomial for type 2 drifts
E042		double	0.0	Coefficient of polynomial for type 2 drifts
E043		double	0.0	Coefficient of polynomial for type 2 drifts
E044		double	0.0	Coefficient of polynomial for type 2 drifts
E045		double	0.0	Coefficient of polynomial for type 2 drifts
E046		double	0.0	Coefficient of polynomial for type 2 drifts
E050		double	0.0	Coefficient of polynomial for type 2 drifts
E051		double	0.0	Coefficient of polynomial for type 2 drifts
E052		double	0.0	Coefficient of polynomial for type 2 drifts
E053		double	0.0	Coefficient of polynomial for type 2 drifts
E054		double	0.0	Coefficient of polynomial for type 2 drifts
E055		double	0.0	Coefficient of polynomial for type 2 drifts
E056		double	0.0	Coefficient of polynomial for type 2 drifts
E060		double	0.0	Coefficient of polynomial for type 2 drifts
E061		double	0.0	Coefficient of polynomial for type 2 drifts
E062		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E063		double	0.0	Coefficient of polynomial for type 2 drifts
E064		double	0.0	Coefficient of polynomial for type 2 drifts
E065		double	0.0	Coefficient of polynomial for type 2 drifts
E066		double	0.0	Coefficient of polynomial for type 2 drifts
E100		double	0.0	Coefficient of polynomial for type 2 drifts
E101		double	0.0	Coefficient of polynomial for type 2 drifts
E102		double	0.0	Coefficient of polynomial for type 2 drifts
E103		double	0.0	Coefficient of polynomial for type 2 drifts
E104		double	0.0	Coefficient of polynomial for type 2 drifts
E105		double	0.0	Coefficient of polynomial for type 2 drifts
E106		double	0.0	Coefficient of polynomial for type 2 drifts
E110		double	0.0	Coefficient of polynomial for type 2 drifts
E111		double	0.0	Coefficient of polynomial for type 2 drifts
E112		double	0.0	Coefficient of polynomial for type 2 drifts
E113		double	0.0	Coefficient of polynomial for type 2 drifts
E114		double	0.0	Coefficient of polynomial for type 2 drifts
E115		double	0.0	Coefficient of polynomial for type 2 drifts
E116		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E120		double	0.0	Coefficient of polynomial for type 2 drifts
E121		double	0.0	Coefficient of polynomial for type 2 drifts
E122		double	0.0	Coefficient of polynomial for type 2 drifts
E123		double	0.0	Coefficient of polynomial for type 2 drifts
E124		double	0.0	Coefficient of polynomial for type 2 drifts
E125		double	0.0	Coefficient of polynomial for type 2 drifts
E126		double	0.0	Coefficient of polynomial for type 2 drifts
E130		double	0.0	Coefficient of polynomial for type 2 drifts
E131		double	0.0	Coefficient of polynomial for type 2 drifts
E132		double	0.0	Coefficient of polynomial for type 2 drifts
E133		double	0.0	Coefficient of polynomial for type 2 drifts
E134		double	0.0	Coefficient of polynomial for type 2 drifts
E135		double	0.0	Coefficient of polynomial for type 2 drifts
E136		double	0.0	Coefficient of polynomial for type 2 drifts
E140		double	0.0	Coefficient of polynomial for type 2 drifts
E141		double	0.0	Coefficient of polynomial for type 2 drifts
E142		double	0.0	Coefficient of polynomial for type 2 drifts
E143		double	0.0	Coefficient of polynomial for type 2 drifts

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E144		double	0.0	Coefficient of polynomial for type 2 drifts
E145		double	0.0	Coefficient of polynomial for type 2 drifts
E146		double	0.0	Coefficient of polynomial for type 2 drifts
E150		double	0.0	Coefficient of polynomial for type 2 drifts
E151		double	0.0	Coefficient of polynomial for type 2 drifts
E152		double	0.0	Coefficient of polynomial for type 2 drifts
E153		double	0.0	Coefficient of polynomial for type 2 drifts
E154		double	0.0	Coefficient of polynomial for type 2 drifts
E155		double	0.0	Coefficient of polynomial for type 2 drifts
E156		double	0.0	Coefficient of polynomial for type 2 drifts
E160		double	0.0	Coefficient of polynomial for type 2 drifts
E161		double	0.0	Coefficient of polynomial for type 2 drifts
E162		double	0.0	Coefficient of polynomial for type 2 drifts
E163		double	0.0	Coefficient of polynomial for type 2 drifts
E164		double	0.0	Coefficient of polynomial for type 2 drifts
E165		double	0.0	Coefficient of polynomial for type 2 drifts
E166		double	0.0	Coefficient of polynomial for type 2 drifts
E200		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E201		double	0.0	Coefficient of polynomial for type 2 drifts
E202		double	0.0	Coefficient of polynomial for type 2 drifts
E203		double	0.0	Coefficient of polynomial for type 2 drifts
E204		double	0.0	Coefficient of polynomial for type 2 drifts
E205		double	0.0	Coefficient of polynomial for type 2 drifts
E206		double	0.0	Coefficient of polynomial for type 2 drifts
E210		double	0.0	Coefficient of polynomial for type 2 drifts
E211		double	0.0	Coefficient of polynomial for type 2 drifts
E212		double	0.0	Coefficient of polynomial for type 2 drifts
E213		double	0.0	Coefficient of polynomial for type 2 drifts
E214		double	0.0	Coefficient of polynomial for type 2 drifts
E215		double	0.0	Coefficient of polynomial for type 2 drifts
E216		double	0.0	Coefficient of polynomial for type 2 drifts
E220		double	0.0	Coefficient of polynomial for type 2 drifts
E221		double	0.0	Coefficient of polynomial for type 2 drifts
E222		double	0.0	Coefficient of polynomial for type 2 drifts
E223		double	0.0	Coefficient of polynomial for type 2 drifts
E224		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E225		double	0.0	Coefficient of polynomial for type 2 drifts
E226		double	0.0	Coefficient of polynomial for type 2 drifts
E230		double	0.0	Coefficient of polynomial for type 2 drifts
E231		double	0.0	Coefficient of polynomial for type 2 drifts
E232		double	0.0	Coefficient of polynomial for type 2 drifts
E233		double	0.0	Coefficient of polynomial for type 2 drifts
E234		double	0.0	Coefficient of polynomial for type 2 drifts
E235		double	0.0	Coefficient of polynomial for type 2 drifts
E236		double	0.0	Coefficient of polynomial for type 2 drifts
E240		double	0.0	Coefficient of polynomial for type 2 drifts
E241		double	0.0	Coefficient of polynomial for type 2 drifts
E242		double	0.0	Coefficient of polynomial for type 2 drifts
E243		double	0.0	Coefficient of polynomial for type 2 drifts
E244		double	0.0	Coefficient of polynomial for type 2 drifts
E245		double	0.0	Coefficient of polynomial for type 2 drifts
E246		double	0.0	Coefficient of polynomial for type 2 drifts
E250		double	0.0	Coefficient of polynomial for type 2 drifts
E251		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E252		double	0.0	Coefficient of polynomial for type 2 drifts
E253		double	0.0	Coefficient of polynomial for type 2 drifts
E254		double	0.0	Coefficient of polynomial for type 2 drifts
E255		double	0.0	Coefficient of polynomial for type 2 drifts
E256		double	0.0	Coefficient of polynomial for type 2 drifts
E260		double	0.0	Coefficient of polynomial for type 2 drifts
E261		double	0.0	Coefficient of polynomial for type 2 drifts
E262		double	0.0	Coefficient of polynomial for type 2 drifts
E263		double	0.0	Coefficient of polynomial for type 2 drifts
E264		double	0.0	Coefficient of polynomial for type 2 drifts
E265		double	0.0	Coefficient of polynomial for type 2 drifts
E266		double	0.0	Coefficient of polynomial for type 2 drifts
E300		double	0.0	Coefficient of polynomial for type 2 drifts
E301		double	0.0	Coefficient of polynomial for type 2 drifts
E302		double	0.0	Coefficient of polynomial for type 2 drifts
E303		double	0.0	Coefficient of polynomial for type 2 drifts
E304		double	0.0	Coefficient of polynomial for type 2 drifts
E305		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E306		double	0.0	Coefficient of polynomial for type 2 drifts
E310		double	0.0	Coefficient of polynomial for type 2 drifts
E311		double	0.0	Coefficient of polynomial for type 2 drifts
E312		double	0.0	Coefficient of polynomial for type 2 drifts
E313		double	0.0	Coefficient of polynomial for type 2 drifts
E314		double	0.0	Coefficient of polynomial for type 2 drifts
E315		double	0.0	Coefficient of polynomial for type 2 drifts
E316		double	0.0	Coefficient of polynomial for type 2 drifts
E320		double	0.0	Coefficient of polynomial for type 2 drifts
E321		double	0.0	Coefficient of polynomial for type 2 drifts
E322		double	0.0	Coefficient of polynomial for type 2 drifts
E323		double	0.0	Coefficient of polynomial for type 2 drifts
E324		double	0.0	Coefficient of polynomial for type 2 drifts
E325		double	0.0	Coefficient of polynomial for type 2 drifts
E326		double	0.0	Coefficient of polynomial for type 2 drifts
E330		double	0.0	Coefficient of polynomial for type 2 drifts
E331		double	0.0	Coefficient of polynomial for type 2 drifts
E332		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E333		double	0.0	Coefficient of polynomial for type 2 drifts
E334		double	0.0	Coefficient of polynomial for type 2 drifts
E335		double	0.0	Coefficient of polynomial for type 2 drifts
E336		double	0.0	Coefficient of polynomial for type 2 drifts
E340		double	0.0	Coefficient of polynomial for type 2 drifts
E341		double	0.0	Coefficient of polynomial for type 2 drifts
E342		double	0.0	Coefficient of polynomial for type 2 drifts
E343		double	0.0	Coefficient of polynomial for type 2 drifts
E344		double	0.0	Coefficient of polynomial for type 2 drifts
E345		double	0.0	Coefficient of polynomial for type 2 drifts
E346		double	0.0	Coefficient of polynomial for type 2 drifts
E350		double	0.0	Coefficient of polynomial for type 2 drifts
E351		double	0.0	Coefficient of polynomial for type 2 drifts
E352		double	0.0	Coefficient of polynomial for type 2 drifts
E353		double	0.0	Coefficient of polynomial for type 2 drifts
E354		double	0.0	Coefficient of polynomial for type 2 drifts
E355		double	0.0	Coefficient of polynomial for type 2 drifts
E356		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E360		double	0.0	Coefficient of polynomial for type 2 drifts
E361		double	0.0	Coefficient of polynomial for type 2 drifts
E362		double	0.0	Coefficient of polynomial for type 2 drifts
E363		double	0.0	Coefficient of polynomial for type 2 drifts
E364		double	0.0	Coefficient of polynomial for type 2 drifts
E365		double	0.0	Coefficient of polynomial for type 2 drifts
E366		double	0.0	Coefficient of polynomial for type 2 drifts
E400		double	0.0	Coefficient of polynomial for type 2 drifts
E401		double	0.0	Coefficient of polynomial for type 2 drifts
E402		double	0.0	Coefficient of polynomial for type 2 drifts
E403		double	0.0	Coefficient of polynomial for type 2 drifts
E404		double	0.0	Coefficient of polynomial for type 2 drifts
E405		double	0.0	Coefficient of polynomial for type 2 drifts
E406		double	0.0	Coefficient of polynomial for type 2 drifts
E410		double	0.0	Coefficient of polynomial for type 2 drifts
E411		double	0.0	Coefficient of polynomial for type 2 drifts
E412		double	0.0	Coefficient of polynomial for type 2 drifts
E413		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E414		double	0.0	Coefficient of polynomial for type 2 drifts
E415		double	0.0	Coefficient of polynomial for type 2 drifts
E416		double	0.0	Coefficient of polynomial for type 2 drifts
E420		double	0.0	Coefficient of polynomial for type 2 drifts
E421		double	0.0	Coefficient of polynomial for type 2 drifts
E422		double	0.0	Coefficient of polynomial for type 2 drifts
E423		double	0.0	Coefficient of polynomial for type 2 drifts
E424		double	0.0	Coefficient of polynomial for type 2 drifts
E425		double	0.0	Coefficient of polynomial for type 2 drifts
E426		double	0.0	Coefficient of polynomial for type 2 drifts
E430		double	0.0	Coefficient of polynomial for type 2 drifts
E431		double	0.0	Coefficient of polynomial for type 2 drifts
E432		double	0.0	Coefficient of polynomial for type 2 drifts
E433		double	0.0	Coefficient of polynomial for type 2 drifts
E434		double	0.0	Coefficient of polynomial for type 2 drifts
E435		double	0.0	Coefficient of polynomial for type 2 drifts
E436		double	0.0	Coefficient of polynomial for type 2 drifts
E440		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E441		double	0.0	Coefficient of polynomial for type 2 drifts
E442		double	0.0	Coefficient of polynomial for type 2 drifts
E443		double	0.0	Coefficient of polynomial for type 2 drifts
E444		double	0.0	Coefficient of polynomial for type 2 drifts
E445		double	0.0	Coefficient of polynomial for type 2 drifts
E446		double	0.0	Coefficient of polynomial for type 2 drifts
E450		double	0.0	Coefficient of polynomial for type 2 drifts
E451		double	0.0	Coefficient of polynomial for type 2 drifts
E452		double	0.0	Coefficient of polynomial for type 2 drifts
E453		double	0.0	Coefficient of polynomial for type 2 drifts
E454		double	0.0	Coefficient of polynomial for type 2 drifts
E455		double	0.0	Coefficient of polynomial for type 2 drifts
E456		double	0.0	Coefficient of polynomial for type 2 drifts
E460		double	0.0	Coefficient of polynomial for type 2 drifts
E461		double	0.0	Coefficient of polynomial for type 2 drifts
E462		double	0.0	Coefficient of polynomial for type 2 drifts
E463		double	0.0	Coefficient of polynomial for type 2 drifts
E464		double	0.0	Coefficient of polynomial for type 2 drifts

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E465		double	0.0	Coefficient of polynomial for type 2 drifts
E466		double	0.0	Coefficient of polynomial for type 2 drifts
E500		double	0.0	Coefficient of polynomial for type 2 drifts
E501		double	0.0	Coefficient of polynomial for type 2 drifts
E502		double	0.0	Coefficient of polynomial for type 2 drifts
E503		double	0.0	Coefficient of polynomial for type 2 drifts
E504		double	0.0	Coefficient of polynomial for type 2 drifts
E505		double	0.0	Coefficient of polynomial for type 2 drifts
E506		double	0.0	Coefficient of polynomial for type 2 drifts
E510		double	0.0	Coefficient of polynomial for type 2 drifts
E511		double	0.0	Coefficient of polynomial for type 2 drifts
E512		double	0.0	Coefficient of polynomial for type 2 drifts
E513		double	0.0	Coefficient of polynomial for type 2 drifts
E514		double	0.0	Coefficient of polynomial for type 2 drifts
E515		double	0.0	Coefficient of polynomial for type 2 drifts
E516		double	0.0	Coefficient of polynomial for type 2 drifts
E520		double	0.0	Coefficient of polynomial for type 2 drifts
E521		double	0.0	Coefficient of polynomial for type 2 drifts

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E522		double	0.0	Coefficient of polynomial for type 2 drifts
E523		double	0.0	Coefficient of polynomial for type 2 drifts
E524		double	0.0	Coefficient of polynomial for type 2 drifts
E525		double	0.0	Coefficient of polynomial for type 2 drifts
E526		double	0.0	Coefficient of polynomial for type 2 drifts
E530		double	0.0	Coefficient of polynomial for type 2 drifts
E531		double	0.0	Coefficient of polynomial for type 2 drifts
E532		double	0.0	Coefficient of polynomial for type 2 drifts
E533		double	0.0	Coefficient of polynomial for type 2 drifts
E534		double	0.0	Coefficient of polynomial for type 2 drifts
E535		double	0.0	Coefficient of polynomial for type 2 drifts
E536		double	0.0	Coefficient of polynomial for type 2 drifts
E540		double	0.0	Coefficient of polynomial for type 2 drifts
E541		double	0.0	Coefficient of polynomial for type 2 drifts
E542		double	0.0	Coefficient of polynomial for type 2 drifts
E543		double	0.0	Coefficient of polynomial for type 2 drifts
E544		double	0.0	Coefficient of polynomial for type 2 drifts
E545		double	0.0	Coefficient of polynomial for type 2 drifts

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E546		double	0.0	Coefficient of polynomial for type 2 drifts
E550		double	0.0	Coefficient of polynomial for type 2 drifts
E551		double	0.0	Coefficient of polynomial for type 2 drifts
E552		double	0.0	Coefficient of polynomial for type 2 drifts
E553		double	0.0	Coefficient of polynomial for type 2 drifts
E554		double	0.0	Coefficient of polynomial for type 2 drifts
E555		double	0.0	Coefficient of polynomial for type 2 drifts
E556		double	0.0	Coefficient of polynomial for type 2 drifts
E560		double	0.0	Coefficient of polynomial for type 2 drifts
E561		double	0.0	Coefficient of polynomial for type 2 drifts
E562		double	0.0	Coefficient of polynomial for type 2 drifts
E563		double	0.0	Coefficient of polynomial for type 2 drifts
E564		double	0.0	Coefficient of polynomial for type 2 drifts
E565		double	0.0	Coefficient of polynomial for type 2 drifts
E566		double	0.0	Coefficient of polynomial for type 2 drifts
E600		double	0.0	Coefficient of polynomial for type 2 drifts
E601		double	0.0	Coefficient of polynomial for type 2 drifts
E602		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E603		double	0.0	Coefficient of polynomial for type 2 drifts
E604		double	0.0	Coefficient of polynomial for type 2 drifts
E605		double	0.0	Coefficient of polynomial for type 2 drifts
E606		double	0.0	Coefficient of polynomial for type 2 drifts
E610		double	0.0	Coefficient of polynomial for type 2 drifts
E611		double	0.0	Coefficient of polynomial for type 2 drifts
E612		double	0.0	Coefficient of polynomial for type 2 drifts
E613		double	0.0	Coefficient of polynomial for type 2 drifts
E614		double	0.0	Coefficient of polynomial for type 2 drifts
E615		double	0.0	Coefficient of polynomial for type 2 drifts
E616		double	0.0	Coefficient of polynomial for type 2 drifts
E620		double	0.0	Coefficient of polynomial for type 2 drifts
E621		double	0.0	Coefficient of polynomial for type 2 drifts
E622		double	0.0	Coefficient of polynomial for type 2 drifts
E623		double	0.0	Coefficient of polynomial for type 2 drifts
E624		double	0.0	Coefficient of polynomial for type 2 drifts
E625		double	0.0	Coefficient of polynomial for type 2 drifts
E626		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E630		double	0.0	Coefficient of polynomial for type 2 drifts
E631		double	0.0	Coefficient of polynomial for type 2 drifts
E632		double	0.0	Coefficient of polynomial for type 2 drifts
E633		double	0.0	Coefficient of polynomial for type 2 drifts
E634		double	0.0	Coefficient of polynomial for type 2 drifts
E635		double	0.0	Coefficient of polynomial for type 2 drifts
E636		double	0.0	Coefficient of polynomial for type 2 drifts
E640		double	0.0	Coefficient of polynomial for type 2 drifts
E641		double	0.0	Coefficient of polynomial for type 2 drifts
E642		double	0.0	Coefficient of polynomial for type 2 drifts
E643		double	0.0	Coefficient of polynomial for type 2 drifts
E644		double	0.0	Coefficient of polynomial for type 2 drifts
E645		double	0.0	Coefficient of polynomial for type 2 drifts
E646		double	0.0	Coefficient of polynomial for type 2 drifts
E650		double	0.0	Coefficient of polynomial for type 2 drifts
E651		double	0.0	Coefficient of polynomial for type 2 drifts
E652		double	0.0	Coefficient of polynomial for type 2 drifts
E653		double	0.0	Coefficient of polynomial for type 2 drifts

HKPOLY continued

Applies kick according to a Hamiltonian that's a polynomial function of x and y together with a generalized drift also given as a polynomial of qx and qy

Parameter Name	Units	Type	Default	Description
E654		double	0.0	Coefficient of polynomial for type 2 drifts
E655		double	0.0	Coefficient of polynomial for type 2 drifts
E656		double	0.0	Coefficient of polynomial for type 2 drifts
E660		double	0.0	Coefficient of polynomial for type 2 drifts
E661		double	0.0	Coefficient of polynomial for type 2 drifts
E662		double	0.0	Coefficient of polynomial for type 2 drifts
E663		double	0.0	Coefficient of polynomial for type 2 drifts
E664		double	0.0	Coefficient of polynomial for type 2 drifts
E665		double	0.0	Coefficient of polynomial for type 2 drifts
E666		double	0.0	Coefficient of polynomial for type 2 drifts
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FACTOR		double	1	additional factor to apply
N_REPEATS		long	1	Number of times to repeat the drift-kick-drift sequence. Strength of each application is reduced by this factor.
DRIFT_TYPE		short	1	If 1, then use D[i][j]. If 2, then use E[i][j][k].
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element imposes kicks on the beam according to a Hamiltonian that is a polynomial function of x and y

$$H_k \Delta s = \sum_{i=0}^{6} \sum_{j=0}^{6} K_{ij} x^i y^j \tag{49}$$

where K_{00} is ignored. The changes to the momenta are determined via Hamilton's equations, e.g.,

$$\Delta q_x = -\frac{\partial H_k \Delta s}{\partial x} \tag{50}$$

It also implements a generalized drift that is described by another Hamiltonian

$$H_d \Delta s = (1+\delta) \sum_{i=0}^{6} \sum_{j=0}^{6} D_{ij} \left(\frac{q_x}{1+\delta} \right)^i \left(\frac{q_y}{1+\delta} \right)^j \tag{51}$$

where D_{00} is ignored. Again, the changes to the positions are determined via Hamilton's equations, e.g.,

$$\Delta x = \frac{\partial H_d \Delta s}{\partial q_x} \tag{52}$$

In version 2019.1.0, another option was added for the drift Hamiltonian. This is activated by setting the paramter `DRIFT_TYPE` to 2 (the default is 1) and setting the `E` values instead of the `D` values. In this case, the δ dependence is under user control

$$H_d \Delta s = \sum_{i=0}^{6} \sum_{j=0}^{6} \sum_{k=0}^{6} E_{ijk} q_x^i q_y^j \delta^k \tag{53}$$

where E_{000} is ignored.

In more detail, the drift Hamiltonian is applied on both sides of the kick Hamiltonian, but with half strength.

For example, a quadrupole of length L with integrated strength $K_1 L$ could be specified by setting $K_{20} = -K_{02} = K_1 L/2$ and $D_2 0 = D02 = L/2$. A sextupole with integrated strength $K_2 L$ could be specified by setting $K_{30} = K_2 L/6$ and $K_{12} = K_2 L/2$ and $D_2 0 = D02 = L/2$. The purpose, however, is not to simulate such elements, since they can be more conveniently simulated with `KQUAD` or `KSEXT`. It is rather to simulate elements that may not be described by the usual multipoles.

10.44 HMON—A horizontal position monitor, accepting a rpn equation for the readout as a function of the actual position (x).

A horizontal position monitor, accepting a rpn equation for the readout as a function of the actual position (x).
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
WEIGHT		double	1	weight in correction
TILT		double	0.0	rotation about longitudinal axis
CALIBRATION		double	1	calibration factor for readout
SETPOINT	M	double	0.0	steering setpoint
ORDER		short	0	matrix order
READOUT		STRING	NULL	rpn expression for readout (actual position supplied in variable x)
CO_FITPOINT		short	0	If nonzero, then closed orbit value is placed in variable <name>#<occurence>.xco
STORE_TURN_BY_TURN		short	0	If nonzero, then turn-by-turn horizontal position readout and number of particles are placed in variables <name>#<occurence>.x/n.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

If `STORE_TURN_BY_TURN` is nonzero, then the computed horizontal BPM reading is stored in *ElementName#ElementOccurence*.x during tracking. This can be used in the `expression` in `modulate_elements` to create position-triggered changes to elements.

10.45 IBSCATTER—A simulation of intra-beam scattering.

A simulation of intra-beam scattering.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
FACTOR		double	1	factor by which to multiply growth rates before using
DO_X		short	1	do x-plane scattering?
DO_Y		short	1	do y-plane scattering?
DO_Z		short	1	do z-plane scattering?
SMOOTH		short	1	Use smooth method instead of random numbers?
FORCE_MATCHED_TWISS		short	0	Force computations to be done with twiss parameters of the beamline, not the beam.
ISRING		short	1	Is it storage ring?
NSLICE		long	1	The number of slices per bunch
INTERVAL		long	1	Interval in passes at which to update output file.
FILENAME		STRING	NULL	Output filename.
BUNCHED_BEAM_MODE		short	1	If non-zero, then do calculations bunch-by-bunch.
PARALLEL_INTEGRATION		short	1	If non-zero, then uses parallel method for integration in Pelegant.
VERBOSE		short	0	If non-zero, then print updates during calculations.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is used for simulating intra-beam scattering (IBS) effect. The IBS algorithm is based on the Bjorken and Mtingwa's [15] formula, and with an extension of including vertical dispersion. It can be used for both storage ring and Linac.

To initialize IBS calculation, one or more IBSCATTER elements must be inserted into the beamline. `elegant` calculates the integrated IBS growth rates between IBSCATTERs (or from beginning of the beamline to the first IBSCATTER), then scatter particles at each IBSCATTER element. Beam's parameters are updated for use in downstream elements.

This method requires that IBSCATTER can not be installed at the beginning of beamline. The number of other elements between IBSCATTERs or from the beginning of beamline to the first IBSCATTER has to be 2 or more. For storage ring, an IBSCATTER must be installed at the end of beamline.

Because the IBS growth rates are energy dependent, special caution is needed for calculations with accelerating beam. The user needs to split their accelerating cavity into several pieces, so that γ has no large changes between elements.

The user can examine the calculation through an optional SDDS output file - *filename*. The file has a multiple page structure. Each slice at pass i at each IBSCATTER element occupies one page. Each page contains integrated IBS growth rates between IBSCATTERs (or from beginning of the beamline to first IBSCATTER) as parameters, and local rates for elements in between as tabular data.

10.46 ILMATRIX—An Individualized Linear Matrix for each particle for fast symplectic tracking with chromatic and amplitude-dependent effects

An Individualized Linear Matrix for each particle for fast symplectic tracking with chromatic and amplitude-dependent effects
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	Length (used for position and time-of-flight computation)
NUX		double	0.0	Horizontal tune
NUY		double	0.0	Vertical tune
NUX1M		double	0.0	First chromatic derivative of the horizontal tune
NUY1M		double	0.0	First chromatic derivative of the vertical tune
NUX2M		double	0.0	Second chromatic derivative of the horizontal tune
NUY2M		double	0.0	Second chromatic derivative of the vertical tune
NUX3M		double	0.0	Third chromatic derivative of the horizontal tune
NUY3M		double	0.0	Third chromatic derivative of the vertical tune
NUX1AX	$1/M$	double	0.0	First amplitude derivative of the horizontal tune wrt Ax
NUY1AX	$1/M$	double	0.0	First amplitude derivative of the vertical tune wrt Ax
NUX1AY	$1/M$	double	0.0	First amplitude derivative of the horizontal tune wrt Ay
NUY1AY	$1/M$	double	0.0	First amplitude derivative of the vertical tune wrt Ay
NUX2AX	$1/M^2$	double	0.0	Second amplitude derivative of the horizontal tune wrt Ax
NUY2AX	$1/M^2$	double	0.0	Second amplitude derivative of the vertical tune wrt Ax
NUX2AY	$1/M^2$	double	0.0	Second amplitude derivative of the horizontal tune wrt Ay
NUY2AY	$1/M^2$	double	0.0	Second amplitude derivative of the vertical tune wrt Ay

`ILMATRIX` continued

An Individualized Linear Matrix for each particle for fast symplectic tracking with chromatic and amplitude-dependent effects

Parameter Name	Units	Type	Default	Description
NUX1AX1AY	$1/M^2$	double	0.0	Amplitude derivative of the horizontal tune wrt Ax and Ay
NUY1AX1AY	$1/M^2$	double	0.0	Amplitude derivative of the vertical tune wrt Ax and Ay
BETAX	M	double	0.0	On-momentum horizontal beta function
BETAY	M	double	0.0	On-momentum vertical beta function
BETAX1M	M	double	0.0	First chromatic derivative of horizontal beta function
BETAY1M	M	double	0.0	First chromatic derivative of vertical beta function
ALPHAX		double	0.0	On-momentum horizontal alpha function
ALPHAY		double	0.0	On-momentum vertical alpha function
ALPHAX1M		double	0.0	First chromatic derivative of horizontal alpha function
ALPHAY1M		double	0.0	First chromatic derivative of vertical alpha function
ETAX	M	double	0.0	On-momentum horizontal eta function
ETAPX		double	0.0	On-momentum horizontal eta' function
ETAY	M	double	0.0	On-momentum vertical eta function
ETAPY		double	0.0	On-momentum vertical eta' function
ETAX1	M	double	0.0	First chromatic derivative of horizontal eta function
ETAPX1		double	0.0	First chromatic derivative of horizontal eta' function
ETAY1	M	double	0.0	First chromatic derivative of vertical eta function
ETAPY1		double	0.0	First chromatic derivative of vertical eta' function

ILMATRIX continued

An Individualized Linear Matrix for each particle for fast symplectic tracking with chromatic and amplitude-dependent effects

Parameter Name	Units	Type	Default	Description
ALPHAC		double	0.0	First-order momentum compaction factor
ALPHAC2		double	0.0	Second-order momentum compaction factor
ALPHAC3		double	0.0	Third-order momentum compaction factor
DS1AX		double	0.0	First amplitude derivative of the path length wrt Ax
DS1AY		double	0.0	First amplitude derivative of the path length wrt Ay
DS2AX	$1/M$	double	0.0	Second amplitude derivative of the path length wrt Ax
DS2AY	$1/M$	double	0.0	Second amplitude derivative of the path length wrt Ay
DS1AX1AY	$1/M$	double	0.0	Amplitude derivative of the path length wrt Ax and Ay
TILT	RAD	double	0.0	Rotation angle about the longitudinal axis.
CROSS_RESONANCE		short	0	If zero, then particles that cross an integer or half-integer resonance are considered lost.
VERBOSITY		short	0	If nonzero, then information about particle losses is printed out.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element allows fast, symplectic tracking of transport through a periodic cell with chromatic and amplitude-dependent tunes, beta functions, and dispersion. This is done by computing a linear matrix for every particle using Twiss parameters, tunes, dispersion, etc., supplied by the user. The user can also supply selected chromatic and amplitude derivatives of these quantities, which are used to compute the individual particle's beta functions, tune, dispersion, etc., which in turn allows computing the individual particle's linear matrix.

The starting point is the well-known expression for the one-turn linear matrix in terms of the lattice functions

$$R_q = \begin{pmatrix} \cos 2\pi\nu_q + \alpha_q \sin 2\pi\nu_q & \beta_q \sin 2\pi\nu_q \\ -\gamma_q \sin 2\pi\nu_q & \cos 2\pi\nu_q - \alpha_q \sin 2\pi\nu_q \end{pmatrix} \tag{54}$$

where ν_q is the tune in the q plane. We can expand the quantities in the matrix using

$$\nu_q = \nu_{q,0} + \sum_{n=1}^{3} \left(\frac{\partial^n \nu_q}{\partial \delta^n}\right)_0 \frac{\delta^n}{n!} + \sum_{n=1}^{2} \left(\frac{\partial^n \nu_q}{\partial A_x^n}\right)_0 \frac{A_x^n}{n!} + \sum_{n=1}^{2} \left(\frac{\partial^n \nu_q}{\partial A_y^n}\right)_0 \frac{A_y^n}{n!} + \left(\frac{\partial^2 \nu_q}{\partial A_x \partial A_y}\right)_0 A_x A_y \quad (55)$$

where $\delta = (p - p_0)/p_0$ is the fractional momentum offset, $A_q = (q_\beta^2 + (\alpha_q q_\beta + \beta_q q'_\beta)^2)/\beta_q$ is the betatron amplitude, and the betatron coordinates are computed using

$$q_\beta = q - \delta \left(\eta_q + \left(\frac{\partial \eta_q}{\partial \delta}\right)_0 \delta\right) \quad (56)$$

and

$$q'_\beta = q' - \delta \left(\eta'_q + \left(\frac{\partial \eta'_q}{\partial \delta}\right)_0 \delta\right) \quad (57)$$

At each turn, δ, A_x, and A_y are computed for each particle. The user-supplied values of the various derivatives are then used to compute the tunes for each particle. Similar expansions are used to compute the other lattice functions. This allows computing the 2x2 transfer matrices for the betatron coordinates in the x and planes, then advancing the betatron coordinates one turn, after which the full coordinates are recomputed by adding back the momentum-dependent closed orbit.

The pathlength is computed using the expansion

$$\Delta s = L \sum_{n=1}^{3} \alpha_{c,n} \delta^n + \sum_{n=1}^{4} R_{5n} x_{\beta,n} + \sum_{n=1}^{2} \left(\frac{\partial^n s}{\partial A_x^n}\right)_0 \frac{A_x^n}{n!} + \sum_{n=1}^{2} \left(\frac{\partial^n s}{\partial A_y^n}\right)_0 \frac{A_y^n}{n!} + \left(\frac{\partial^2 s}{\partial A_x \partial A_y}\right)_0 A_x A_y \quad (58)$$

where $\alpha_{c,1}$ is the linear momentum compaction factor. Note that in keeping with convention the higher-order momentum compaction is expressed by polynomial coefficients, not derivatives. The terms dependent on betatron amplitude are expressed in terms of the more typical derivatives. Note the difference between the R_{5n} terms (added in version 2019.4) and those dependent on $A_{x,y}$: the former are oscillatory while the latter will accumulate. The `frequency_map` command can be used to compute path-length dependence on betatron amplitude.

Using this element is very similar to using the `setup_linear_chromatic_tracking` command. The advantage is that using `LMATRIX`, one can split a ring into segments and place, for example, impedance elements between the segments.

This element was inspired by requests from Y. Chae (APS).

N.B.: There is a bug related to using `ILMATRIX` that will result in a crash if one does not request computation of the twiss parameters. If you encounter this problem, just add the following statement after the `run_setup` command:

```
&twiss_output
        matched = 1
&end
```

10.47 IONEFFECTS—Simulates ionization of residual gas and interaction with the beam.

Simulates ionization of residual gas and interaction with the beam.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
DISABLE		long	0	If non-zero, turn off ion effects in the region covered by this element.
MACRO_IONS		long	0	If positive, overrides the default value set in the ion_effects command, giving the number of macro ions generated per bunch passage.
GENERATION_INTERVAL		long	0	If positive, overrides the default value set in the ion_effects command, giving the number of macro ions generated per bunch passage.
X_SPAN		double	0.0	If positive, gives the region over which ions are kept.
Y_SPAN		double	0.0	If positive, gives the region over which ions are kept.
NX_POISSON		long	0	If positive, the number of x bins for Poisson solver for ion fields.
NY_POISSON		long	0	If positive, the number of y bins for Poisson solver for ion fields.
X_BIN_DIVISOR		double	0.0	If positive, gives the ratio of electron beam sigma to bin size for ion field calculation.
Y_BIN_DIVISOR		double	0.0	If positive, gives the ratio of electron beam sigma to bin size for ion field calculation.
X_RANGE_MULTIPLIER		double	0.0	If positive, gives the ratio of ion binning region size to ion 80% x range.
Y_RANGE_MULTIPLIER		double	0.0	If positive, gives the ratio of ion binning region size to ion 80% y range.

Simulates ionization of residual gas and interaction with the beam.

Parameter Name	Units	Type	Default	Description
X_SIGMA_LIMIT_MULTIPLIER		double	0.0	If positive, gives lower limit on bi-gaussian fit sigma values in units of the ion bin size.
Y_SIGMA_LIMIT_MULTIPLIER		double	0.0	If positive, gives lower limit on bi-gaussian fit sigma values in units of the ion bin size.
STARTPASS		long	0	If positive, gives the pass on which ion effects start.
ENDPASS		long	-1	If positive, gives the pass on which ion effects end.
PASSINTERVAL		long	1	Interval between ion effects modeling.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

NB: This element is new and considered experimental. Please report issues back to the developers.

This element provides serial or parallel simulation of the interaction of residual gas ions with the electron beam. It must be used in concert with the `ion_effects` command, described in 7.34.

Modeling of residual ions has these features:

- s-dependent gas pressure profiles for any number of species.
- Arbitrary ion species, specified by a user-provided file that includes the cross sections.
- User-defined locations for ion generation. Each `IONEFFECTS` element represents the ions present in a segment of the accelerator. The segments start and end half way between successive `IONEFFECTS` elements. The segment for the first element starts at the beginning of the ring, and the last element covers the end of the ring.
- Arbitrary fill patterns. Uniform fills can be set up using the `bunched_beam` command, while custom fills can be set up by generating the beam externally and using the `sdds_beam` command.
- Multiple ionization of trapped ions. For example, a CO+ ion could multiply ionize into CO++, or dissociate into C+.

Some limitations of the model include:

- Fields from electron bunches are computed based on gaussian parameters, which is a reasonably good approximation.

- By default, fields from ions are computed based on gaussian parameters, which is often a somewhat poor approximation. Alternatively, a bi-gaussian form may be used, which uses a sum of two gaussians. This is a much better approximation to the typical distribution, which often has a hot core and long tails. One can also use a tri-gaussian, bi-lorentzian, or tri-lorentzian fit.
- Ions move only transversely and exist only outside of magnets.

Performing ion simulations involves the following steps

1. Prepare file describing the ion properties, as described in 7.34. Each ion is generated by either a source gas or source ion.
2. Prepare file giving gas pressure vs s for the source gases described in the ion properties file.
3. Insert `IONEFFECTS` elements in the lattice. This can be performed using the `insert_elements` command (described in 7.32), or manually by editing the lattice file.
4. Insert `ion_effects` command after the `run_setup` command. See 7.34 for syntax. Note that certain properties of the individual `IONEFFECTS` elements can override the global settings given by in the `ion_effects` command.
5. Generate a bunched beam, using either the `bunched_beam` command or providing an externally-generated beam to the `sdds_beam` command. Section 6 gives more information about bunched beams in `elegant`.

For each bunch passage, the `IONEFFECTS` element does the following:

1. Advance existing ions during bunch gap
2. Eliminate ions that are outside of given boundaries
3. Generate ions
4. Apply kick from beam to ions
5. Apply kick from ions to beam

The line density of ions generated by a single bunch in a single pass is:

$$\lambda_{ion} = \sigma_{ion} \frac{P}{k_B T} N_b \tag{59}$$

where σ_{ion} is the ionization cross section, P is the pressure, k_B is the Boltzmann constant, T is the temperature, and N_b is the bunch population.

The resulting macroparticle charge is:

$$Q_{macro} = \frac{10^{-22} e}{7.5 \times 10^{-3} k_B} \frac{\sigma_{ion} P N_b L_{eff}}{n_{macro} T} \tag{60}$$

Here σ_{ion} has units of Mb, P has units of Torr, $k_B = 1.38 \times 10^{-23}$ J/K, e is the electron charge, L_{eff} is the effective length of the ion element (in m), and n_{macro} is the number of macroparticles generated. The initial ion distribution follows the bunch distribution (assumed to be Gaussian).

The IONEFFECTS element also supports multiple ionization. In the ion_properties file, one can define the SourceName for a given IonName to be another ion. In this case, each macro-ion of type SourceName has a chance of being multiply ionized into type IonName. The calculation is done every multiple_ionization_interval bunch passes. The probability of multiple ionization depends on the cross section and local beam density.

The kick on the ions from the beam is calculated using the Basetti-Erskine formula [52], which assumes the beam is Gaussian in both transverse dimensions. By default, this assumption is also used for the ion distribution. The parameter gaussian_ion_range gives the range (in beam sigma) over which ions are counted, for calculating the ion-beam kicks.

The Gaussian method may be a poor assumption for the ions, in which case the field_calculation_method parameter can be set to bigaussian, which uses a sum of two gaussians. This provides a much better model for the actual distribution, at the expense of a considerable increase in run time. One can also use a trigaussian fit, as well as a bilorentzian or trilorentzian. The ion_bin_divisor and ion_range_multiplier parameters can be used to control the bin size and range, respectively, of the histogram used to approximate the ion charge distribution. The ion_bin_divisor gives the ratio of the rms size of the electron bunch in the plane in question to the bin size.

The ion_range_multiplier parameter is used to determine the range of the histogram. If positive, a rough histogram of the ion distribution (with ten times the desired bin size) is used to estimate the range required to encompass 80% of the ions; half this value is multiplied by the absolute value of ion_range_multiplier to get the half range of the full histogram; a value of 1.5 is is suggested. If zero, the histogram encompasses all of the ions, which may result in a sparse histogram when a few ions have large coordinates. If negative gives the range of the binned coordinates in units of the rms size of the ion distribution.

The ion_histogram_output parameter and related parameters can be used to request output of the ion distribution and the multi-function fit, which is advisable when setting the binning parameters.

The change in momentum of an ion due to the bunch passage is:

$$\Delta p_y + i\Delta p_x = \frac{cN_b r_e m_e}{\gamma}\sqrt{\frac{2\pi}{\sigma_x^2-\sigma_y^2}}\left[w\left(\frac{x+iy}{\sqrt{2(\sigma_x^2-\sigma_y^2)}}\right) - exp\left(\frac{-x^2}{2\sigma_x^2}-\frac{y^2}{2\sigma_y^2}\right)w\left(\frac{\frac{\sigma_y}{\sigma_x}x+i\frac{\sigma_x}{\sigma_y}y}{\sqrt{2(\sigma_x^2-\sigma_y^2)}}\right)\right] \tag{61}$$

where c is the speed of light, N_b is the bunch population, r_e is the classical electron radius (2.82×10^{-15} m), m_e is the electron mass, γ is the relativistic factor (~ 1 for the ions), $\sigma_{x,y}$ are the horizontal and vertical beam sizes, w is the complex error function, and x and y are the distance from the ion to the bunch center.

10.48 KICKER—A combined horizontal-vertical steering magnet implemented as a matrix, up to 2nd order. For time-dependent kickers, see BUMPER.

A combined horizontal-vertical steering magnet implemented as a matrix, up to 2nd order. For time-dependent kickers, see BUMPER.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
HKICK	RAD	double	0.0	x kick angle
VKICK	RAD	double	0.0	y kick angle
TILT	RAD	double	0.0	rotation about longitudinal axis
B2	$1/M^2$	double	0.0	normalized sextupole strength (e.g., kick = KICK*(1+B2*x$\hat{2}$))
HCALIBRATION		double	1	factor applied to obtain x kick
VCALIBRATION		double	1	factor applied to obtain y kick
EDGE_EFFECTS		long	0	include edge effects?
ORDER		long	0	matrix order
STEERING		long	1	use for steering?
SYNCH_RAD		long	0	include classical, single-particle synchrotron radiation?
ISR		long	0	include incoherent synchrotron radiation (quantum excitation)?
LERAD		double	0.0	if L=0, use this length for radiation computations
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.49 KOCT—A canonical kick octupole.

A canonical kick octupole.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K3	$1/M^4$	double	0.0	geometric strength
TILT	RAD	double	0.0	rotation about longitudinal axis
PITCH	RAD	double	0.0	rotation about horizontal axis. Ignored if MALIGN_METHOD=0
YAW	RAD	double	0.0	rotation about vertical axis. Ignored if MALIGN_METHOD=0
BORE	M	double	0.0	bore radius
B	T	double	0.0	field at pole tip (used if bore nonzero)
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE		double	0.0	fractional strength error
N_KICKS		long	4	number of kicks (rounded up to next multipole of 4 if INTEGRATION_ORDER=4). Deprecated. Use N_SLICES.
N_SLICES		long	4	Number of slices (full integrator steps).
SYSTEMATIC_MULTIPOLES		STRING	NULL	input file for systematic multipoles
RANDOM_MULTIPOLES		STRING	NULL	input file for random multipoles
INTEGRATION_ORDER		short	4	integration order (2, 4, or 6)
SQRT_ORDER		short	0	Ignored, kept for backward compatibility only.
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?

A canonical kick octupole.

Parameter Name	Units	Type	Default	Description
ISR1PART		short	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
EXPAND_HAMILTONIAN		short	0	If 1, Hamiltonian is expanded to leading order.
MALIGN_METHOD		short	0	0=original, 1=new entrace-centered, 2=new body-centered
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.50 KPOLY—A thin kick element with polynomial dependence on the coordinates in one plane.

A thin kick element with polynomial dependence on the coordinates in one plane.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
COEFFICIENT	M^{-ORDER}	double	0.0	coefficient of polynomial
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FACTOR		double	1	additional factor to apply
ORDER		long	0	order of polynomial
PLANE		STRING	x	plane to kick (x, y)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.51 KQUAD—A canonical kick quadrupole.

A canonical kick quadrupole.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K1	$1/M^2$	double	0.0	geometric strength
TILT	RAD	double	0.0	rotation about longitudinal axis
PITCH	RAD	double	0.0	rotation about horizontal axis. Ignored if MALIGN_METHOD=0
YAW	RAD	double	0.0	rotation about vertical axis. Ignored if MALIGN_METHOD=0.
BORE	M	double	0.0	bore radius
B	T	double	0.0	pole tip field (used if bore nonzero)
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE		double	0.0	fractional strength error
N_KICKS		long	0	number of kicks (rounded up to next multipole of 4 if INTEGRATION_ORDER=4). Deprecated. Use N_SLICES.
N_SLICES		long	1	Number of slices (full integrator steps).
HKICK	RAD	double	0.0	horizontal correction kick
VKICK	RAD	double	0.0	vertical correction kick
HCALIBRATION		double	1	calibration factor for horizontal correction kick
VCALIBRATION		double	1	calibration factor for vertical correction kick
HSTEERING		short	0	use for horizontal correction?
VSTEERING		short	0	use for vertical correction?
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?

KQUAD continued

A canonical kick quadrupole.

Parameter Name	Units	Type	Default	Description
SYSTEMATIC_MULTIPOLES		STRING	NULL	input file for systematic multipoles
EDGE_MULTIPOLES		STRING	NULL	input file for systematic edge multipoles
RANDOM_MULTIPOLES		STRING	NULL	input file for random multipoles
STEERING_MULTIPOLES		STRING	NULL	input file for multipole content of steering kicks
SYSTEMATIC_MULTIPOLE_FACTOR		double	1	Factor by which to multiply systematic and edge multipoles
RANDOM_MULTIPOLE_FACTOR		double	1	Factor by which to multiply random multipoles
STEERING_MULTIPOLE_FACTOR		double	1	Factor by which to multiply steering multipoles
MIN_NORMAL_ORDER		short	-1	If nonnegative, minimum order of systematic and random normal multipoles to use from data files.
MIN_SKEW_ORDER		short	-1	If nonnegative, minimum order of systematic and random skew multipoles to use from data files.
MAX_NORMAL_ORDER		short	-1	If nonnegative, maximum order of systematic and random normal multipoles to use from data files.
MAX_SKEW_ORDER		short	-1	If nonnegative, maximum order of systematic and random skew multipoles to use from data files.
INTEGRATION_ORDER		short	4	integration order (2, 4, or 6)
SQRT_ORDER		short	0	Ignored, kept for backward compatibility only.
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?

A canonical kick quadrupole.

Parameter Name	Units	Type	Default	Description
ISR1PART		short	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
SR_IN_ORDINARY_MATRIX		short	0	If nonzero, the (tracking-based) matrix used for routine computations includes classical synchrotron radiation if SYNCH_RAD=1.
EDGE1_EFFECTS		short	0	include entrance edge effects?
EDGE2_EFFECTS		short	0	include exit edge effects?
LEFFECTIVE	M	double	0.0	Effective length. Ignored if non-positive.
I0P	M	double	0.0	i0+ fringe integral
I1P	M^2	double	0.0	i1+ fringe integral
I2P	M^3	double	0.0	i2+ fringe integral
I3P	M^4	double	0.0	i3+ fringe integral
LAMBDA2P	M^3	double	0.0	lambda2+ fringe integral
I0M	M	double	0.0	i0- fringe integral
I1M	M^2	double	0.0	i1- fringe integral
I2M	M^3	double	0.0	i2- fringe integral
I3M	M^4	double	0.0	i3- fringe integral
LAMBDA2M	M^3	double	0.0	lambda2- fringe integral
EDGE1_LINEAR		short	1	Use to selectively turn off linear part if EDGE1_EFFECTS nonzero.
EDGE2_LINEAR		short	1	Use to selectively turn off linear part if EDGE2_EFFECTS nonzero.
EDGE1_NONLINEAR_FACTOR		double	1	Use to selectively scale non-linear entrance edge effects if EDGE1_EFFECTS>1
EDGE2_NONLINEAR_FACTOR		double	1	Use to selectively scale non-linear exit edge effects if EDGE2_EFFECTS>1
RADIAL		short	0	If non-zero, converts the quadrupole into a radially-focusing lens

A canonical kick quadrupole.

Parameter Name	Units	Type	Default	Description
EXPAND_HAMILTONIAN		short	0	If 1, Hamiltonian is expanded to leading order.
TRACKING_MATRIX		short	0	If nonzero, gives order of tracking-based matrix up to third order to be used for twiss parameters etc. If zero, 2nd-order analytical matrix is used.
MALIGN_METHOD		short	0	0=original, 1=new entrace-centered, 2=new body-centered
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a quadrupole using a kick method based on symplectic integration. The user specifies the number of kicks and the order of the integration. For computation of twiss parameters and response matrices, this element is treated like a standard thick-lens quadrupole; i.e., the number of kicks and the integration order become irrelevant.

Multipole errors

Specification of systematic and random multipole errors is supported through the `SYSTEMATIC_MULTIPOLES`, `EDGE_MULTIPOLES`, and `RANDOM_MULTIPOLES` fields. These specify, respectively, fixed multipole strengths for the body of the element, fixed multipole strengths for the edges of the element, and random multipole strengths for the body of the element. These fields give the names of SDDS files that supply the multipole data. The files are expected to contain a single page of data with the following elements:

1. Floating point parameter `referenceRadius` giving the reference radius for the multipole data.
2. An integer column named `order` giving the order of the multipole. The order is defined as $(N_{poles} - 2)/2$, so a quadrupole has order 1, a sextupole has order 2, and so on.
3. Floating point columns `normal` and `skew` giving the values for the normal and skew multipole strengths, respectively. (N.B.: previous versions used the names `an` and `bn`, respectively. This is still accepted but deprecated) These are defined as a fraction of the main field strength measured at the reference radius, R: $f_n = \frac{K_n R^n/n!}{K_m R^m/m!}$, where $m = 1$ is the order of the main field and n is the order of the error multipole. A similar relationship holds for the skew multipole fractional strengths. For random multipoles, the values are interpreted as rms values for the distribution.

Specification of systematic higher multipoles due to steering fields is supported through the STEERING_MULTIPOLES field. This field gives the name of an SDDS file that supplies the multipole data. The file is expected to contain a single page of data with the following elements:

1. Floating point parameter referenceRadius giving the reference radius for the multipole data.
2. An integer column named order giving the order of the multipole. The order is defined as $(N_{poles} - 2)/2$. The order must be an even number because of the quadrupole symmetry.
3. Floating point column normal giving the values for the normal multipole strengths, which are driven by the horizontal steering field. (N.B.: previous versions used the name an for this data. This is still accepted but deprecated) normal is specifies the multipole strength as a fraction f_n of the steering field strength measured at the reference radius, R: $f_n = \frac{K_n R^n/n!}{K_m R^m/m!}$, where $m = 0$ is the order of the steering field and n is the order of the error multipole. The skew values (for vertical steering) are deduced from the normal values, specifically, $g_n = f_n * (-1)^{n/2}$.

The dominant systematic multipole term in the steering field is a sextupole. Note that elegant presently *does not* include such sextupole contributions in the computation of the chromaticity via the twiss_output command. However, these chromatic effects will be seen in tracking.

Apertures

Apertures specified via an upstream MAXAMP element or an aperture_input command will be imposed inside this element.

Length specificiation

As of version 29.2, this element incorporates the ability to have different values for the insertion and effective lengths. This is invoked when LEFFECTIVE is positive. In this case, the L parameter is understood to be the physical insertion length. Using LEFFECTIVE is a convenient way to incorporate the fact that the effective length may differ from the physical length and even vary with excitation, without having to modify the drift spaces on either side of the quadrupole element.

Fringe effects

Fringe field effects are based on publications of D. Zhuo *et al.* [34] and J. Irwin *et al.* [35], as well as unpublished work of C. X. Wang (ANL). The fringe field is characterized by 10 integrals given in equations 19, 20, and 21 of [34]. However, the values input into elegant should be normalized by K_1 or K_1^2, as appropriate.

For the exit-side fringe field, let s_1 be the center of the magnet, s_0 be the location of the nominal end of the magnet (for a hard-edge model), and let s_2 be a point well outside the magnet. Using $K_{1,he}(s)$ to represent the hard edge model and $K_1(s)$ the actual field profile, we define the normalized difference as $\tilde{k}(s) = (K_1(s) - K_{1,he}(s))/K_1(s_1)$. (Thus, $\tilde{k}(s) = \tilde{K}(s)/K_0$, using the notation of Zhou *et al.*)

The integrals to be input to elegant are defined as

$$i_0^- = \int_{s_1}^{s_0} \tilde{k}(s)ds \qquad i_0^+ = \int_{s_0}^{s_2} \tilde{k}(s)ds \tag{62}$$

$$i_1^- = \int_{s_1}^{s_0} \tilde{k}(s)(s - s_0)ds \qquad i_1^+ = \int_{s_0}^{s_2} \tilde{k}(s)(s - s_0)ds \tag{63}$$

$$i_2^- = \int_{s_1}^{s_0} \tilde{k}(s)(s - s_0)^2ds \qquad i_2^+ = \int_{s_0}^{s_2} \tilde{k}(s)(s - s_0)^2ds \tag{64}$$

$$i_3^- = \int_{s_1}^{s_0} \tilde{k}(s)(s - s_0)^3ds \qquad i_3^+ = \int_{s_0}^{s_2} \tilde{k}(s)(s - s_0)^3ds \tag{65}$$

$$\lambda_2^- = \int_{s_1}^{s_0} ds \int_{s}^{s_0} ds'\tilde{k}(s)\tilde{k}(s')(s' - s) \qquad \lambda_2^+ = \int_{s_0}^{s_2} ds \int_{s}^{s_2} ds'\tilde{k}(s)\tilde{k}(s')(s' - s) \tag{66}$$

Normally, the effects are dominated by i_1^- and i_1^+. The script `computeQuadFringeIntegrals`, packaged with `elegant`, allows computing these integrals and the effective length if provided with data giving the gradient vs s.

The `EDGE1_EFFECTS` and `EDGE2_EFFECTS` parameters can be used to turn fringe field effects on and off, but also to control the order of the implementation. If the value is 1, linear fringe effects are included. If the value is 2, leading-order (cubic) nonlinear effects are included. If the value is 3 or higher, higher order effects are included.

Misalignments

There are three modes for implementing alignment errors. Which is used is controlled by the value of the `MALIGN_METHOD` parameter:

- `MALIGN_METHOD=0` — This selects the original method, which was the only one available before version 2021.1. The misalignment is referenced to the entrance face. The `YAW` and `PITCH` parameters are ignored.
- `MALIGN_METHOD=1` — This selects a linearized method based on M. Venturini's work [58], with misalignment referenced to the entrance face. The `YAW` and `PITCH` parameters are implemented.
- `MALIGN_METHOD=2` — This selects a linearized method based on M. Venturini's work [58], with misalignment referenced to the magnet center. The `YAW` and `PITCH` parameters are implemented.
- `MALIGN_METHOD=3` — This selects an exact method based on M. Venturini's work [58], with misalignment referenced to the entrance face. The `YAW` and `PITCH` parameters are implemented.
- `MALIGN_METHOD=4` — This selects an exact method based on M. Venturini's work [58], with misalignment referenced to the magnet center. The `YAW` and `PITCH` parameters are implemented.

For elements with non-zero `TILT`, error displacements and rotations are performed in the lab frame.

Radiation effects

If `SYNCH_RAD` is non-zero, classical synchrotron radiation is included in tracking. If, in addition, `ISR` is non-zero, incoherent synchrotron radiation (i.e., quantum-mechanical variation in radiation emitted by different particles) is also included. (To exclude ISR for single-particle tracking, set `ISR1PART=0`.)

If `TRACKING_MATRIX` and `SYNCH_RAD` are non-zero, classical synchrotron radiation can be included in the ordinary matrix (e.g., for `twiss_output` and `matrix_output`) by setting `SR_IN_ORDINARY_MATRIX` to a non-zero value. Symplecticity is not assured, but the results may be interesting nonetheless. A more rigorous approach is to use `moments_output`. `SR_IN_ORDINARY_MATRIX` does not affect tracking.

10.52 KQUSE—A canonical kick element combining quadrupole and sextupole fields.

A canonical kick element combining quadrupole and sextupole fields.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K1	$1/M^2$	double	0.0	geometric quadrupole strength
K2	$1/M^3$	double	0.0	geometric sextupole strength
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE1	M	double	0.0	fractional strength error for K1
FSE2	M	double	0.0	fractional strength error for K2
N_KICKS		long	0	number of kicks. Deprecated. Use N_SLICES.
N_SLICES		long	1	Number of slices (full integrator steps).
INTEGRATION_ORDER		short	4	integration order (2, 4, or 6)
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?
ISR1PART		short	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
MATRIX_TRACKING		short	0	For testing only.
EXPAND_HAMILTONIAN		short	0	If 1, Hamiltonian is expanded to leading order.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.53 KSEXT—A canonical kick sextupole, which differs from the MULT element with ORDER=2 in that it can be used for chromaticity correction.

A canonical kick sextupole, which differs from the MULT element with ORDER=2 in that it can be used for chromaticity correction.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K2	$1/M^3$	double	0.0	geometric strength
K1	$1/M^2$	double	0.0	geometric quadrupole strength error. See notes below!
J1	$1/M^2$	double	0.0	geometric skew quadrupole strength error. See notes below!
TILT	RAD	double	0.0	rotation about longitudinal axis
PITCH	RAD	double	0.0	rotation about horizontal axis. Ignored if MALIGN_METHOD=0
YAW	RAD	double	0.0	rotation about vertical axis. Ignored if MALIGN_METHOD=0
BORE	M	double	0.0	bore radius
B	T	double	0.0	field at pole tip (used if bore nonzero)
N_KICKS		long	0	number of kicks (rounded up to next multipole of 4 if INTEGRATION_ORDER=4. Deprecated. Use N_SLICES.)
N_SLICES		long	1	Number of slices (full integrator steps).
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE		double	0.0	fractional strength error
HKICK	RAD	double	0.0	horizontal correction kick
VKICK	RAD	double	0.0	vertical correction kick
HCALIBRATION		double	1	calibration factor for horizontal correction kick
VCALIBRATION		double	1	calibration factor for vertical correction kick
HSTEERING		short	0	use for horizontal correction?

KSEXT continued

A canonical kick sextupole, which differs from the MULT element with ORDER=2 in that it can be used for chromaticity correction.

Parameter Name	Units	Type	Default	Description
VSTEERING		short	0	use for vertical correction?
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
SYSTEMATIC_MULTIPOLES		STRING	NULL	input file for systematic multipoles
EDGE_MULTIPOLES		STRING	NULL	input file for systematic edge multipoles
RANDOM_MULTIPOLES		STRING	NULL	input file for random multipoles
STEERING_MULTIPOLES		STRING	NULL	input file for multipole content of steering kicks
SYSTEMATIC_MULTIPOLE_FACTOR		double	1	Factor by which to multiply systematic and edge multipoles
RANDOM_MULTIPOLE_FACTOR		double	1	Factor by which to multiply random multipoles
STEERING_MULTIPOLE_FACTOR		double	1	Factor by which to multiply steering multipoles
MIN_NORMAL_ORDER		short	-1	If nonnegative, minimum order of systematic and random normal multipoles to use from data files.
MIN_SKEW_ORDER		short	-1	If nonnegative, minimum order of systematic and random skew multipoles to use from data files.
MAX_NORMAL_ORDER		short	-1	If nonnegative, maximum order of systematic and random normal multipoles to use from data files.
MAX_SKEW_ORDER		short	-1	If nonnegative, maximum order of systematic and random skew multipoles to use from data files.

A canonical kick sextupole, which differs from the MULT element with ORDER=2 in that it can be used for chromaticity correction.

Parameter Name	Units	Type	Default	Description
INTEGRATION_ORDER		short	4	integration order (2, 4, or 6)
SQRT_ORDER		short	0	Ignored, kept for backward compatibility only.
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?
ISR1PART		short	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
EXPAND_HAMILTONIAN		short	0	If 1, Hamiltonian is expanded to leading order.
MALIGN_METHOD		short	0	0=original, 1=new entrace-centered, 2=new body-centered
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a sextupole using a kick method based on symplectic integration. The user specifies the number of kicks and the order of the integration. For computation of twiss parameters, chromaticities, and response matrices, this element is treated like a standard thick-lens sextuupole; i.e., the number of kicks and the integration order become irrelevant.

Multipole errors

Specification of systematic and random multipole errors is supported through the `SYSTEMATIC_MULTIPOLES`, `EDGE_MULTIPOLES`, and `RANDOM_MULTIPOLES` fields. These specify, respectively, fixed multipole strengths for the body of the element, fixed multipole strengths for the edges of the element, and random multipole strengths for the body of the element. These fields give the names of SDDS files that supply the multipole data. The files are expected to contain a single page of data with the following elements:

1. Floating point parameter `referenceRadius` giving the reference radius for the multipole data.
2. An integer column named `order` giving the order of the multipole. The order is defined as $(N_{poles} - 2)/2$, so a quadrupole has order 1, a sextupole has order 2, and so on.
3. Floating point columns `normal` and `skew` giving the values for the normal and skew multipole strengths, respectively. (N.B.: previous versions used the names `an` and `bn`, respectively. This is still accepted but deprecated) These are defined as a fraction of the main field strength measured at the reference radius, R: $f_n = \frac{K_n R^n/n!}{K_m R^m/m!}$, where $m = 2$ is the order of the main field

and n is the order of the error multipole. A similar relationship holds for the skew multipole fractional strengths. For random multipoles, the values are interpreted as rms values for the distribution.

Specification of systematic higher multipoles due to steering fields is supported through the STEERING_MULTIPOLES field. This field gives the name of an SDDS file that supplies the multipole data. The file is expected to contain a single page of data with the following elements:

1. Floating point parameter `referenceRadius` giving the reference radius for the multipole data.
2. An integer column named `order` giving the order of the multipole. The order is defined as $(N_{poles} - 2)/2$. The order must be an even number because of the quadrupole symmetry.
3. Floating point column `normal` giving the values for the normal multipole strengths, which are driven by the horizontal steering field. (N.B.: previous versions used the name `an` for this data. This is still accepted but deprecated) `normal` is specifies the multipole strength as a fraction f_n of the steering field strength measured at the reference radius, R: $f_n = \frac{K_n R^n/n!}{K_m R^m/m!}$, where $m = 0$ is the order of the steering field and n is the order of the error multipole. The skew values (for vertical steering) are deduced from the `normal` values, specifically, $g_n = f_n * (-1)^{n/2}$.

Another way of introducing errors is via the K1 and J1 parameters, which allow introducing a normal and skew quadrupole **error** term. For tracking, the strength of these values can be arbitrarily high without introducing errors. However, the matrix analysis (e.g., for determination of tunes and beta functions) assumes that these are weak effects and high accuracy should not be expected if this is not true. If K1 is significant, then use of the KQUSE element is preferred.

Apertures

Apertures specified via an upstream MAXAMP element or an `aperture_input` command will be imposed inside this element.

Misalignments

There are three modes for implementing alignment errors. Which is used is controlled by the value of the MALIGN_METHOD parameter:

- MALIGN_METHOD=0 — This selects the original method, which was the only one available before version 2021.1. The misalignment is referenced to the entrance face. The YAW and PITCH parameters are ignored.
- MALIGN_METHOD=1 — This selects a linearized method based on M. Venturini's work [58], with misalignment referenced to the entrance face. The YAW and PITCH parameters are implemented.
- MALIGN_METHOD=2 — This selects a linearized method based on M. Venturini's work [58], with misalignment referenced to the magnet center. The YAW and PITCH parameters are implemented.
- MALIGN_METHOD=3 — This selects an exact method based on M. Venturini's work [58], with misalignment referenced to the entrance face. The YAW and PITCH parameters are implemented.
- MALIGN_METHOD=4 — This selects an exact method based on M. Venturini's work [58], with misalignment referenced to the magnet center. The YAW and PITCH parameters are implemented.

For elements with non-zero TILT, error displacements and rotations are performed in the lab frame.

10.54 LGBEND—A multi-segment straight longitudinal dipole magnet

A multi-segment straight longitudinal dipole magnet
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	arc length (ignored)
CONFIGURATION	*NULL*	STRING	NULL	configuration file from straightDipoleFringeCalc
APERTURE_DATA	*NULL*	STRING	NULL	aperture data vs distance traveled from start (aperture_data format)
TILT	*RAD*	double	0.0	rotation about incoming longitudinal axis
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
DZ	*M*	double	0.0	misalignment
EYAW	*M*	double	0.0	misalignment
EPITCH	*M*	double	0.0	misalignment
ETILT	*RAD*	double	0.0	error rotation about incoming longitudinal axis
FSE		double	0.0	fractional strength error
N_SLICES		long	4	Number of slices (full integrator steps) per segment.
INTEGRATION_ORDER		short	4	integration order (2, 4, or 6)
EDGE_ORDER		short	2	Gives order of edge effects.
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?
ISR1PART		short	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
SR_IN_ORDINARY_MATRIX		short	0	If nonzero, the (tracking-based) matrix used for routine computations includes classical synchrotron radiation if SYNCH_RAD=1.

A multi-segment straight longitudinal dipole magnet

Parameter Name	Units	Type	Default	Description
USE_RAD_DIST		short	0	If nonzero, overrides SYNCH_RAD and ISR, causing simulation of radiation from distributions, optionally including opening angle.
ADD_OPENING_ANGLE		short	1	If nonzero, radiation opening angle effects are added if USE_RAD_DIST is nonzero.
OPTIMIZE_FSE		short	1	Optimize strength (FSE) of first and last segments to obtain the ideal deflection angle and final trajectory.
COMPENSATE_KN		short	1	If nonzero, K1 and K2 strengths are adjusted to compensate for the changes in FSE needed to center the trajectory.
VERBOSE		short	0	If nonzero, print messages showing optimized FSE and x offset.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element provides a symplectic straight-pole, multi-segment bending magnet with the exact Hamiltonian in Cartesian coordinates. The quadrupole, sextupole, and other multipole terms are defined in Cartesian coordinates. The element is restricted to having rectangular ends for each segment. It is, in essence, like a series of `CCBEND` [61] elements concatenated into a whole.

The `LGBEND` element has relatively few explicit parameters, giving the illusion of simplicity. A custom configuration file is used to specify the many parameters of an `LGBEND`. This file is generated using the companion program `straightDipoleFringeCalc` from a generalized gradient expansion (GGE). The GGE can be created using either `computeCBGGE` (for cylindrical-boundary data) or `computeRBGGE` for (rectangular-boundary data). There is an example in the `elegant` examples collection.

One issue with `LGBEND`, as with `CCBEND` and to a lesser degree `CSBEND`, is that the final reference trajectory is not guaranteed to be on axis. To address this, but default `LGBEND` will automatically adjust the strength of the first and last segments to minimize the x and x' coordinates of the reference particle. (This can be defeated by setting `OPTIMIZE_FSE=0`.) In doing this, the K_1 and

K_2 values of those segments are by default automatically scaled to ensure that the integrated quadrupole and sextupole are not chagned. (This can be defeated by setting COMPENSATE_KN=0.)

Radiation effects

If SYNCH_RAD is non-zero, classical synchrotron radiation is included in tracking. Incoherent synchrotron radiation, when requested with ISR=1, normally uses gaussian distributions for the excitation of the electrons. (To exclude ISR for single-particle tracking, set ISR1PART=0.) Setting USE_RAD_DIST=1 invokes a more sophisticated algorithm that uses correct statistics for the photon energy and number distributions. In addition, if USE_RAD_DIST=1 one may also set ADD_OPENING_ANGLE=1, which includes the photon angular distribution when computing the effect on the emitting electron.

If SYNCH_RAD and SR_IN_ORDINARY_MATRIX are non-zero, classical synchrotron radiation will be included in the ordinary matrix (e.g., for twiss_output and matrix_output). Symplecticity is not assured, but the results may be interesting nonetheless. A more rigorous approach is to use moments_output. SR_IN_ORDINARY_MATRIX does not affect tracking.

Adding errors

Misalignments are performed in the body-centered frame using Venturini's method [58] based on the values provided for DX, DY, DZ, ETILT, EPITCH, and EYAW. The FSE parameter is used to impart a global fractional strength error, which affects not only the dipole field but also any quadrupole or sextupole terms. The TILT parameter is used to specify the design orientation of the magnet.

Matrix generation

elegant will use tracking to determine the transport matrix for LGBEND elements, which is needed for computation of twiss parameters and other operations. This can require some time, so elegant will cache the matrices and re-use them for identical elements. Still, there is a performance benefit to be had from using the parallel version, particularly when assignment of errors prevents sharing of results among many elements.

10.55 LMIRROR—A mirror for light optics

A mirror for light optics
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
RX	*M*	double	0.0	radius in horizontal plane
RY	*M*	double	0.0	radius in vertical plane
THETA	*RAD*	double	0.0	angle of incidence (in horizontal plane)
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
DZ	*M*	double	0.0	misalignment
TILT	*RAD*	double	0.0	misalignment rotation about longitudinal axis
YAW	*RAD*	double	0.0	misalignment rotation about vertical axis
PITCH	*RAD*	double	0.0	misalignment rotation about transverse horizontal axis
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.56 LRWAKE—Long-range (inter-bunch and inter-turn) longitudinal and transverse wake

Long-range (inter-bunch and inter-turn) longitudinal and transverse wake
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
INPUTFILE		STRING	NULL	name of file giving Green function
TCOLUMN		STRING	NULL	column in INPUTFILE containing time data
WXCOLUMN		STRING	NULL	column in INPUTFILE containing horizontal dipole Green function
WYCOLUMN		STRING	NULL	column in INPUTFILE containing vertical dipole Green function
WZCOLUMN		STRING	NULL	column in INPUTFILE containing longitudinal Green function
QXCOLUMN		STRING	NULL	column in INPUTFILE containing horizontal quadrupole Green function
QYCOLUMN		STRING	NULL	column in INPUTFILE containing vertical quadrupole Green function
FACTOR		double	1	factor by which to multiply wakes
XFACTOR		double	1	factor by which to multiply horizontal dipole wake
YFACTOR		double	1	factor by which to multiply vertical dipole wake
ZFACTOR		double	1	factor by which to multiply longitudinal dipole wake
QXFACTOR		double	1	factor by which to multiply horizontal quadrupole wake
QYFACTOR		double	1	factor by which to multiply vertical quadrupole wake
TURNS_TO_KEEP		long	128	number of turns of data to retain
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the wake to full strength.

Long-range (inter-bunch and inter-turn) longitudinal and transverse wake

Parameter Name	Units	Type	Default	Description
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element provides serial and parallel modeling of long range, multi-bunch, multi-pass, non-resonant wakes. Resonant wakes can be modeled using the *RFMODE elements, while short-range wakes are modeled with WAKE, TRWAKE, ZLONGIT, ZTRANSVERSE, and RFCW.

For the LRWAKE element, the beam is assumed to be bunched and wakes are computed bunch-to-bunch. The long-range wake is assumed to be constant within any single bunch.

To use this element, the beam has to be prepared in a special way so that elegant can recognize which particles belong to which bunches. See Section 6 for details. Given a properly prepared beam, the algorithm works as follows.

- Each processor uses arrays to record
  - How many particles are in each of B bunches,
  - The sum of the arrival times t at the LRWAKE element for the particles in each bunch, and
  - The sum of x and y at the LRWAKE element for the particles in each bunch.
- These arrays are summed across all the processors and used to compute the moments $\langle t \rangle$, $\langle x \rangle$, and $\langle y \rangle$ for each bunch, as well as the charge in each bunch.
- Arrays of length B from N prior turns are kept in a buffer
  - Buffer for turns $N-1$ to 1 is copied to slots N through 2, thus overwriting the data for the oldest turn.
  - The data for latest turn is copied into slot 1.
- For each bunch, sums are performed over all prior bunches/turns to compute the voltage. For the longitudinal wake, we have

$$V_z(b) = \sum_{i=b}^{N*B} q_i W_z(\langle t_b \rangle - \langle t_i \rangle). \tag{67}$$

  A positive value *decelerates* the particle. For the horizontal dipole wake we have

$$V_x(b) = \sum_{i=b}^{N*B} q_i \langle x_i \rangle W_x(\langle t_b \rangle - \langle t_i \rangle), \tag{68}$$

  with the vertical wake being similar. In both cases, a positive value deflects the particle toward positive x or y for a positive offset of the driving particle.

- The quadrupole wakes may also be included. In this case, the contribution to the horizontal wake is

$$V_x(b) = \sum_{i=b}^{N*B} q_i x_p W_x(\langle t_b \rangle - \langle t_i \rangle), \tag{69}$$

where x_p is the coordinate of the probe particle. The vertical wake is similar.

To use LRWAKE, the user provides the wakes (functions of t) in an SDDS file. These wakes may extend over an arbitrary number of turns, with the user declaring how many turns to actually use as part of the element definition. However, they should be zero within the region occupied by a single bunch, to avoid double-counting with the true short-range wake. (Note that the above sums include the self-wake.) Similarly, the short-range should be zero for times comparable to the bunch spacing.

Note that the quadrupole wakes are in some cases related to the dipole wakes by constant numerical factors [48]. In such a case, one may name the same column for `QXCOLUMN` (`QYCOLUMN`) and `WXCOLUMN` (`WYCOLUMN`) and then specify `QXFACTOR` (`QYFACTOR`) appropriately.

10.57 LSCDRIFT—Longitudinal space charge impedance

Longitudinal space charge impedance
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
LEFFECTIVE	M	double	0.0	effective length (used if L=0)
BINS		long	0	number of bins for current histogram
SMOOTHING		short	0	Use Savitzky-Golay filter to smooth current histogram?
SG_HALFWIDTH		short	1	Savitzky-Golay filter half-width for smoothing current histogram
SG_ORDER		short	1	Savitzky-Golay filter order for smoothing current histogram
INTERPOLATE		short	1	Interpolate wake?
LSC		short	1	Include longitudinal space-charge impedance? If zero, acts like ordinary drift.
AUTO_LEFFECTIVE		short	0	In nonzero and if L=0, the LEFFECTIVE parameter is set to the length of the previous element.
LOW_FREQUENCY_CUTOFF0		double	-1	Highest spatial frequency at which low-frequency cutoff filter is zero. If not positive, no low-frequency cutoff filter is applied. Frequency is in units of Nyquist (0.5/binsize).
LOW_FREQUENCY_CUTOFF1		double	-1	Lowest spatial frequency at which low-frequency cutoff filter is 1. If not given, defaults to LOW_FREQUENCY_CUTOFF1.
HIGH_FREQUENCY_CUTOFF0		double	-1	Spatial frequency at which smoothing filter begins. If not positive, no frequency filter smoothing is done. Frequency is in units of Nyquist (0.5/binsize).

Longitudinal space charge impedance

Parameter Name	Units	Type	Default	Description
HIGH_FREQUENCY_CUTOFF1		double	-1	Spatial frequency at which smoothing filter is 0. If not given, defaults to HIGH_FREQUENCY_CUTOFF0.
RADIUS_FACTOR		double	1.7	LSC radius is (Sx+Sy)/2*RADIUS_FACTOR
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates longitudinal space charge in a drift space using the method described in [22]. This is based on the longitudinal space charge impedance per unit length

$$Z_{lsc}(k) = \frac{iZ_0}{\pi k r_b^2}\left[1 - \frac{kr_b}{\gamma}K_1\left(\frac{kr_b}{\gamma}\right)\right] \tag{70}$$

If `L` is 0 and `LEFFECTIVE` is not, then the element provides a LSC kick with impedance given by $Z_{lsc}L_{effective}$. This can be used to insert an LSC kick that integrates the longitudinal space charge effect of a section of a lattice. This should be used only for cases where there is very little relative longitudinal motion of particles.

Two simple filters are provided.

- The parameters `HIGH_FREQUENCY_CUTOFF0` and `HIGH_FREQUENCY_CUTOFF1` are used to filter out high frequencies, i.e., they provide a low-pass or noise filter. The filter has value 1 for $f <$ `HIGH_FREQUENCY_CUTOFF0`, value 0 for $f >$ `HIGH_FREQUENCY_CUTOFF1`, and linear variation in between.
- The parameters `LOW_FREQUENCY_CUTOFF0` and `LOW_FREQUENCY_CUTOFF1` are used to filter out low frequencies, i.e., they provide a high-pass filter. The filter has value 0 for $f <$ `LOW_FREQUENCY_CUTOFF0`, value 1 for $f >$ `LOW_FREQUENCY_CUTOFF1`, and linear variation in between.

]

10.58 LSRMDLTR—A non-symplectic numerically integrated planar undulator including optional co-propagating laser beam for laser modulation of the electron beam.

A non-symplectic numerically integrated planar undulator including optional co-propagating laser beam for laser modulation of the electron beam.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
BU	T	double	0.0	Undulator peak field
TGU_GRADIENT	$1/M$	double	0.0	Transverse gradient divided by maximum on-axis field.
TGU_COMP_FACTOR	$NULL$	double	1	Use to adjust constant field component to reduce trajectory error.
PERIODS		long	0	Number of undulator periods.
METHOD	$NULL$	STRING	non-adaptive runge-kutta	integration method (runge-kutta, bulirsch-stoer, modified-midpoint, two-pass modified-midpoint, leap-frog, non-adaptive runge-kutta)
FIELD_EXPANSION	$NULL$	STRING	leading terms	ideal, exact, or "leading terms"
ACCURACY	$NULL$	double	0.0	Integration accuracy for adaptive integration. (Not recommended)
N_STEPS		long	0	Number of integration steps for non-adaptive integration.
POLE_FACTOR1		double	0.1557150345504	Strength factor for the first and last pole.
POLE_FACTOR2		double	0.380687012288581	Strength factor for the second and second-to-last pole.
POLE_FACTOR3		double	0.802829337348179	Strength factor for the third and third-to-last pole.
LASER_WAVELENGTH	M	double	0.0	Laser wavelength. If zero, the wavelength is calculated from the resonance condition.
LASER_PEAK_POWER	W	double	0.0	laser peak power
LASER_W0	M	double	1	laser spot size at waist, $w_0 = \sqrt{2}\sigma_x = \sqrt{2}\sigma_y$

A non-symplectic numerically integrated planar undulator including optional co-propagating laser beam for laser modulation of the electron beam.

Parameter Name	Units	Type	Default	Description
LASER_PHASE	RAD	double	0.0	laser phase
LASER_X0	M	double	0.0	laser horizontal offset at center of wiggler
LASER_Y0	M	double	0.0	laser vertical offset at center of wiggler
LASER_Z0	M	double	0.0	offset of waist position from center of wiggler
LASER_TILT	RAD	double	0.0	laser tilt
LASER_M		short	0	laser horizontal mode number (<5)
LASER_N		short	0	laser vertical mode number (<5)
SYNCH_RAD		short	0	Include classical, single-particle synchrotron radiation?
ISR		short	0	Include quantum excitation?
HELICAL		short	0	If non-zero, simulate helical undulator.
TIME_PROFILE	$NULL$	STRING	NULL	<filename>=<x>+<y> form specification of input file giving time-dependent modulation of the laser electric and magnetic fields.
TIME_OFFSET	S	double	0.0	Time offset of the laser profile.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a planar undulator, together with an optional co-propagating laser beam that can be used as a beam heater or modulator. The simulation is done by numerical integration of the Lorentz equation. It is not symplectic, and hence this element is not recommended for long-term tracking simulation of undulators in storage rings.

The fields in the undulator can be expressed in one of three ways. The FIELD_EXPANSION parameter is used to control which method is used.

- The exact field, given by (see section 3.1.5 of the *Handbook of Accelerator Physics and Engineering*)

$$B_x = 0, \tag{71}$$

$$B_y = B_0 \cosh k_u y \cos k_u z, \tag{72}$$

and

$$B_z = -B_0 \sinh k_u y \sin k_u z, \tag{73}$$

where $k_u = 2\pi/\lambda_u$ and λ_u is the undulator period. This is the most precise method, but also the slowest.

Experimental feature: One may also model a transverse gradient undulator (TGU) by setting the `TGU_GRADIENT` parameter to a non-zero value. In this case, taking a as the normalized gradient, the fields are [54]

$$B_x = \frac{aB_0 \sinh k_u y \cos k_u z}{k_u}, \tag{74}$$

$$B_y = B_0 \left((1 + ax) \cosh k_u y \cos k_u z + \frac{aC}{2k_u^2} \frac{eB_0}{\gamma m_e c} \right) \tag{75}$$

and

$$B_z = -B_0(1 + ax) \sinh k_u y \sin k_u z, \tag{76}$$

where γ is the central relativistic factor for the beam and C is given by the `TGU_COMP_FACTOR` parameter. This factor, and the term it multiplies, is present in order to help suppress the trajectory error at the end of the device. It may require adjustment in order to achieve the desired level of correction. In addition, the user may need to adjust the pole-strength factors and include external misalignments and steering magnets in order to supress not only the trajectory error, but also dispersion errors.

- The field expanded to leading order in y:

$$B_y = B_0(1 + \frac{1}{2}(k_u y)^2) \cos k_u z, \tag{77}$$

and

$$B_z = -B_0 k_u y \sin k_u z. \tag{78}$$

In most cases, this gives results that are very close to the exact fields, at a savings of 10% in computation time. This is the default mode.

- The "ideal" field:

$$B_y = B_0 \cos k_u z, \tag{79}$$

$$B_z = -B_0 k_u y \sin k_u z. \tag{80}$$

This is about 10% faster than the leading-order mode, but less precise. Also, *it does not include vertical focusing*, so it is not generally recommended.

If `HELICAL` is set to a nonzero value, a helical device is modeled by combining the fields of two planar devices, one of which is rotated 90 degrees and displaced one quarter wavelength. Again, the FIELD_EXPANSION parameter is used to control which method is used.

- The exact fields are

$$B_x = -B_0 \cosh k_u x \sin k_u z, \tag{81}$$

$$B_y = B_0 \cosh k_u y \cos k_u z, \tag{82}$$

and

$$B_z = -B_0 \sinh k_u y \sin k_u z - B_0 \sinh k_u x \cos k_u z, \tag{83}$$

- The field expanded to leading order in x and y:

$$B_x = -B_0(1 + \frac{1}{2}(k_u x)^2) \sin k_u z, \tag{84}$$

$$B_y = B_0(1 + \frac{1}{2}(k_u y)^2) \cos k_u z, \tag{85}$$

and

$$B_z = -B_0 k_u y \sin k_u z - B_0 k_u x \cos k_u z. \tag{86}$$

- The "ideal" field is

$$B_x = -B_0 \sin k_u z, \tag{87}$$

$$B_y = B_0 \cos k_u z, \tag{88}$$

$$B_z = 0 \tag{89}$$

This is about 10% faster than the leading-order mode, but less precise. Also, *it does not include vertical focusing*, so it is not generally recommended.

The expressions for the laser field used by this element are from A. Chao's article "Laser Acceleration — Focussed Laser," available on-line at http://www.slac.stanford.edu/~achao/LaserAccelerationFocussed.pdf . The implementation covers laser modes TEMij, where $0 \leq i \leq 4$ and $0 \leq j \leq 4$.

By default, if the laser wavelength is not given, it is computed from the resonance condition:

$$\lambda_l = \frac{\lambda_u}{2\gamma^2} \left(1 + \frac{1}{2} K^2\right), \tag{90}$$

where γ is the relativistic factor for the beam and K is the undulator parameter.

The adaptive integrator doesn't work well for this element, probably due to sudden changes in field derivatives in the first and last three poles (a result of the implementation of the undulator terminations). Hence, the default integrator is non-adaptive Runge-Kutta. The integration accuracy is controlled via the N_STEPS parameter. N_STEPS should be about 100 times the number of undulator periods.

The three pole factors are defined so that the trajectory is centered about $x = 0$ and $x' = 0$ with zero dispersion. This wouldn't be true with the standard two-pole termination, which might cause problems overlapping the laser with the electron beam.

The laser time profile can be specified using the `TIME_PROFILE` parameter to specify the name of an SDDS file containing the profile. If given, the electric and magnetic fields of the laser are multiplied by the profile $P(t)$. Hence, the laser intensity is multiplied by $P^2(t)$. By default $t = 0$ in the profile is lined up with $\langle t \rangle$ in the electron bunch. This can be changed with the `TIME_OFFSET` parameter. A positive value of `TIME_OFFSET` moves the laser profile forward in time (toward the head of the bunch).

Explanation of <filename>=<x>+<y> format: Several elements in `elegant` make use of data from external files to provide input waveforms. The external files are SDDS files, which may have many columns. In order to provide a convenient way to specify both the filename and the columns to use, we frequently employ <filename>=<x>+<y> format for the parameter value. For example, if the parameter value is `waveform.sdds=t+A`, then it means that columns `t` and `A` will be taken from file `waveform.sdds`. The first column is always the independent variable (e.g., time, position, or frequency), while the second column is the dependent quantity.

10.59 LTHINLENS—A thin lens for light optics

A thin lens for light optics
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
FX	M	double	0.0	focal length in horizontal plane
FY	M	double	0.0	focal length in vertical plane
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
TILT	RAD	double	0.0	misalignment rotation about longitudinal axis
YAW	RAD	double	0.0	misalignment rotation about vertical axis
PITCH	RAD	double	0.0	misalignment rotation about transverse horizontal axis
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.60 MAGNIFY—An element that allows multiplication of phase-space coordinates of all particles by constants.

An element that allows multiplication of phase-space coordinates of all particles by constants.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
MX		double	1	factor for x coordinates
MXP		double	1	factor for x' coordinates
MY		double	1	factor for y coordinates
MYP		double	1	factor for y' coordinates
MS		double	1	factor for s coordinates
MDP		double	1	factor for (p-pCentral)/pCentral
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.61 MALIGN—A misalignment of the beam, implemented as a zero-order matrix.

A misalignment of the beam, implemented as a zero-order matrix.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
DXP		double	0.0	delta x'
DYP		double	0.0	delta y'
DX	M	double	0.0	delta x
DY	M	double	0.0	delta y
DZ	M	double	0.0	delta z
DT	S	double	0.0	delta t
DP		double	0.0	delta p/pCentral
DE		double	0.0	delta gamma/gammaCentral
ON_PASS		long	-1	pass on which to apply
FORCE_MODIFY_MATRIX		long	0	modify the matrix even if on_pass>=0
START_PID		long	-1	starting particleID for particles to affect. By default, all particles are affected.
END_PID		long	-1	ending particleID for particles to affect. By default, all particles are affected.
FLOOR		long	0	if non-zero, floor coordinates are changed, which is probably a bad idea
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The default value of the `PASS` parameter (-1) means that the misalignment is imposed on the beam *every* pass. This is appropriate for static misalignments. When using the `MALIGN` element to kick the beam for beam dynamics studies in rings, `PASS>=0` is required. If `PASS=0`, closed orbit computation and correction will include the effect of the kick; however, matrix-based computations by default will not (set `FORCE_MODIFY_MATRIX=1` to change this). If `PASS>0`, then closed orbit computation and correction do not include the kick, which is probably what is desired in beam dynamics studies in rings.

10.62 MAPSOLENOID—A numerically-integrated solenoid specified as a map of (Bz, Br) vs (z, r).

A numerically-integrated solenoid specified as a map of (Bz, Br) vs (z, r).
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
ETILT	RAD	double	0.0	misalignment
EYAW	RAD	double	0.0	misalignment
EPITCH	RAD	double	0.0	misalignment
N_STEPS		long	100	number of steps (for nonadaptive integration)
INPUTFILE		STRING	NULL	SDDS file containing (Br, Bz) vs (r, z). Each page should have values for a fixed r.
RCOLUMN		STRING	NULL	column containing r values
ZCOLUMN		STRING	NULL	column containing z values
BRCOLUMN		STRING	NULL	column containing Br values
BZCOLUMN		STRING	NULL	column containing Bz values
FACTOR		double	0.0001	factor by which to multiply fields in file
BXUNIFORM		double	0.0	uniform horizontal field to superimpose on solenoid field
BYUNIFORM		double	0.0	uniform vertical field to superimpose on solenoid field
LUNIFORM		double	0.0	length of uniform field superimposed on solenoid field
ACCURACY		double	0.0001	integration accuracy
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, non-adaptive runge-kutta, modified midpoint)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The input file must be an SDDS file and must have one page for each value of the radius. Each page must be sorted in increasing order by the z coordinate. The `z` and `r` data must be in meters, while the `Bz` and `Br` data must be in Tesla.

Starting with a single-page file containing (z, r, Bz, Br), one can easily produce the required file using this command sequence

```
sddssort solenoid.sdds -pipe=out -column=r -column=z \
 | sddsbreak -pipe=in mapSolenoid.sdds -change=r
```

The preferred integrator is non-adaptive Runge-Kutta, since the adaptive integrators tend to have convergence problems.

N.B.: for historical reasons, the default value of the `FACTOR` parameter is set to `0.0001`, which is apt to introduce confusion.

10.63 MARK—A marker, equivalent to a zero-length drift space.

A marker, equivalent to a zero-length drift space.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
DX	M	double	0.0	non-functional misalignment (e.g., for girder)
DY	M	double	0.0	non-functional misalignment (e.g., for girder)
FITPOINT		short	`0`	Supply local values of Twiss parameters, moments, floor coordinates, matrices, etc. for optimization?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

If `FITPOINT=0`, this element results only in generation of additional output rows in the various files that contain output vs s. For example, Twiss parameters, closed orbits, and matrices vs s will all contain a row for each occurrence of each marker element.

If `FITPOINT=1`, the element has additional functionality in the context of optimizations. In particular, for occurrence N of the defined element *Element*, a series of symbols are created of the form *Element#N.quantity*, where *quantity* has the following values:

- The quantity `pCentral` will be available, giving the reference value of $\beta\gamma$ at the marker location.
- The quantities `Cx`, `Cxp`, `Cy`, `Cyp`, `Cs`, and `Cdelta` will be available, giving coordinate centroid values from tracking to the marker location.
- The quantities `Sx`, `Sxp`, `Sy`, `Syp`, `Ss`, and `Sdelta` will be available, giving coordinate rms values $\sqrt{\langle (x_i - \langle x_i \rangle)^2 \rangle}$ at the marker location from tracking.
- The quantity `Particles` will be available, giving the number of particles tracked to the marker location.
- The quantities `s`ij will be available, giving $\langle (x_i - \langle x_i \rangle)(x_j - \langle x_j \rangle) \rangle$ from tracking at the marker location, where $1 \leq i \leq 6$ and $i < j \leq 6$.
- The quantities `betaxBeam`, `alphaxBeam`, `betayBeam`, and `alphayBeam`, which are the twiss parameters computed from the beam moments obtained by tracking, will be available.

- The quantities Rij will be available, for $1 \leq i \leq 6$ and $1 \leq j \leq 6$, giving the accumulated first-order transport matrix to the marker location.

- If the default matrix order (as set in run_setup) is 2 or greater, the quantities Tijk will be available, for $1 \leq i \leq 6$, $1 \leq j \leq 6$, and $1 \leq k \leq j$, giving the accumulated second-order transport matrix to the marker location.

- If Twiss parameter calculations are being performed (via twiss_output with output_at_each_step=1), then the quantities alphax, betax, nux, psix, etax, etapx, and etaxp, along with similarly-named quantities for the vertical plane, will be available, giving twiss parameter values at the marker location. Note that etapx and etaxp are the same, being alternate names for η'_x. If radiation integrals are requested, the values of the radiation integrals are available in the quantities I1, I2, etc.

- If coupled Twiss parameter calculations are being performed (via coupled_twiss_output with output_at_each_step=1), then the quantities betax1, betax2, betay1, betay2, cetax, cetay, and tilt will be available. (These are the two beta functions for x and y, the coupled dispersion values for x and y, and the beam tilt).

- If moments calculations are being performed (via moments_output with output_at_each_step=1), then the quantities sijm, $1 \leq i \leq j \leq 6$, giving the 21 unique elements of the sigma matrix, are available, as are sijbetam, $1 \leq i\ leqj4$, giving the 6 unique elements with the dispersive terms removed. The quantities cim, $1 \leq i \leq 6$, are also created, giving the 6 centroids from the moments computation. The projected betatron emittances are available using exbeta and eybeta. In addition, the emittances of the three modes are available using eim, $1 \leq i \leq 3$, without any prefix. The m on the end of the symbols is to distinguish them from the moments computed from tracking.

- If floor coordinate calculations are begin performed (via floor_coordinates), then the quantities X, Y, Z, theta, phi, psi, and s will be available. These are, respectively, the three position coordinates, the three angle coordinates, and the total arc length at the marker location.

The misalignment controls for this element are non-functional, in the sense that they do not affect the beam. However, when combined with external scripts and the GROUP parameter, one can use this feature to implement girder misalignments using pairs of markers to indicate the ends of the girders. A future version of elegant will implement this internally.

10.64 MATR—Explicit matrix input from a text file, in the format written by the print_matrix command.

Explicit matrix input from a text file, in the format written by the print_matrix command.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
FRACTION	*NULL*	double	1	The provided matrix M is interpolated with the identity matrix I according to f*M+(1-f)*I.
FILENAME		STRING		input file
ORDER		short	1	matrix order
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The input file for this element uses a simple text format. It is nearly identical to the output in the `printout` file generated by the `matrix_output` and `analyze_map` commands. For example, for a 1st-order matrix, the file would have the following appearance:
C: *C1 C2 C3 C4 C5 C6*
R1: *R11 R12 R13 R14 R15 R16*
R2: *R21 R22 R23 R24 R25 R26*
R3: *R31 R32 R33 R34 R35 R36*
R4: *R41 R42 R43 R44 R45 R46*
R5: *R51 R52 R53 R54 R55 R56*
R6: *R61 R62 R63 R64 R65 R66*

Items in normal type must be entered exactly as shown, whereas those in italics must be provided by the user. The colons are important! For this particular example, one would set `ORDER=1` in the `MATR` definition. Typically, the *Ci* are zero, except for *C5*, which is usually equal to the length of the element (which must be specified with the `L` parameter in the `MATR` definition).

As of release 2019.2, the required format changed slightly. In the new version, the start of the matrix is determined by reading through the file until a line starting with `C:` is found. In the past, instead of starting with `C:`, the first line of the matrix could start with any string terminated by a colon, but that line had to be the first line in the file, which conflicted with the format emitted by `analyze_map`.

The `FRACTION` parameter can be used to interpolate the matrix elements between the matrix

M_0 read from `FILENAME` and the identity matrix I, according to

$$M = fM_0 + (1 - f)I. \tag{91}$$

This can be used, for example, to gradually ramp in the effect as part of an optimization. N.B.: in general, the matrix does not have unit determinant unless $f = 0$ or $f = 1$, so this feature should be used only as a knob to assist finding a solution with $f = 1$. Exceptions are when M_0 is a drift space or thin-lens quadrupole matrix, in which cases the determinant of M is always 1.

10.65 MATTER—A Coulomb-scattering and energy-absorbing element simulating material in the beam path.

A Coulomb-scattering and energy-absorbing element simulating material in the beam path.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
LEFFECTIVE	M	double	0.0	effective length (used if L=0)
XO	M	double	0.0	radiation length
ENERGY_DECAY		long	0	If nonzero, then particles will lose energy due to material using a simple exponential model.
ENERGY_STRAGGLE		long	0	Use simple-minded energy straggling model coupled with ENERGY_DECAY=1?
NUCLEAR_BREMSSTRAHLUNG		long	0	Model energy loss to nuclear bremsstrahlung? If enabled, set ENERGY_DECAY=0 to disable simpler model.
ELECTRON_RECOIL		long	0	If non-zero, electron recoil during Coulomb scattering is included (results in energy change).
Z		long	0	Atomic number
A	AMU	double	0.0	Atomic mass
RHO	KG/M^3	double	0.0	Density
PRESSURE	$PASCAL$	double	0.0	Pressure. Used with temperature and atomic mass to compute density for ideal gas.
TEMPERATURE	K	double	0.0	Temperature. Used with pressure and atomic mass to compute density for ideal gas.
MULTIPLICITY		long	1	Atoms per gas molecule.
PLIMIT		double	0.05	Probability cutoff for each slice
WIDTH	M	double	0.0	Full width of slots. If 0, no slots are present.
SPACING	M	double	0.0	Center-to-center spacing of slots. If 0, no slots are present.

A Coulomb-scattering and energy-absorbing element simulating material in the beam path.

Parameter Name	Units	Type	Default	Description
TILT	RAD	double	0.0	Tilt of slot array about the longitudinal axis.
CENTER	M	double	0.0	Position of center of slot array in rotated frame.
N_SLOTS		long	0	Number of empty slots in material. If <=0, an infinite array is assumed.
START_PASS		long	-1	If non-negative, pass on which to start interaction with beam.
END_PASS		long	-1	If non-negative, pass on which to end interaction with beam.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is based on section 3.3.1 of the *Handbook of Accelerator Physics and Engineering*, specifically, the subsections **Single Coulomb scattering of spin-$\frac{1}{2}$ particles**, **Multiple Coulomb scattering through small angles**, and **Radiation length**. There are two aspects to this element: scattering and energy loss.

Scattering. The multiple Coulomb scattering formula is used whenever the thickness of the material is greater than $0.001X_o$, where X_o is the radiation length. (Note that this is inaccurate for materials thicker than $100X_o$.) For this regime, the user need only specify the material thickness (L) and the radiation length (XO).

For materials thinner than $0.001X_o$, the user must specify additional parameters, namely, the atomic number (Z), atomic mass (A), and mass density (RHO) of the material. Note that the density is given in units of kg/m^3. (Multiply by 10^3 to convert g/cm^3 to kg/m^3.) In addition, the simulation parameter PLIMIT may be modified.

To understand this parameter, one must understand how `elegant` simulates the thin materials. First, it computes the expected number of scattering events per particle, $E = \sigma_T nL = \frac{K_1\pi^3 nL}{K_2^2+K_2*\pi^2}$, where n is the number density of the material, L is the thickness of the material, $K_1 = (\frac{2Zr_e}{\beta^2\gamma})^2$, and $K_2 = \frac{\alpha^2 Z^{\frac{2}{3}}}{(\beta\gamma)^2}$, with r_e the classical electron radius and α the fine structure constant. The material is then broken into N slices, where $N = E/P_{limit}$. For each slice, each simulation particle has a probability E/N of scattering. If scattering occurs, the location within the slice is computed using a uniform distribution over the slice thickness.

For each scatter that occurs, the scattering angle, θ is computed using the cumulative probability distribution $F(\theta > \theta_o) = \frac{K_2(\pi^2-\theta_o^2)}{\pi^2(K_2+\theta_o^2)}$. This can be solved for θ_o, giving $\theta_o = \sqrt{\frac{(1-F)K_2\pi^2}{K_2+F\pi^2}}$. For each scatter, F is chosen from a uniform random distribution on $[0, 1]$.

Energy loss. There are two ways to compute energy loss in materials, using a simple minded approach and using the bremsstrahlung cross section. The latter is recommended, but the former is kept for backward compatibility.

- To enable bremsstrahlung simulation, simply set `NUCLEAR_BREMSSTRAHLUNG=1`. Note that the energy loss is not correlated with the scattering angle, which is not entirely physical but should be reasonable for large numbers of scattering events.

- To use the simplified approach:

  - Set `ENERGY_DECAY=1`. Energy loss simulation is very simple. The energy loss per unit distance traveled, x, is $\frac{dE}{dx} = -E/X_o$. Hence, in traveling through a material of thickness L, the energy of each particle is transformed from E to Ee^{-L/X_o}.
  - Optionally, set `ENERGY_STRAGGLE=1`. **Not recomemnded. Exists only for backward compatibility.** This adds variation in the energy lost by particles. The model is *very*, *very* crude and **not recommended**. It assumes that the standard deviation of the energy loss is equal to half the mean energy loss. This is an overestimate, we think, and is provided to give an upper bound on the effects of energy straggling until a real model can be developed. Note one obvious problem with this: if you split a MATTER element of length L into two pieces of length L/2, the total energy loss will not not change, but the induced energy spread will be about 30% lower, due to addition in quadrature.

Slotted absorber. If the `WIDTH` and `SPACING` parameters are set to non-zero values, then a slotted absorber is simulated. The number of slots is by default infinite, but can be limited by setting `N_SLOTS` to a positive value; in this case, the slot array is centered about the transverse coordinate given by the `CENTER` parameter.

Note that the simulation contains a simplification in that particles cannot leave or enter the material through the side of the slot. I.e., if a particle is inside (outside) the material when it hits the front face of the object, it is assumed to remain inside (outside) until it has passed the object. For long objects, breaking the simulation up into multiple MATTER elements is suggested if a slotted arrangement is being simulated.

One-sided scrapers. One sided scrapers may be modeled using the `SCRAPER` element. It uses the same material-modeling algorithm as described here.

10.66 MAXAMP—A collimating element that sets the maximum transmitted particle amplitudes for all following elements, until the next MAXAMP.

A collimating element that sets the maximum transmitted particle amplitudes for all following elements, until the next MAXAMP.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
X_MAX	M	double	0.0	x half-aperture
Y_MAX	M	double	0.0	y half-aperture
ELLIPTICAL		long	0	is aperture elliptical?
EXPONENT		long	2	exponent for boundary equation in elliptical mode. 2 is a true ellipse.
YEXPONENT		long	0	y exponent for boundary equation in elliptical mode. If zero, defaults to EXPONENT.
OPEN_SIDE		STRING	NULL	which side, if any, is open (+x, -x, +y, -y)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element sets the aperture for itself and all subsequent elements. The settings are in force until another `MAXAMP` element is seen. Settings are also enforced inside of `KQUAD`, `KSEXT`, `KOCT`, `KQUSE`, `CSBEND`, and `CSRCSBEND` elements.

This can introduce unexpected behavior when beamlines are reflected. For example, consider the beamline

```
...
L1:  LINE=( ... )
L2:  LINE=( ... )
MA1: MAXAMP,X_MAX=0.01,Y_MAX=0.005
MA2: MAXAMP,X_MAX=0.01,Y_MAX=0.002
BL1: LINE=(MA1,L1,MA2,L2)
BL:  LINE=(BL1,-BL1)
```

This is equivalent to

```
BL:  LINE=(MA1,L1,MA2,L2,-L2,MA2,-L1,MA1)
```

Note that the aperture MA1 is the aperture for all of the first instance of beamline L1, but that MA2 is the aperture for the second instance, -L1. This is probably not what was intended. To prevent this, it is recommended to always use MAXAMP elements in pairs:

```
BL1: LINE=(MA2,MA1,L1,MA1,MA2,L2)
BL:  LINE=(BL1,-BL1)
```

which is equivalent to

```
BL:  LINE=(MA2,MA1,L1,MA1,MA2,L2,-L2,MA2,MA1,-L1,MA1,MA2)
```

Now, both instances of L1 have the aperture defined by MA1 and both instances of L2 have the aperture defined by MA2.

The default values of X_MAX and Y_MAX are 0, which causes the aperture to be ignored. This means one cannot use the MAXAMP element to simulate a completely blocked beam pipe.

10.67 MBUMPER—A time-dependent multipole kicker magnet. The waveform is in SDDS format, with time in seconds and amplitude normalized to 1.

A time-dependent multipole kicker magnet. The waveform is in SDDS format, with time in seconds and amplitude normalized to 1.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
STRENGTH		double	0.0	geometric strength in 1/m$\hat{(}$order+1)
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
TIME_OFFSET	S	double	0.0	time offset of waveform
ORDER		long	0	multipole order, where 1 is quadrupole, 2 is sextupole, etc.
PERIODIC		long	0	is waveform periodic?
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
FIRE_ON_PASS		long	0	pass number to fire on
N_KICKS		long	0	Number of kicks to use for simulation.
WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving kick factor vs time
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a time-dependent multipole kicker magnet. To use this element, you must supply an SDDS file giving the time-dependent waveform. The element is called `MBUMPER` to because `HKICK`, `VKICK`, `KICKER` are used for steering magnets.

The arrival time of the beam is taken to define the reference time, $t = 0$. Hence, if the waveform file has the maximum amplitude at $t = 0$, the beam will get kicked at the peak of the waveform.

If the waveform peaks at $t = t_{peak}$, then setting TIME_OFFSET equal to $-t_{peak}$ will ensure that the beam is kicked at the peak amplitude.

By default, the kicker fires on the first beam passage. However, if FIRE_ON_PASS is used, then the kicker is treated like a drift space until the specified pass. Note that teh first pass is 0, not 1.

If PHASE_REFERENCE is non-zero, then the initial timing is taken from the first time-dependent element that has the same PHASE_REFERENCE value. This would allow, for example, simulating several kickers firing at the same time. Delays relative to this reference time can then be given with positive adjustments to TIME_OFFSET.

The input file need not have equispaced points in time. However, the time values should increase monotonically.

This element simulates a quadrupole or higher order kicker only. For dipole kickers, see the BUMPER element.

Explanation of <filename>=<x>+<y> format: Several elements in elegant make use of data from external files to provide input waveforms. The external files are SDDS files, which may have many columns. In order to provide a convenient way to specify both the filename and the columns to use, we frequently employ <filename>=<x>+<y> format for the parameter value. For example, if the parameter value is waveform.sdds=t+A, then it means that columns t and A will be taken from file waveform.sdds. The first column is always the independent variable (e.g., time, position, or frequency), while the second column is the dependent quantity.

10.68 MHISTOGRAM—Request for multiple dimensions (1, 2, 4 or 6) histogram output of particle coordinates.

Request for multiple dimensions (1, 2, 4 or 6) histogram output of particle coordinates.
Parallel capable? : no
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
FILE1D		STRING	NULL	filename for 1d histogram output, possibly incomplete (see below)
FILE2DH		STRING	NULL	filename for 2d x-x' histogram output, possibly incomplete (see below)
FILE2DV		STRING	NULL	filename for 2d y-y' histogram output, possibly incomplete (see below)
FILE2DL		STRING	NULL	filename for 2d dt-deltaP histogram output, possibly incomplete (see below)
FILE4D		STRING	NULL	filename for 4d x-x'-y-y' histogram output, possibly incomplete (see below)
FILE6D		STRING	NULL	filename for 6d x-x'-y-y'-dt-deltaP histogram output, possibly incomplete (see below)
INPUT_BINS		STRING	NULL	Name of SDDS file contains input bin number.
INTERVAL		long	1	interval in passes between output.
START_PASS		long	0	starting pass for output
NORMALIZE		short	1	normalize histogram with number of particles?
DISABLE		short	0	If nonzero, no output will be generated.
LUMPED		short	0	If nonzero, then results at elements with same name will be output to a single multipage SDDS file.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is used to generate multiple dimension (1, 2, 4, or 6) histogram output of particle coordinates.

The calculation is set up through output filename: `FILE1D`, `FILE2DH`, `FILE2DV`, `FILE2DL`, `FILE4D`, `FILE6D`. They may be an incomplete filename (see `HISTOGRAM` for detail). If `LUMPED` set to non zero, then results are directed to a multi page SDDS file with each page contains data of same elements `MHISTOGRAM` but at difference occurrence instead of multiple SDDS files. In this case the "%ld" in filename is ignored.

The bin number used to do histogram analysis is given through a SDDS file from `INPUT_BINS`. It contains 4 columns: `Bins_1D`, `Bins_2D`, `Bins_4D`, `Bins_6D`; and 6 rows (x, x', y, y', dt, delta). A non-zero value in `Bins_1D` is a switch for doing histogram analysis in corresponding dimension, and the maximum value in `Bins_1D` is used as bin number to do the analysis.

The normalization is different from `HISTOGRAM` as we alwayse treat bin-size = 1.

The output file uses the general format designed for a n-dimensional histogram data. It must contains a column named "Frequency" (Type: "double"), and following parameters:

- `ND` — Type: long; Value: "n".
- `Variable??Name` — Type: "string". "??" counts from "0" to "ND-1" in double digits format, same for all following parameters.
- `Variable??Min` — Type: "double". Minimum value of "??" variable.
- `Variable??Max` — Type: "double". Maximum value of "??" variable.
- `Variable??Interval` — Type: "double". Bin size of "??" variable.
- `Variable??Dimension` — Type: "long". Total number of bins of "??" variable. Variable??Dimension = (Variable??Max - Variable??Min)/Variable??Interval+1.

The data is arranged as it has a "ND" index counter $[i_{ND-1}|...|i_1]$, where i_{ND-1} takes value from "0" to "Variable[%02d ND-1]Dimension".

10.69 MODRF—A first-order matrix RF cavity with exact phase dependence, plus optional amplitude and phase modulation.

A first-order matrix RF cavity with exact phase dependence, plus optional amplitude and phase modulation.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
VOLT	V	double	0.0	nominal voltage
PHASE	DEG	double	0.0	nominal phase
FREQ	Hz	double	500000000	nominal frequency
Q		double	0.0	cavity Q
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
AMMAG		double	0.0	magnitude of amplitude modulation (fraction value)
AMPHASE	DEG	double	0.0	phase of amplitude modulation
AMFREQ	Hz	double	0.0	frequency of amplitude modulation
AMDECAY	$1/s$	double	0.0	exponential decay rate of amplitude modulation
PMMAG	DEG	double	0.0	magnitude of phase modulation
PMPHASE	DEG	double	0.0	phase of phase modulation
PMFREQ	Hz	double	0.0	frequency of phase modulation
PMDECAY	$1/s$	double	0.0	exponential decay rate of phase modulation
FIDUCIAL		STRING	NULL	mode for determining fiducial arrival time (light, tmean, first, pmaximum)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is very similar to the `RFCA` element, except that the amplitude and phase of the cavity can be modulated.

The phase convention is as follows, assuming a positive rf voltage: `PHASE=90` is the crest for acceleration. PHASE=180 is the stable phase for a storage ring above transition without energy losses.

The element works by first computing the fidicial arrival time $\bar{t}$. Using this, the effective voltage is computed using the amplitude modulation parameters, according to

$$V_e = V_0(1 + A_{am} \sin(\omega_{am}\bar{t} + \phi_{am}) \exp(-\alpha_{am}\bar{t})) \tag{92}$$

where V_0 is the nominal cavity voltage VOLT, A_{am} is AMMAG, ω_{am} is the angular frequency corresponding to AMFREQ, ϕ_{am} is the amplitude modulation phase corresponding to AMPHASE (converted from degrees to radians), and α_{am} is AMDECAY.

The phase of the phase modulation is computed using

$$\phi_{pm} = \omega_{pm}\bar{t} + \Delta\phi_{pm}, \tag{93}$$

where ω_{pm} is the angular frequency corresponding to PMFREQ and $\Delta\phi_{pm}$ is the phase offset corresponding to PMPHASE (converted from degrees to radians). The rf phase for the centroid is then computed using

$$\phi = \omega_0\bar{t} + \phi_0 + \Phi_m \sin(\phi_{pm}) \exp(-\alpha_{pm}\bar{t}), \tag{94}$$

where ω_0 is the nominal rf angular frequency (corresponding to FREQ), ϕ_0 corresponds to PHASE (converted to radians), Φ_m corresponds to PMMAG (converted to radians), and α_{pm} corresponds to PMDECAY.

The effective instantaneous rf angular frequency is

$$\omega = \omega_0 + \omega_{pm}\Phi_m \cos\phi_{pm}. \tag{95}$$

Using all of the above, the voltage seen by a particle arriving at time t is then

$$V = V_e \sin(\omega(t - \bar{t}) + \phi). \tag{96}$$

10.70 MONI—A two-plane position monitor, accepting two rpn equations for the readouts as a function of the actual positions (x and y).

A two-plane position monitor, accepting two rpn equations for the readouts as a function of the actual positions (x and y).
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
WEIGHT		double	1	weight in correction
TILT		double	0.0	rotation about longitudinal axis
XCALIBRATION		double	1	calibration factor for x readout
YCALIBRATION		double	1	calibration factor for y readout
XSETPOINT	M	double	0.0	x steering setpoint
YSETPOINT	M	double	0.0	y steering setpoint
ORDER		short	0	matrix order
XREADOUT		STRING	NULL	rpn expression for x readout (actual position supplied in variables x, y
YREADOUT		STRING	NULL	rpn expression for y readout (actual position supplied in variables x, y
CO_FITPOINT		short	0	If nonzero, then closed orbit values are placed in variables <name>#<occurence>.xco and <name>#<occurence>.yco
STORE_TURN_BY_TURN		short	0	If nonzero, then turn-by-turn horizontal/vertical position readouts and number of particles are placed in variable <name>#<occurence>.x/y/n.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

If `STORE_TURN_BY_TURN` is nonzero, then the computed BPM readings are stored in *Element-*

Name#ElementOccurence.x and *ElementName#ElementOccurence*.y during tracking. These can be used in the `expression` in `modulate`elements— to create position-triggered changes to elements.

10.71 MRFDF—Zero-length Multipole RF DeFlector from dipole to decapole

Zero-length Multipole RF DeFlector from dipole to decapole
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
FACTOR		double	1	A factor by which to multiply all components.
TILT	RAD	double	0.0	rotation about longitudinal axis
A1	V/m	double	0.0	Vertically-deflecting dipole
A2	V/m^2	double	0.0	Skew quadrupole
A3	V/m^3	double	0.0	Skew sextupole
A4	V/m^4	double	0.0	Skew octupole
A5	V/m^5	double	0.0	Skew decapole
B1	V/m	double	0.0	Horizontally-deflecting dipole
B2	V/m^2	double	0.0	Normal quadrupole
B3	V/m^3	double	0.0	Normal sextupole
B4	V/m^4	double	0.0	Normal octupole
B5	V/m^5	double	0.0	Normal decapole
FREQUENCY1	HZ	double	2856000000	Dipole frequency
FREQUENCY2	HZ	double	2856000000	Quadrupole frequency
FREQUENCY3	HZ	double	2856000000	Sextupole frequency
FREQUENCY4	HZ	double	2856000000	Octupole frequency
FREQUENCY5	HZ	double	2856000000	Decapole frequency
PHASE1	HZ	double	0.0	Dipole phase
PHASE2	HZ	double	0.0	Quadrupole phase
PHASE3	HZ	double	0.0	Sextupole phase
PHASE4	HZ	double	0.0	Octupole phase
PHASE5	HZ	double	0.0	Decapole phase
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
START_PASS		long	-1	If non-negative, pass on which to start modeling cavity.
END_PASS		long	-1	If non-negative, pass on which to end modeling cavity.
START_PID		long	-1	If non-negative, lowest particle ID to which deflection is applied.

Zero-length Multipole RF DeFlector from dipole to decapole

Parameter Name	Units	Type	Default	Description
END_PID		long	-1	If non-negative, highest particle ID to which deflection is applied.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates an rf deflector with specified multipole content.

Assuming for simplicity that $y = 0$, the momentum change in the horizontal plane is

$$\Delta p_x = \frac{e}{mc^2 k} \sum_{i=1}^{5} i b_i x^{i-1} \cos\phi_i, \tag{97}$$

where $k = \omega/c$ and $p_x = \beta_x \gamma$. The deflection is

$$\Delta x' \approx \frac{\Delta p_x}{p_z}, \tag{98}$$

where the approximation results from the fact that $p_z = \beta_z \gamma$ also changes in order to satisfy Maxwell's equations.

10.72 MULT—A canonical kick multipole.

A canonical kick multipole.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
KNL	M^{-ORDER}	double	0.0	integrated geometric strength
TILT	RAD	double	0.0	rotation about longitudinal axis
BORE	M	double	0.0	bore radius
BTIPL	TM	double	0.0	integrated field at pole tip, used if BORE nonzero
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FACTOR		double	1	factor by which to multiply strength
ORDER		short	1	multipole order
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
EXPAND_HAMILTONIAN		short	0	If 1, Hamiltonian is expanded to leading order.
N_SLICES		long	4	Number of slices (full integrator steps).
N_KICKS		long	4	Deprecated. Use N_SLICES.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a multipole element using 4th-order sympletic integration. A single multipole order, n, is given. The multipole strength is specified by giving

$$K_n L = \left(\frac{\partial^n B_y}{\partial x^n}\right)_{x=y=0} \frac{L}{B\rho}, \tag{99}$$

where $B\rho$ is the beam rigidity. A quadrupole is $n = 1$, a sextupole is $n = 2$, and so on.
The relationship between the pole tip field and $K_n L$ is

$$K_n L = \frac{n!\, B_{tip} L}{r^n (B\rho)}, \tag{100}$$

where r is the bore radius.

10.73 NIBEND—A numerically-integrated dipole magnet with various extended-fringe-field models.

A numerically-integrated dipole magnet with various extended-fringe-field models.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	arc length
ANGLE	*RAD*	double	0.0	bending angle
E1	*RAD*	double	0.0	entrance edge angle
E2	*RAD*	double	0.0	exit edge angle
TILT		double	0.0	rotation about incoming longitudinal axis
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
DZ	*M*	double	0.0	misalignment
FINT		double	0.5	edge-field integral
HGAP	*M*	double	0.0	half-gap between poles
FP1	*M*	double	10	fringe parameter (tanh model)
FP2	*M*	double	0.0	not used
FP3	*M*	double	0.0	not used
FP4	*M*	double	0.0	not used
FSE		double	0.0	fractional strength error
ETILT	*RAD*	double	0.0	error rotation about incoming longitudinal axis
ACCURACY		double	0.0001	integration accuracy (for non-adaptive integration, used as the step-size)
MODEL		STRING	linear	fringe model (hard-edge, linear, cubic-spline, tanh, quintic, enge1, enge3, enge5)
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, modified-midpoint, two-pass modified-midpoint, leap-frog, non-adaptive runge-kutta)
SYNCH_RAD		long	0	include classical, single-particle synchrotron radiation?
ADJUST_BOUNDARY		long	1	adjust fringe boundary position to make symmetric trajectory? (Not done if ADJUST_FIELD is nonzero.)

A numerically-integrated dipole magnet with various extended-fringe-field models.

Parameter Name	Units	Type	Default	Description
ADJUST_FIELD		long	0	adjust central field strength to make symmetric trajectory?
FUDGE_PATH_LENGTH		long	1	fudge central path length to force it to equal the nominal length L?
FRINGE_POSITION		long	0	0=fringe centered on reference plane, -1=fringe inside, 1=fringe outside.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

For the NIBEND element, there are various fringe field models available. In the following descriptions, l_f is the extend of the fringe field, which starts at $z = 0$ for convenience in the expressions. Also, $K = \frac{1}{g}\int_{-}\infty^{\infty} F_y(z)(1 - F_y(z))dz$ is K. Brown's fringe field integral (commonly called FINT), where g is the full magnet gap and $\vec{F} = \vec{B}/B_0$, B_0 being the value of the magnetic field well inside the magnet.

- **Linear fringe field:**

$$F_y = zF_a \tag{101}$$
$$F_z = yF_a \tag{102}$$
$$F_a = 1/(6Kg) \tag{103}$$

For this model, the user specifies FINT and HGAP only.

- **Cubic-spline fringe field:**

$$F_y = F_a z^2 + F_b z^3 + y^2(-F_a - 3F_b z) \tag{104}$$
$$F_z = (2F_a z + 3F_b z^2)y \tag{105}$$
$$F_a = 3/l_f^2 \tag{106}$$
$$F_b = -2/l_f^3 \tag{107}$$
$$l_f = 70Kg/9 \tag{108}$$

For this model, the user specifies FINT and HGAP only.

- **Tanh-like fringe field:**

$$
\begin{aligned}
F_y &= \frac{1}{2}(1+\tanh F_a z) + \frac{1}{2}(yF_a \mathrm{sech} F_a z)^2 \tanh F_a z + && (109)\\
&\quad \frac{1}{24}(yF_a \mathrm{sech} F_a z)^4 \mathrm{sech} F_a z (11 \sinh F_a z - \sinh 3F_a z) && (110)\\
F_z &= \frac{1}{2} y F_a \mathrm{sech}^2 F_a z + \frac{1}{6}(yF_a \mathrm{sech} F_a z)^3 \mathrm{sech} F_a z (2-\cosh 2F_a z)) + && (111)\\
&\quad \frac{1}{120}(yF_a \mathrm{sech} F_a z)^5 \mathrm{sech} F_a z (33 - 26\cosh 2F_a z + cosh4F_z z) && (112)\\
F_a &= 1/(2Kg) && (113)\\
l_f &= P_1/F_a && (114)
\end{aligned}
$$

For this model, the user specifies `FINT` and `HGAP`, along with the parameter `FP1`, which is the quantity P_1 in the last equation. It determines the length of the fringe field that is integrated.

- **Quintic-spline fringe field, to third order in y:**

$$
\begin{aligned}
F_y &= (F_a z^3 + F_b z^4 + F_c z^5) + y^2 z (3F_a + 6F_b z + 10F_c z^2) && (115)\\
F_z &= y(3F_a z^2 + 4F_b z^3 + 5F_c z^4) + y^3(-F_a - 4F_b z - 10F_c z^2) && (116)\\
F_a &= 10/l_f^3 && (117)\\
F_b &= -15/l_f^4 && (118)\\
F_c &= 6/l_f^5 && (119)\\
l_f &= 231Kg/25 && (120)
\end{aligned}
$$

For this model, the user specifies `FINT` and `HGAP` only.

- **Enge model with 3 coefficients:**

$$
\begin{aligned}
F_0 &= \frac{1}{1+e^{a_1 + a_2 z/D + a_3 (z/D)^2}} && (121)\\
F_y &= F_0 - \frac{1}{2} y^2 F_0^{(2)} + \frac{1}{24} y^4 F_0^{(4)} && (122)\\
F_z &= y F_0^{(1)} - \frac{1}{6} y^3 F_0^{(3)} + \frac{1}{120} y^5 F_0^{(5)} && (123)
\end{aligned}
$$

where $F_0^{(n)} = \frac{\partial^n F_0}{\partial z^n}$.

The user may choose "enge1", "enge3", or "enge5", where the number indicates the order of the expansion of F_z with respect to y.

The need only specify `FINT` and `HGAP.` The Enge parameters are then automatically determined to give the correct linear focusing.

However, if user gives non-zero value for `FP2,` then `FINT` and `HGAP` are ignored. `FP2`, `FP3`, and `FP4` and taken as the Enge coefficients a_1, a_2, and a_3, respectively.

10.74 NISEPT—A numerically-integrated dipole magnet with a Cartesian gradient.

A numerically-integrated dipole magnet with a Cartesian gradient.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	arc length
ANGLE	RAD	double	0.0	bend angle
E1	RAD	double	0.0	entrance edge angle
B1	$1/M$	double	0.0	normalized gradient (K1=B1*L/ANGLE)
Q1REF	M	double	0.0	distance from septum at which bending radius is L/ANGLE
FLEN	M	double	0.0	fringe field length
ACCURACY		double	0.0001	integration accuracy
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, modified-midpoint, two-pass modified-midpoint, leap-frog, non-adaptive runge-kutta
MODEL		STRING	linear	fringe model (hard-edge, linear, cubic-spline, tanh, quintic
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.75 OCTU—An octupole implemented as a third-order matrix. Use KOCT for symplectic tracking.

An octupole implemented as a third-order matrix. Use KOCT for symplectic tracking.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K3	$1/M^3$	double	0.0	geometric strength
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE		double	0.0	fractional strength error
ORDER		short	0	matrix order
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.76 PEPPOT—A pepper-pot plate.

A pepper-pot plate.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
RADII	M	double	0.0	hole radius
TRANSMISSION		double	0.0	transmission of material
TILT	RAD	double	0.0	rotation about longitudinal axis
THETA_RMS	RAD	double	0.0	rms scattering from material
N_HOLES		long	0	number of holes
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.77 PFILTER—An element for energy and momentum filtration.

An element for energy and momentum filtration.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
DELTALIMIT		double	-1	maximum fractional momentum deviation
LOWERFRACTION		double	0.0	fraction of lowest-momentum particles to remove
UPPERFRACTION		double	0.0	fraction of highest-momentum particles to remove
FIXPLIMITS		long	0	fix the limits in p from LOWERFRACTION and UPPERFRACTION applied to first beam
BEAMCENTERED		long	0	if nonzero, center for DELTALIMIT is average beam momentum
BINS		long	1024	number of bins
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.78 POLYSERIES—Tracks through a Polynomial series map specified by a file containing coefficients.

Tracks through a Polynomial series map specified by a file containing coefficients.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FILENAME		STRING	NULL	name of file containing talor series map data
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element allows transforming the beam using an arbitrary polynomial in the six phase space coordinates, x, q_x, y, q_y, s, and δ. Please note that the canonical momenta are used here, not the slopes.

The input file is required to have six pages, each of which gives the polynomial for a different output coordinate. The file has the following elements:

- String parameter `Coordinate` names the output coordinate. The file must contain a page for each coordinate, using the names `x`, `qx`, `y`, `qy`, `s`, and `delta`.
- Integer columns `Ix`, `Iqx`, `Iy`, `Iqy`, `Is`, and `Idelta`, which give the exponent to be used for the indicated coordinates.
- Floating-point column `Coefficient` giving the coefficient by which to multiply the product of powers of the coordinates. I.e., the term is $Cx^{\mathrm{Ix}}q_x^{\mathrm{Iqx}}y^{\mathrm{Iy}}q_y^{\mathrm{Iqy}}s^{\mathrm{Is}}\delta^{\mathrm{Idelta}}$

Note that in the present version, dependence on s is actually ignored.

10.79 QUAD—A quadrupole implemented as a matrix, up to 3rd order. Use KQUAD for symplectic tracking.

A quadrupole implemented as a matrix, up to 3rd order. Use KQUAD for symplectic tracking.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K1	$1/M^2$	double	0.0	geometric strength
TILT	RAD	double	0.0	rotation about longitudinal axis
PITCH	RAD	double	0.0	rotation about horizontal axis. Ignored if MALIGN_METHOD=0
YAW	RAD	double	0.0	rotation about vertical axis. Ignored if MALIGN_METHOD=0
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE		double	0.0	fractional strength error
HKICK	RAD	double	0.0	horizontal correction kick
VKICK	RAD	double	0.0	vertical correction kick
HCALIBRATION		double	1	calibration factor for horizontal correction kick
VCALIBRATION		double	1	calibration factor for vertical correction kick
HSTEERING		short	0	use for horizontal steering?
VSTEERING		short	0	use for vertical steering?
ORDER		short	0	matrix order
EDGE1_EFFECTS		short	1	include entrance edge effects?
EDGE2_EFFECTS		short	1	include exit edge effects?
FRINGE_TYPE		STRING	fixed-strength	type of fringe: "inset", "fixed-strength", or "integrals"
FFRINGE		double	0.0	For non-integrals mode, fraction of length occupied by linear fringe region.
LEFFECTIVE	M	double	-1	Effective length. Ignored if non-positive. Cannot be used with non-zero FFRINGE.

A quadrupole implemented as a matrix, up to 3rd order. Use KQUAD for symplectic tracking.

Parameter Name	Units	Type	Default	Description
I0P	M	double	0.0	i0+ fringe integral
I1P	M^2	double	0.0	i1+ fringe integral
I2P	M^3	double	0.0	i2+ fringe integral
I3P	M^4	double	0.0	i3+ fringe integral
LAMBDA2P	M^3	double	0.0	lambda2+ fringe integral
I0M	M	double	0.0	i0- fringe integral
I1M	M^2	double	0.0	i1- fringe integral
I2M	M^3	double	0.0	i2- fringe integral
I3M	M^4	double	0.0	i3- fringe integral
LAMBDA2M	M^3	double	0.0	lambda2- fringe integral
RADIAL		short	0	If non-zero, converts the quadrupole into a radially-focusing lens
MALIGN_METHOD		short	0	0=original, 1=new entrace-centered, 2=new body-centered
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a quadrupole using a matrix of first, second, or third order.

Length specification As of version 29.2, this element incorporates the ability to have different values for the insertion and effective lengths. This is invoked when `LEFFECTIVE` is positive. In this case, the `L` parameter is understood to be the physical insertion length. Using `LEFFECTIVE` is a convenient way to incorporate the fact that the effective length may differ from the physical length and even vary with excitation, without having to modify the drift spaces on either side of the quadrupole element.

Fringe effects By default, the element has hard edges and constant field within the defined length, `L`. However, this element supports two different methods of implementing fringe fields. Which method is used is determined by the `FRINGE_TYPE` parameter.

Edge integral method The most recent and preferred implementation of fringe field effects is based on edge integrals and is invoked by setting `FRINGE_TYPE` to "integrals". This method is compatible with the use of `LEFFECTIVE`. However, it provides a first-order matrix only.

The model is based on publications of D. Zhuo *et al.* [34] and J. Irwin *et al.* [35], as well as unpublished work of C. X. Wang (ANL). The fringe field is characterized by 10 integrals given in equations 19, 20, and 21 of [34]. However, the values input into `elegant` should be normalized by K_1 or K_1^2, as appropriate.

For the exit-side fringe field, let s_1 be the center of the magnet, s_0 be the location of the nominal end of the magnet (for a hard-edge model), and let s_2 be a point well outside the magnet. Using $K_{1,he}(s)$ to represent the hard edge model and $K_1(s)$ the actual field profile, we define the normalized difference as $\tilde{k}(s) = (K_1(s) - K_{1,he}(s))/K_1(s_1)$. (Thus, $\tilde{k}(s) = \tilde{K}(s)/K_0$, using the notation of Zhou *et al.*)

The integrals to be input to `elegant` are defined as

$$i_0^- = \int_{s_1}^{s_0} \tilde{k}(s)ds \qquad i_0^+ = \int_{s_0}^{s_2} \tilde{k}(s)ds \tag{124}$$

$$i_1^- = \int_{s_1}^{s_0} \tilde{k}(s)(s-s_0)ds \qquad i_1^+ = \int_{s_0}^{s_2} \tilde{k}(s)(s-s_0)ds \tag{125}$$

$$i_2^- = \int_{s_1}^{s_0} \tilde{k}(s)(s-s_0)^2ds \qquad i_2^+ = \int_{s_0}^{s_2} \tilde{k}(s)(s-s_0)^2ds \tag{126}$$

$$i_3^- = \int_{s_1}^{s_0} \tilde{k}(s)(s-s_0)^3ds \qquad i_3^+ = \int_{s_0}^{s_2} \tilde{k}(s)(s-s_0)^3ds \tag{127}$$

$$\lambda_2^- = \int_{s_1}^{s_0} ds \int_{s}^{s_0} ds\prime\tilde{k}(s)\tilde{k}(s\prime)(s\prime - s) \qquad \lambda_2^+ = \int_{s_0}^{s_2} ds \int_{s}^{s_2} ds\prime\tilde{k}(s)\tilde{k}(s\prime)(s\prime - s) \tag{128}$$

Normally, the effects are dominated by i_1^- and i_1^+.

Trapazoidal models This method is based on a third-order matrix formalism and the assumption that the fringe fields depend linearly on z. Although the third-order matrix is computed, it is important to note that the assumed fields do not satisfy Maxwell's equations.

To invoke this method, one specifies "inset" or "fixed-strength" for the `FRINGE_TYPE` parameter and then provides a non-zero value for `FFRINGE`. If `FFRINGE` is zero (the default), then the magnet is hard-edged regardless of the setting of `FRINGE_TYPE`. If `FFRINGE` is positive, then the magnet has linear fringe fields of length `FFRINGE*L/2` at each end. That is, the total length of fringe field from both ends combined is `FFRINGE*L`.

Depending on the value of `FRINGE_TYPE`, the fringe fields are modeled as contained within the length `L` ("inset" type) or extending symmetrically outside the length `L` ("fixed-strength" type).

For "inset" type fringe fields, the length of the "hard core" part of the quadrupole is `L*(1-FFRINGE)`. For "fixed-strength" type fringe fields, the length of the hard core is `L*(1-FFRINGE/2)`. In the latter case, the fringe gradient reaches 50% of the hard core value at the nominal boundaries of the magnet. This means that the integrated strength of the magnet does not change as the `FFRINGE` parameter is varied. This is not the case with "inset" type fringe fields.

Misalignments

There are three modes for implementing alignment errors. Which is used is controlled by the value of the `MALIGN_METHOD` parameter:

- `MALIGN_METHOD=0` — This selects the original method, which was the only one available before version 2021.1. The misalignment is referenced to the entrance face. The `YAW` and `PITCH` parameters are ignored.

- `MALIGN_METHOD=1` — This selects a linearized method based on M. Venturini's work [58], with misalignment referenced to the entrance face. The `YAW` and `PITCH` parameters are implemented.

- MALIGN_METHOD=2 — This selects a linearized method based on M. Venturini's work [58], with misalignment referenced to the magnet center. The YAW and PITCH parameters are implemented.

- MALIGN_METHOD=3 — This selects an exact method based on M. Venturini's work [58], with misalignment referenced to the entrance face. The YAW and PITCH parameters are implemented.

- MALIGN_METHOD=4 — This selects an exact method based on M. Venturini's work [58], with misalignment referenced to the magnet center. The YAW and PITCH parameters are implemented.

For elements with non-zero TILT, error displacements and rotations are performed in the lab frame.

10.80 QUFRINGE—An element consisting of a linearly increasing or decreasing quadrupole field.

An element consisting of a linearly increasing or decreasing quadrupole field.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K1	$1/M^2$	double	0.0	peak geometric strength
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE		double	0.0	fractional strength error
DIRECTION		long	0	1=entrance, -1=exit
ORDER		long	0	matrix order
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.81 RAMPP—A momentum-ramping element that changes the central momentum according to an SDDS-format file of the momentum factor vs time in seconds.

A momentum-ramping element that changes the central momentum according to an SDDS-format file of the momentum factor vs time in seconds.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving momentum factor vs time
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

Explanation of <filename>=<x>+<y> format: Several elements in `elegant` make use of data from external files to provide input waveforms. The external files are SDDS files, which may have many columns. In order to provide a convenient way to specify both the filename and the columns to use, we frequently employ <filename>=<x>+<y> format for the parameter value. For example, if the parameter value is `waveform.sdds=t+A`, then it means that columns `t` and `A` will be taken from file `waveform.sdds`. The first column is always the independent variable (e.g., time, position, or frequency), while the second column is the dependent quantity.

10.82 RAMPRF—A voltage-, phase-, and/or frequency-ramped RF cavity, implemented like RFCA.

A voltage-, phase-, and/or frequency-ramped RF cavity, implemented like RFCA.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
VOLT	V	double	0.0	nominal voltage
PHASE	DEG	double	0.0	nominal phase
FREQ	Hz	double	500000000	nominal frequency
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
VOLT_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving voltage waveform factor vs time
PHASE_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving phase offset vs time (requires FREQ_WAVEFORM)
FREQ_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving frequency-factor vs time (requires PHASE_WAVEFORM)
FIDUCIAL		STRING	NULL	mode for determining fiducial arrival time (light, tmean, first, pmaximum)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

Explanation of <filename>=<x>+<y> format: Several elements in `elegant` make use of data from external files to provide input waveforms. The external files are SDDS files, which may have many columns. In order to provide a convenient way to specify both the filename and the columns to use, we frequently employ <filename>=<x>+<y> format for the parameter value. For example, if the parameter value is `waveform.sdds=t+A`, then it means that columns `t` and `A` will be taken from file `waveform.sdds`. The first column is always the independent variable (e.g., time, position, or frequency), while the second column is the dependent quantity.

10.83 RBEN—A rectangular dipole, implemented as a SBEND with edge angles, up to 2nd order. Use CSBEND for symplectic tracking.

A rectangular dipole, implemented as a SBEND with edge angles, up to 2nd order. Use CSBEND for symplectic tracking.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	magnet (straight) length
ANGLE	RAD	double	0.0	bend angle
K1	$1/M^2$	double	0.0	geometric focusing strength
E1	RAD	double	0.0	entrance edge angle
E2	RAD	double	0.0	exit edge angle
TILT	RAD	double	0.0	rotation about incoming longitudinal axis
K2	$1/M^3$	double	0.0	geometric sextupole strength
H1	$1/M$	double	0.0	entrance pole-face curvature
H2	$1/M$	double	0.0	exit pole-face curvature
HGAP	M	double	0.0	half-gap between poles
FINT		double	0.5	edge-field integral
DX	M	double	0.0	misaligment of entrance
DY	M	double	0.0	misalignment of entrance
DZ	M	double	0.0	misalignment of entrance
FSE		double	0.0	fractional strength error of all components
FSE_DIPOLE		double	0.0	fractional strength error of dipole component
FSE_QUADRUPOLE		double	0.0	fractional strength error of quadrupole component
ETILT	RAD	double	0.0	error rotation about incoming longitudinal axis
ETILT_SIGN		short	1	Sign of ETILT relative to TILT. -1 is the old convention prior to 2020.5
EDGE1_EFFECTS		short	1	include entrance edge effects?
EDGE2_EFFECTS		short	1	include exit edge effects?
ORDER		short	0	matrix order
EDGE_ORDER		short	0	edge matrix order
TRANSPORT		short	0	use (incorrect) TRANSPORT equations for T436 of edge?
USE_BN		short	0	use B1 and B2 instead of K1 and K2 values?

A rectangular dipole, implemented as a SBEND with edge angles, up to 2nd order. Use CSBEND for symplectic tracking.

Parameter Name	Units	Type	Default	Description
B1	$1/M$	double	0.0	K1 = B1/rho, where rho is bend radius
B2	$1/M^2$	double	0.0	K2 = B2/rho
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

When adding errors, care should be taken to choose the right parameters. The `FSE` and `ETILT` parameters are used for assigning errors to the strength and alignment relative to the ideal values given by `ANGLE` and `TILT`. One can also assign errors to `ANGLE` and `TILT`, but this has a different meaning: in this case, one is assigning errors to the survey itself. The reference beam path changes, so there is no orbit/trajectory error. The most common thing is to assign errors to `FSE` and `ETILT`. Note that when adding errors to `FSE`, the error is assumed to come from the power supply, which means that multipole strengths also change.

Special note about splitting dipoles: when dipoles are long, it is common to want to split them into several pieces, to get a better look at the interior optics. When doing this, care must be exercised not to change the optics. `elegant` has some special features that are designed to reduce or manage potential problems. At issue is the need to turn off edge effects between the portions of the same dipole.

First, one can simply use the `divide_elements` command to set up the splitting. Using this command, `elegant` takes care of everything.

Second, one can use a series of dipoles *with the same name*. In this case, elegant automatically turns off interior edge effects. This is true when the dipole elements directly follow one another or are separated by a MARK element.

Third, one can use a series of dipoles with different names. In this case, you must also use the `EDGE1_EFFECTS` and `EDGE2_EFFECTS` parameters to turn off interior edge effects.

10.84 RCOL—A rectangular collimator.

A rectangular collimator.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
X_MAX	M	double	0.0	half-width in x
Y_MAX	M	double	0.0	half-width in y
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
OPEN_SIDE		STRING	NULL	which side, if any, is open (+x, -x, +y, -y)
INVERT		short	0	If non-zero, particles inside the aperture are lost while those outside are transmitted.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.85 RECIRC—An element that defines the point to which particles recirculate in multi-pass tracking

An element that defines the point to which particles recirculate in multi-pass tracking
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
I_RECIRC_ELEMENT		long	0	
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.86 REFLECT—Reflects the beam back on itself, which is useful for multiple beamline matching.

Reflects the beam back on itself, which is useful for multiple beamline matching.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
DUMMY		long	0	
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.87 REMCOR—An element to remove correlations from the tracked beam to simulate certain types of correction.

An element to remove correlations from the tracked beam to simulate certain types of correction.
Parallel capable? : no
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
X		short	1	remove correlations in x?
XP		short	1	remove correlations in x'?
Y		short	1	remove correlations in y?
YP		short	1	remove correlations in y'?
WITH		short	6	coordinate to remove correlations with (1,2,3,4,5,6)=(x,x',y,y',s,dP/Po)
ONCE_ONLY		short	0	compute correction only for first beam, apply to all?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.88 RFCA—An rf cavity with exact phase dependence, first order in transverse coordinates.

An rf cavity with exact phase dependence, first order in transverse coordinates.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
VOLT	V	double	0.0	peak voltage
PHASE	DEG	double	0.0	phase
FREQ	Hz	double	500000000	frequency
Q		double	0.0	cavity Q (for cavity that charges up to given voltage from 0)
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
CHANGE_P0		short	0	does cavity change central momentum?
CHANGE_T		short	0	set to 1 for long runs to avoid rounding error in phase
FIDUCIAL		STRING	NULL	mode for determining fiducial arrival time (light, tmean, first, pmaximum)
END1_FOCUS		short	0	include focusing at entrance?
END2_FOCUS		short	0	include focusing at exit?
STANDING_WAVE		short	0	Use standing-wave cavity model instead of default traveling wave?
BODY_FOCUS_MODEL		STRING	NULL	None (default), SRS or TW1.
N_KICKS		long	0	Number of kicks to use for kick method. Set to zero for matrix method.
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
T_REFERENCE	S	double	-1	arrival time of reference particle
LINEARIZE		short	0	Linearize phase dependence?
LOCK_PHASE		short	0	Lock phase to given value regardless of bunch centroid motion?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The phase convention is as follows, assuming a positive rf voltage: `PHASE=90` is the crest for acceleration. `PHASE=180` is the stable phase for a storage ring above transition without energy losses.

Body focus models: By default, no body focusing (i.e., focusing along the cavity) is included. Two options are included:

SRS — Simplified Rosenzweig-Serafini, based on Rosenzweig and Serafini, Phys. Rev. E 49 (2), 1599. As suggested by N. Towne (NSLS), I simplified this to assume a pure pi-mode standing wave. To use this, one must also set `STANDING_WAVE=1`.

TW1 — Traveling Wave 1, which includes the radial electric and azimuthal magnetic field components for a pure traveling wave given by

$$E_z = E_0 \sin(\omega t - kz + \phi) \tag{129}$$

To use this, one must set `STANDING_WAVE=0` (the default). One must also have `N_KICKS`≥ 10.

The `CHANGE_T` parameter may be needed for reasons that stem from `elegant`'s internal use of the total time-of-flight as the longitudinal coordinate. If the accelerator is very long or a large number of turns are being tracked, rounding error may affect the simulation, introducing spurious phase jumps. By setting `CHANGE_T=1`, you can force `elegant` to modify the time coordinates of the particles to subtract off NT_{rf}, where T_{tf} is the rf period and $N = \lfloor t/T_{tf} + 0.5 \rfloor$. If you are tracking a ring with rf at some harmonic h of the revolution frequency, this will result in the time coordinates being relative to the ideal revolution period, $T_{rf} * h$. If you have multiple rf cavities in a ring, you need only use this feature on one of them. Also, you can use `CHANGE_T=1` if you simply prefer to have the offset time coordinates in output files and analysis.

N.B.: *Do not use* `CHANGE_T=1` *if you have rf cavities that are not at harmonics of one another or if you have other time-dependent elements that are not resonant. Also, if you have harmonic cavities, only use CHANGE˙T on the cavity with the lowest frequency. Failure to follow these rules can result in wrong results and crashes.*

10.89 RFCW—A combination of RFCA, WAKE, TRWAKE, and LSCDRIFT.

A combination of RFCA, WAKE, TRWAKE, and LSCDRIFT.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
CELL_LENGTH	M	double	0.0	cell length (used to scale wakes, which are assumed to be given for a cell, according to L/CELL_LENGTH)
VOLT	V	double	0.0	voltage
PHASE	DEG	double	0.0	phase
FREQ	Hz	double	500000000	frequency
Q		double	0.0	cavity Q (for cavity that charges up to voltage from 0)
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
CHANGE_P0		long	0	does element change central momentum?
CHANGE_T		long	0	see RFCA documentation
FIDUCIAL		STRING	NULL	mode for determining fiducial arrival time (light, tmean, first, pmaximum)
END1_FOCUS		short	0	include focusing at entrance?
END2_FOCUS		short	0	include focusing at exit?
STANDING_WAVE		short	0	Use standing-wave cavity model instead of default traveling wave?
BODY_FOCUS_MODEL		STRING	NULL	None (default) or SRS (simplified Rosenzweig/Serafini for standing wave)
N_KICKS		long	0	Number of kicks to use for kick method. Set to zero for matrix method.
ZWAKE		long	1	If zero, longitudinal wake is turned off.
TRWAKE		long	1	If zero, transverse wakes are turned off.
WAKEFILE		STRING	NULL	name of file containing Green functions

A combination of RFCA, WAKE, TRWAKE, and LSCDRIFT.

Parameter Name	Units	Type	Default	Description
ZWAKEFILE		STRING	NULL	if WAKEFILE=NULL, optional name of file containing longitudinal Green function
TRWAKEFILE		STRING	NULL	if WAKEFILE=NULL, optional name of file containing transverse Green functions
TCOLUMN		STRING	NULL	column containing time data
WXCOLUMN		STRING	NULL	column containing x Green function
WYCOLUMN		STRING	NULL	column containing y Green function
WZCOLUMN		STRING	NULL	column containing longitudinal Green function
N_BINS		long	0	number of bins for current histogram
INTERPOLATE		long	0	interpolate wake?
SMOOTHING		long	0	Use Savitzky-Golay filter to smooth current histogram?
SG_HALFWIDTH		long	4	Savitzky-Golay filter half-width for smoothing
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
T_REFERENCE	S	double	-1	arrival time of reference particle
LINEARIZE		long	0	Linearize phase dependence?
LSC		long	0	Include longitudinal space-charge impedance?
LSC_BINS		long	1024	Number of bins for LSC calculations
LSC_INTERPOLATE		long	1	Interpolate computed LSC wake?
LSC_LOW_FREQUENCY_CUTOFF0		double	-1	Highest spatial frequency at which low-frequency cutoff filter is zero. If not positive, no low-frequency cutoff filter is applied. Frequency is in units of Nyquist (0.5/binsize).

A combination of RFCA, WAKE, TRWAKE, and LSCDRIFT.

Parameter Name	Units	Type	Default	Description
LSC_LOW_FREQUENCY_CUTOFF1		double	-1	Lowest spatial frequency at which low-frequency cutoff filter is 1. If not given, defaults to LOW_FREQUENCY_CUTOFF1.
LSC_HIGH_FREQUENCY_CUTOFF0		double	-1	Spatial frequency at which smoothing filter begins for LSC. If not positive, no frequency filter smoothing is done. Frequency is in units of Nyquist (0.5/binsize).
LSC_HIGH_FREQUENCY_CUTOFF1		double	-1	Spatial frequency at which smoothing filter is 0 for LSC. If not given, defaults to HIGH_FREQUENCY_CUTOFF0.
LSC_RADIUS_FACTOR		double	1.7	LSC radius is (Sx+Sy)/2*RADIUS_FACTOR
WAKES_AT_END		long	0	Do wake kicks at end of segment (for backward compatibility)?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is a combination of the `RFCA`, `WAKE`, and `TRWAKE` elements. As such, it provides combined simulation of an rf cavity with longitudinal and transverse wakes, as well as longitudinal space charge.

For the wakes, the input files and their interpretation are identical to `WAKE` and `TRWAKE`, except that the transverse and longitudinal wakes are interpreted as the wakes for a single cell of length given by the `CELL_LENGTH` parameter.

Users should read the entries for `WAKE`, `TRWAKE`, and `RFCA` for more details on this element.

This element simulates longitudinal space charge using the method described in [22]. This is based on the longitudinal space charge impedance per unit length

$$Z_{lsc}(k) = \frac{iZ_0}{\pi k r_b^2}\left[1 - \frac{k r_b}{\gamma} K_1\left(\frac{k r_b}{\gamma}\right)\right] \tag{130}$$

N.B.: *Do not use* `CHANGE_T=1` *if you have rf cavities that are not at harmonics of one another or if you have other time-dependent elements that are not resonant. Also, if you have harmonic cavities, only use CHANGE˙T on the cavity with the lowest frequency. Failure to follow these rules can result in wrong results and crashes.*

10.90 RFDF—A simple traveling or standing wave deflecting RF cavity.

A simple traveling or standing wave deflecting RF cavity.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
PHASE	DEG	double	0.0	phase
TILT	RAD	double	0.0	rotation about longitudinal axis
FREQUENCY	HZ	double	2856000000	frequency
VOLTAGE	V	double	0.0	voltage
FSE		double	0.0	Fractional Strength Error
B2		double	0.0	Normalized sextupole strength, kick=(1+b2*(x$\hat{2}$-y$\hat{2}$)/2)...
TIME_OFFSET	S	double	0.0	time offset (adds to phase)
N_KICKS		long	0	number of kicks (0=autoscale)
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
STANDING_WAVE		short	0	If nonzero, then cavity is standing wave.
VOLTAGE_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving voltage waveform factor vs time
VOLTAGE_PERIODIC		short	0	If non-zero, voltage waveform is periodic with period given by time span.
ALIGN_WAVEFORMS		short	0	If non-zero, waveforms' t=0 is aligned with first bunch arrival time.
VOLTAGE_NOISE		double	0.0	Rms fractional noise level for voltage.
PHASE_NOISE	DEG	double	0.0	Rms noise level for phase.
GROUP_VOLTAGE_NOISE		double	0.0	Rms fractional noise level for voltage linked to group.
GROUP_PHASE_NOISE	DEG	double	0.0	Rms noise level for phase linked to group.
VOLTAGE_NOISE_GROUP		long	0	Group number for voltage noise.

A simple traveling or standing wave deflecting RF cavity.

Parameter Name	Units	Type	Default	Description
PHASE_NOISE_GROUP		long	0	Group number for phase noise.
START_PASS		long	-1	If non-negative, pass on which to start modeling cavity.
END_PASS		long	-1	If non-negative, pass on which to end modeling cavity.
START_PID		long	-1	If non-negative, lowest particle ID to which deflection is applied.
END_PID		long	-1	If non-negative, highest particle ID to which deflection is applied.
DRIFT_MATRIX		short	0	If non-zero, calculations involving matrices assume this element is a drift space.
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
MAGNETIC_DEFLECTION		short	0	If non-zero, deflection is assumed to be performed by a magnetic field, rather than electric field (default).
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This cavity provides a transverse deflection that is constant as a function of transverse coordinates. It is probably the best model for a real cavity, because real cavities contain a mixture of TM- and TE-like modes that result in a uniform deflection.

For simplicity of use, the deflection is specified as a voltage, even though it originates in a magnetic field. The magnetic field is

$$B = B_0\hat{y}\cos\omega t \tag{131}$$

The corresponding electric field is obtained from Faraday's law (MKS units)

$$\left(\nabla \times \vec{E}\right)_y = -\left(\frac{\partial \vec{B}}{\partial t}\right)_y. \tag{132}$$

Assuming $E_x = E_y = 0$, we have

$$E_z = B_0\omega x \sin\omega t. \tag{133}$$

The change in momenta (in units of mc) in passing through a slice of length ΔL is

$$\Delta p_x = \frac{qB_0\Delta L}{mc}\cos\omega t \tag{134}$$

$$\Delta p_y = 0 \tag{135}$$

$$\Delta p_z = \frac{qB_0\omega x\Delta L}{mc^2}\sin\omega t \tag{136}$$

If we want to think in terms of a deflecting voltage, we can re-write this as

$$\Delta p_x = \frac{qV}{mc^2}\cos\omega t \tag{137}$$

$$\Delta p_y = 0 \tag{138}$$

$$\Delta p_z = \frac{qV}{mc^2}kx\sin\omega t, \tag{139}$$

where $k = \omega/c$.

Explanation of <filename>=<x>+<y> format: Several elements in `elegant` make use of data from external files to provide input waveforms. The external files are SDDS files, which may have many columns. In order to provide a convenient way to specify both the filename and the columns to use, we frequently employ <filename>=<x>+<y> format for the parameter value. For example, if the parameter value is `waveform.sdds=t+A`, then it means that columns `t` and `A` will be taken from file `waveform.sdds`. The first column is always the independent variable (e.g., time, position, or frequency), while the second column is the dependent quantity.

10.91 RFMODE—A simulation of a beam-driven TM monopole mode of an RF cavity.

A simulation of a beam-driven TM monopole mode of an RF cavity.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
RA	Ohm	double	0.0	shunt impedance, Ra=V2̂/P
RS	Ohm	double	0.0	shunt impedance (Rs=Ra/2)
Q		double	0.0	cavity Q
FREQ	Hz	double	0.0	Resonant frequency of the cavity mode
CHARGE	C	double	0.0	Deprecated—use CHARGE element)
INITIAL_V	V	double	0.0	initial beam-loading voltage
INITIAL_PHASE	RAD	double	0.0	initial beam-loading phase
INITIAL_T	S	double	0.0	time at which INITIAL_V and INITIAL_PHASE held
BETA		double	0.0	normalized load impedance
BIN_SIZE	S	double	0.0	bin size for current histogram
N_BINS		long	20	number of bins for current histogram
INTERPOLATE		long	0	if non-zero, interpolate voltage within bins
PRELOAD		long	0	preload cavity with steady-state field
PRELOAD_CHARGE	C	double	0.0	beam charge used for preloading calculations
PRELOAD_FACTOR		double	1	multiply preloaded field by this value
PRELOAD_HARMONIC		long	0	If detuning from harmonic is greater than half the revolution frequency, automatic determination of the rf harmonic will fail. Give the harmonic explicitly with this parameter.
RIGID_UNTIL_PASS		long	0	don't affect the beam until this pass
DETUNED_UNTIL_PASS		long	0	cavity is completely detuned until this pass

`RFMODE` continued

A simulation of a beam-driven TM monopole mode of an RF cavity.

Parameter Name	Units	Type	Default	Description
SAMPLE_INTERVAL		long	1	passes between samples to RECORD file
FLUSH_INTERVAL		long	1000	samples between flushing output to RECORD file
RECORD		STRING	NULL	output file for cavity fields
SINGLE_PASS		long	0	if nonzero, don't accumulate field from pass to pass
PASS_INTERVAL		long	1	interval in passes at which to apply PASS_INTERVAL times the field (may increase speed)
FREQ_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving frequency/f0 vs time, where f0 is the frequency given with the FREQ parameter
Q_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving qualityFactor/Q0 vs time, where Q0 is the quality factor given the the Q parameter.
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the impedance to full strength.
BINLESS		long	0	If nonzero, use algorithm that doesn't requiring binning. Best for few particles, widely spaced.
RESET_FOR_EACH_STEP		long	1	If nonzero, voltage and phase are reset for each simulation step.
LONG_RANGE_ONLY		long	0	If nonzero, induced voltage from present turn does not affect bunch. Results are not self-consistent!

A simulation of a beam-driven TM monopole mode of an RF cavity.

Parameter Name	Units	Type	Default	Description
ALLOW_UNBINNED_PARTICLES		long	0	If nonzero, will keep running even if some particles fall outside the binning region. Use with caution!
N_CAVITIES		long	1	effect is multiplied by this number, simulating N identical cavities
BUNCHED_BEAM_MODE		long	1	If 1, then do calculations bunch-by-bunch. If >1, use pseudo bunches.
BUNCH_INTERVAL	S	double	0.0	For pseudo-bunch mode, time between bunches.
DRIVE_FREQUENCY	Hz	double	0.0	drive frequency from generator. If zero, no generator voltage is applied.
V_SETPOINT	V	double	0.0	setpoint for total cavity voltage
PHASE_SETPOINT	DEG	double	0.0	setpoint for total cavity phase
UPDATE_INTERVAL		long	1	update interval of feedback in units of rf period
READ_OFFSET		long	0	Offset in buckets of point at which voltage and phase are read for feedback relative to the first bunch passage. A positive value corresponds to reading before bunch passage.
ADJUSTMENT_START		long	0	Pass on which to begin adjustment of the effective voltage setpoint.
ADJUSTMENT_END		long	0	Pass on which to stop adjustment of the effective voltage setpoint.
ADJUSTMENT_INTERVAL		long	100	Interval in passes between adjustment of the effective voltage setpoint.

A simulation of a beam-driven TM monopole mode of an RF cavity.

Parameter Name	Units	Type	Default	Description
ADJUSTMENT_FRACTION		double	0.0	Fraction of voltage setpoint error taken out on each adjustment step
AMPLITUDE_FILTER		STRING	NULL	IIR filter specification for amplitude feedback
PHASE_FILTER		STRING	NULL	IIR filter specification for phase feedback
IN_PHASE_FILTER		STRING	NULL	IIR filter specification for in-phase component feedback
QUADRATURE_FILTER		STRING	NULL	IIR filter specification for quadrature component feedback
FEEDBACK_RECORD		STRING	NULL	output file for feedback data
MUTE_GENERATOR		long	-1	If nonnegative, gives the pass on which to mute the generator. This simulates an rf trip.
GENERATOR_FACTOR		double	1	Multiplies the generator current by the specified factor.
NOISE_ALPHA_GEN		STRING	NULL	<filename>=<x>+<y> specifying alpha(t) for generator noise.
NOISE_PHI_GEN		STRING	NULL	<filename>=<x>+<y> specifying dphi(t) for generator noise, in radians.
NOISE_ALPHA_V		STRING	NULL	<filename>=<x>+<y> specifying alpha(t) for voltage noise.
NOISE_PHI_V		STRING	NULL	<filename>=<x>+<y> specifying dphi(t) for voltage noise, in radians.
NOISE_I_GEN		STRING	NULL	<filename>=<x>+<y> specifying ni(t) for in-phase generator noise.

A simulation of a beam-driven TM monopole mode of an RF cavity.

Parameter Name	Units	Type	Default	Description
NOISE_Q_GEN		STRING	NULL	<filename>=<x>+<y> specifying nq(t) for quadrature generator noise.
NOISE_I_V		STRING	NULL	<filename>=<x>+<y> specifying ei(t) for in-phase voltage noise.
NOISE_Q_V		STRING	NULL	<filename>=<x>+<y> specifying eq(t) for quadrature voltage noise.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a beam-driven monopole mode cavity using the fundamental theorem of beam loading and phasor rotation. In addition, a generator-driven field may be included using a feedback system [44].

Note on phase conventions: the phase convention for the `PHASE_SETPOINT` parameter of `RFMODE` is the same as for the `PHASE` parameter of `RFCA`. However, in the output files from `RFMODE`, i.e., the files requested with the `RECORD` and `FEEDBACK_RECORD` parameters, a different convention is used, which differs by -90 degrees from the `PHASE_SETPOINT` parameter.

The feedback implementation uses either amplitude and phase feedback or else in-phase and quadrature feedback. Figure 3 shows the model used for the feedback system. More information is available in [44].

Rf feedback is active when a non-zero value is given for `DRIVE_FREQUENCY` and when either `AMPLITUDE_FILTER` and `PHASE_FILTER` or else `IN_PHASE_FILTER` and `QUADRATURE_FILTER` are given. These parameters name SDDS files that define filters in the z domain

$$H(z) = \frac{\sum_{i=0}^{m} b_i z^{-i}}{\sum_{i=0}^{r} a_i z^{-i}} \tag{140}$$

which translates into the discrete time domain as

$$y[n] = -\frac{1}{a_0}\sum_{i=1}^{r} a_i y_{r-i} + \frac{1}{a_0}\sum_{i=0}^{m} b_i x_{n-i}, \tag{141}$$

where $y[n]$ is the filter output (e.g., additional generator current) and $x[n]$ is the filter input (e.g., the voltage error signal). The filter files must each contain two columns:

- `denominator` — The a_i filter coefficients. These should generally sum to zero or nearly zero.

- `numerator` — The b_i filter coefficients. These are generally very small.

Each file can consist of up to four pages, with each page representing one filter in a parallel filter bank formed from all pages. The output of the four stages is added to obtain the control signal. The number of rows in the page is given by the larger of r or m; in the case of multiple pages, the number of rows in each page is given by the largest r or m from the highest order filter amongst the pages, by simply stuffing the rows of the other pages with zeros, i.e., rows representing orders of z down to z^{-l} where l is the largest r or m of the filter pages.

For example, a digital low-pass filter with DC gain of k can be specified as

$$y[1] = e^{-pT} y[0] + k(1 - e^{-pT}) x[0], \tag{142}$$

where p is the frequency of the pole and T is the sample interval.

The feedback loop reads the cavity state and acts on the generator at a fixed interval (in buckets) of `UPDATE_INTERVAL`. The timing of this activity is aligned to the arrival time of the first bunch in the `RFMODE` element. By default (`READ_OFFSET`=0), the timing is such that the state is read just before the next arrival of that bunch; in particular, it is 180 degrees ahead of that arrival. If bunches are equally spaced by, say N_b buckets, the `UPDATE_INTERVAL` parameter should ideally be mN_b, where $m > 0$ is an integer. This ensures that the state is read at a fixed timing relative to the bunches.

The rf feedback feature makes use of the voltage amplitude measured when there is no bunch present. The `RECORD` file shows the voltage seen by the beam, computed by averaging over the voltage for each particle. These may deviate by values from a few percent to of order ten percent, depending on the loss factor for the cavity and the number of bunchess; this is caused by the fact that the rate at which an intense bunch removes energy from the cavity will typically, albeit briefly, exceed the power from the generator. To reduce the impact of this effect, one may use the `ADJUSTMENT_FRACTION`, `ADJUSTMENT_START`, and `ADJUSTMENT_INTERVAL` parameters to modify the voltage setpoint. If `ADJUSTMENT_FRACTION` is non-zero, then for every `ADJUSTMENT_INTERVAL`th pass after the `ADJUSTMENT_START`th pass, the voltage setpoint will be adjusted based on a comparison of the bunch-averaged voltage to the user's setpoint. E.g., if the bunch-averaged voltage is 100 V too low and `ADJUSTMENT_FRACTION` is 0.1, the voltage setpoint will be raised by 10 V. Users should note that if `ADJUSTMENT_FRACTION` is too large or `ADJUSTMENT_INTERVAL` is too small, the system may be unstable.

Normally, the field dumped in the cavity by one particle affects trailing particles in the same turn. However, if one is also using a `WAKE` or `ZLONGIT` element to simulate the short-range wake of the cavity, this would be double-counting. In that case, one can use `LONG_RANGE_ONLY=1` to suppress the same-turn effects of the `RFMODE` element.

Two output files are available: the `RECORD` file includes bunch-by-bunch data on the beam-induced fields and the total cavity fields. The `FEEDBACK_RECORD` file includes tick-by-tick data from the feedback system simulation; *writing this file this can significantly degrade performance.*

NB: when `BUNCHED_BEAM_MODE` is set to a value other than 1, in order to obtain the effect of several bunches while tracking only one bunch, the total charge set with the `TOTAL` parameter of the `CHARGE` element should equal the charge in a single bunch, not the entire beam. However, when `BUNCHED_BEAM_MODE`=1 (allowing an indeterminant number of bunches to be actually present), then `TOTAL` should be the total for all bunches together.

Explanation of <filename>=<x>+<y> format: Several elements in `elegant` make use

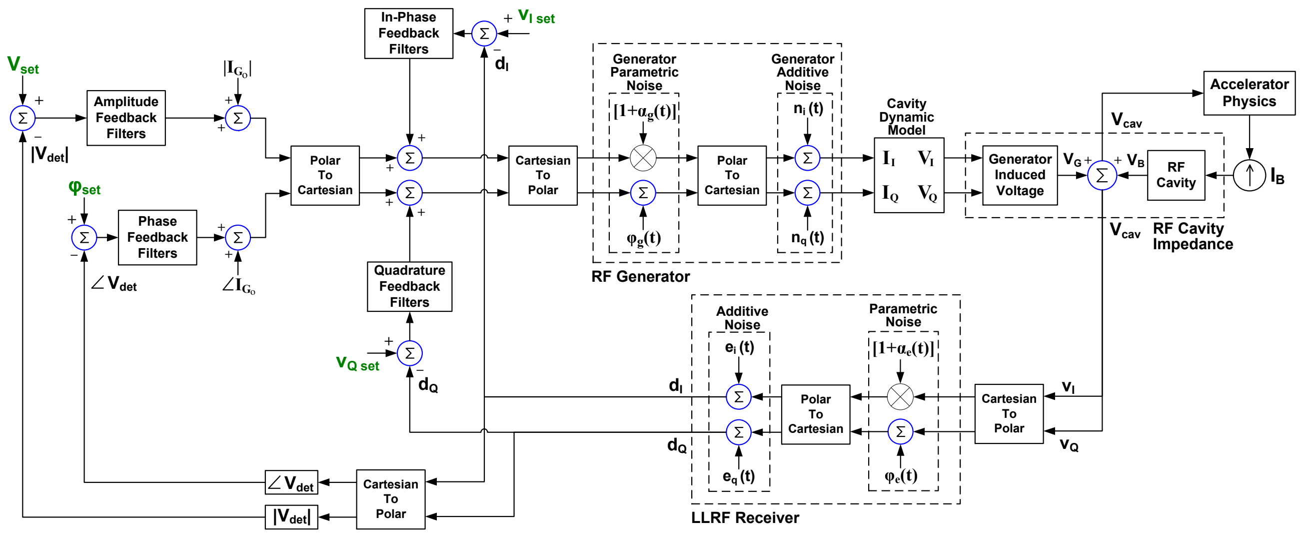

Figure 3: Rf feedback model used by the RFMODE element. Courtesy T. Berenc, Argonne National Laboratory.

of data from external files to provide input waveforms. The external files are SDDS files, which may have many columns. In order to provide a convenient way to specify both the filename and the columns to use, we frequently employ <filename>=<x>+<y> format for the parameter value. For example, if the parameter value is `waveform.sdds=t+A`, then it means that columns `t` and `A` will be taken from file `waveform.sdds`. The first column is always the independent variable (e.g., time, position, or frequency), while the second column is the dependent quantity.

10.92 RFTM110—Tracks through a TM110-mode (deflecting) rf cavity with all magnetic and electric field components. NOT RECOMMENDED—See below.

Tracks through a TM110-mode (deflecting) rf cavity with all magnetic and electric field components.
NOT RECOMMENDED—See below.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
PHASE	DEG	double	0.0	phase
TILT	RAD	double	0.0	rotation about longitudinal axis
FREQUENCY	HZ	double	2856000000	frequency
VOLTAGE	V	double	0.0	peak deflecting voltage
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
VOLTAGE_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving voltage waveform factor vs time
VOLTAGE_PERIODIC		short	0	If non-zero, voltage waveform is periodic with period given by time span.
ALIGN_WAVEFORMS		short	0	If non-zero, waveforms' t=0 is aligned with first bunch arrival time.
VOLTAGE_NOISE		double	0.0	Rms fractional noise level for voltage.
PHASE_NOISE	DEG	double	0.0	Rms noise level for phase.
GROUP_VOLTAGE_NOISE		double	0.0	Rms fractional noise level for voltage linked to group.
GROUP_PHASE_NOISE	DEG	double	0.0	Rms noise level for phase linked to group.
VOLTAGE_NOISE_GROUP		long	0	Group number for voltage noise.
PHASE_NOISE_GROUP		long	0	Group number for phase noise.
START_PASS		long	-1	If non-negative, pass on which to start modeling cavity.
END_PASS		long	-1	If non-negative, pass on which to end modeling cavity.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

NB: Although this element is correct insofar as it uses the fields for a pure TM110 mode, it is recommended that the `RFDF` element be used instead. In a real deflecting cavity with entrance and exit tubes, the deflecting mode is a hybrid TE/TM mode, in which the deflection has no dependence on the radial coordinate.

To derive the field expansion, we start with some results from Jackson[17], section 8.7. The longitudinal electric field for a TM mode is just

$$E_z = -2iE_0\Psi(\rho,\phi)\cos\left(\frac{p\pi z}{d}\right)e^{-i\omega t}, \tag{143}$$

where p is an integer, d is the length of the cavity, and we use cylindrical coordinates (ρ, ϕ, z). The factor of $-2i$ represents a choice of sign and phase convention. We are interested in the TM110 mode, so we set $p = 0$. In this case, we have

$$E_x = E_y = 0 \tag{144}$$

and (using CGS units)

$$\vec{H} = -2iE_0\frac{i\epsilon\omega}{ck^2}\hat{z}\times\nabla\Psi e^{-i\omega t}. \tag{145}$$

For a cylindrical cavity, the function Ψ for the $m = 1$ aximuthal mode is

$$\Psi(\rho,\phi) = J_1(k\rho)\cos\phi, \tag{146}$$

where $k = x_{11}/R$, x_{11} is the first zero of $J_1(x)$, and R is the cavity radius. We don't need to know the cavity radius, since $k = \omega/c$, where ω is the resonant frequency. By choosing $\cos\phi$ for the aximuthal dependence, we'll get a magnetic field primarily in the vertical direction.

In MKS units, the magnetic field is

$$\vec{B} = \frac{2E_0}{kc}e^{-i\omega t}\left(\hat{\rho}\frac{J_1(k\rho)}{\rho}\sin\phi + \hat{\phi}\cos\phi\frac{\partial J_1(k\rho)}{\partial\rho}\right). \tag{147}$$

Using `mathematica`, we expanded these expressions to sixth order in $k*\rho$. Here, we present only the expressions to second order. Taking the real parts only, we now have

$$\begin{aligned} E_z &\approx E_0 k\rho\cos\phi\sin\omega t && (148)\\ cB_\rho &\approx E_0\left(1-\frac{k^2\rho^2}{8}\right)\sin\phi\cos\omega t && (149)\\ cB_\phi &\approx E_0\left(1-\frac{3k^2\rho^2}{8}\right)\cos\phi\cos\omega t && (150) \end{aligned}$$

The Cartesian components of $\vec{B}$ can be computed easily

$$\begin{aligned} cB_x &= cB_\rho\cos\phi - cB_\phi\sin\phi && (151)\\ &= \frac{E_0}{4}\rho^2k^2\cos\phi\sin\phi\cos\omega t && (152)\\ cB_y &= cB_\rho\sin\phi + cB_\phi\cos\phi && (153)\\ &= E_0\left(1-\frac{k^2\rho^2(2\cos^2\phi+1)}{8}\right)\cos\omega t && (154) \end{aligned}$$

The Lorentz force on an electron is $F = -eE_z\hat{z} - ec\vec{\beta}\times\vec{B}$, giving

$$\begin{aligned} F_x/e &= \beta_z cB_y && (155)\\ F_y/e &= -\beta_z cB_x && (156)\\ F_z/e &= -E_z - \beta_x cB_y + \beta_y cB_x && (157) \end{aligned}$$

We see that for $\rho \to 0$, we have $E_z = 0$, $B_x = 0$, and

$$cB_y = E_0 \cos \omega t. \tag{158}$$

Hence, for $\omega t = 0$ and $E_0 > 0$ we have $F_x > 0$. This explains our choice of sign and phase convention above. Indeed, owing to the factor of 2, we have a peak deflection of eE_0L/E, where L is the cavity length and E the beam energy. Thus, if $V = E_0L$ is specified in volts, and the beam energy expressed in electron volts, the deflection is simply the ratio of the two. As a result, we've chosen to parametrize the deflection strength simply by referring to the "deflecting voltage," V.

Explanation of <filename>=<x>+<y> format: Several elements in `elegant` make use of data from external files to provide input waveforms. The external files are SDDS files, which may have many columns. In order to provide a convenient way to specify both the filename and the columns to use, we frequently employ <filename>=<x>+<y> format for the parameter value. For example, if the parameter value is `waveform.sdds=t+A`, then it means that columns `t` and `A` will be taken from file `waveform.sdds`. The first column is always the independent variable (e.g., time, position, or frequency), while the second column is the dependent quantity.

10.93 RFTMEZ0—A TM-mode RF cavity specified by the on-axis Ez field.

A TM-mode RF cavity specified by the on-axis Ez field.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
FREQUENCY	HZ	double	2856000000	frequency
PHASE	RAD	double	0.0	phase
EZ_PEAK	V	double	0.0	Peak on-axis longitudinal electric field
TIME_OFFSET	S	double	0.0	time offset (adds to phase)
PHASE_REFERENCE		long	0	phase reference number (to link to other time-dependent elements)
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
ETILT	RAD	double	0.0	misalignment rotation about longitudinal axis
EPITCH	RAD	double	0.0	misalignment rotation about vertical axis. Ignored if MALIGN_METHOD=0
EYAW	RAD	double	0.0	misalignment rotation about horizontal axis. Ignored if MALIGN_METHOD=0
N_STEPS		long	100	number of steps (for nonadaptive integration)
RADIAL_ORDER		short	1	highest order in off-axis expansion
CHANGE_P0		short	0	does element change central momentum?
INPUTFILE		STRING	NULL	file containing Ez vs z at r=0
ZCOLUMN		STRING	NULL	column containing z values
EZCOLUMN		STRING	NULL	column containing Ez values
SOLENOID_FILE		STRING	NULL	file containing map of Bz and Br vs z and r. Each page contains values for a single r.
SOLENOID_ZCOLUMN		STRING	NULL	column containing z values for solenoid map.

A TM-mode RF cavity specified by the on-axis Ez field.

Parameter Name	Units	Type	Default	Description
SOLENOID_RCOLUMN		STRING	NULL	column containing r values for solenoid map. If omitted, data is assumed to be for r=0 and an on-axis expansion is performed.
SOLENOID_BZCOLUMN		STRING	NULL	column containing Bz values for solenoid map.
SOLENOID_BRCOLUMN		STRING	NULL	column containing Br values for solenoid map. If omitted, data is assumed to be for r=0 and an on-axis expansion is performed.
SOLENOID_FACTOR		double	1	factor by which to multiply solenoid fields.
SOLENOID_DX	M	double	0.0	misalignment
SOLENOID_DY	M	double	0.0	misalignment
SOLENOID_DZ	M	double	0.0	misalignment
SOLENOID_ETILT	RAD	double	0.0	misalignment
SOLENOID_EYAW	RAD	double	0.0	misalignment
SOLENOID_EPITCH	RAD	double	0.0	misalignment
BX_STRAY		double	0.0	Uniform stray horizontal field
BY_STRAY		double	0.0	Uniform stray vertical field
ACCURACY		double	0.0001	integration accuracy
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, non-adaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")
FIELD_TEST_FILE		STRING	NULL	filename for output of test fields (r=0)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.94 RIMULT—Multiplies radiation integrals by a given factor. Use to compute emittance for collection of various types of cells.

Multiplies radiation integrals by a given factor. Use to compute emittance for collection of various types of cells.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
FACTOR		double	1	factor
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.95 RMDF—A linearly-ramped electric field deflector, using an approximate analytical solution FOR LOW ENERGY PARTICLES.

A linearly-ramped electric field deflector, using an approximate analytical solution FOR LOW ENERGY PARTICLES.
Parallel capable? : no
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
TILT	RAD	double	0.0	rotation about longitudinal axis
RAMP_TIME	S	double	1e-09	length of ramp
VOLTAGE	V	double	0.0	full voltage
GAP	M	double	0.01	gap between plates
TIME_OFFSET	S	double	0.0	time offset of ramp start
N_SECTIONS		long	10	number of sections
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.96 ROTATE—An element that rotates the beam about the longitudinal axis.

An element that rotates the beam about the longitudinal axis.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
TILT	*RAD*	double	0.0	rotation about longitudinal axis
EXCLUDE_FLOOR		short	0	if non-zero, does not affect the floor coordinates
EXCLUDE_OPTICS		short	0	if non-zero, does not affect the optics (i.e., transfer matrix is unit matrix)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The sign convention for the `TILT` parameter is confusing on this element. In particular, a positive `TILT` rotates the beam counter-clockwise about the longitudinal axis. This is the opposite sense to rotations of elements, where a positive `TILT` rotates the element clockwise about the longitudinal axis.

Hence, if one wanted to rotate a series of elements by 0.1 rad, one could do the following:

```
ROT1: ROTATE,TILT=0.1
ROT2: ROTATE,TILT=-0.1
BL: line=(ROT1,...,ROT2)
```

The `TILT` value for `ROT1` is the same (including the sign) as the individual `TILT` values one would give to all the elements represented by `...`.

10.97 SAMPLE—An element that reduces the number of particles in the beam by interval-based or random sampling.

An element that reduces the number of particles in the beam by interval-based or random sampling.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
FRACTION		double	1	fraction to keep
INTERVAL		long	1	interval between sampled particles
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.98 SBEN—A sector dipole implemented as a matrix, up to 2nd order. Use CSBEND for symplectic tracking.

A sector dipole implemented as a matrix, up to 2nd order. Use CSBEND for symplectic tracking.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	arc length
ANGLE	RAD	double	0.0	bend angle
K1	$1/M^2$	double	0.0	geometric focusing strength
E1	RAD	double	0.0	entrance edge angle
E2	RAD	double	0.0	exit edge angle
TILT	RAD	double	0.0	rotation about incoming longitudinal axis
K2	$1/M^3$	double	0.0	geometric sextupole strength
H1	$1/M$	double	0.0	entrance pole-face curvature
H2	$1/M$	double	0.0	exit pole-face curvature
HGAP	M	double	0.0	half-gap between poles
FINT		double	0.5	edge-field integral
DX	M	double	0.0	misaligment of entrance
DY	M	double	0.0	misalignment of entrance
DZ	M	double	0.0	misalignment of entrance
FSE		double	0.0	fractional strength error of all components
FSE_DIPOLE		double	0.0	fractional strength error of dipole component
FSE_QUADRUPOLE		double	0.0	fractional strength error of quadrupole component
ETILT	RAD	double	0.0	error rotation about incoming longitudinal axis
ETILT_SIGN		short	1	Sign of ETILT relative to TILT. -1 is the old convention prior to 2020.5.
EDGE1_EFFECTS		short	1	include entrance edge effects?
EDGE2_EFFECTS		short	1	include exit edge effects?
ORDER		short	0	matrix order
EDGE_ORDER		short	0	edge matrix order
TRANSPORT		short	0	use (incorrect) TRANSPORT equations for T436 of edge?
USE_BN		short	0	use B1 and B2 instead of K1 and K2 values?

A sector dipole implemented as a matrix, up to 2nd order. Use CSBEND for symplectic tracking.

Parameter Name	Units	Type	Default	Description
B1	$1/M$	double	0.0	K1 = B1/rho, where rho is bend radius
B2	$1/M^2$	double	0.0	K2 = B2/rho
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

Some confusion may exist about the edge angles, particularly the signs. For a sector magnet, we have of course `E1=E2=0`. For a symmetric rectangular magnet, `E1=E2=ANGLE/2`. If `ANGLE` is negative, then so are `E1` and `E2`. To understand this, imagine a rectangular magnet with positive `ANGLE`. If the magnet is flipped over, then `ANGLE` becomes negative, as does the bending radius ρ. Hence, to keep the focal length of the edge $1/f = -\tan E_i/\rho$ constant, we must also change the sign of E_i.

When adding errors, care should be taken to choose the right parameters. The `FSE` and `ETILT` parameters are used for assigning errors to the strength and alignment relative to the ideal values given by `ANGLE` and `TILT`. One can also assign errors to `ANGLE` and `TILT`, but this has a different meaning: in this case, one is assigning errors to the survey itself. The reference beam path changes, so there is no orbit/trajectory error. The most common thing is to assign errors to `FSE` and `ETILT`. Note that when adding errors to `FSE`, the error is assumed to come from the power supply, which means that multipole strengths also change.

Special note about splitting dipoles: when dipoles are long, it is common to want to split them into several pieces, to get a better look at the interior optics. When doing this, care must be exercised not to change the optics. `elegant` has some special features that are designed to reduce or manage potential problems. At issue is the need to turn off edge effects between the portions of the same dipole.

First, one can simply use the `divide_elements` command to set up the splitting. Using this command, `elegant` takes care of everything.

Second, one can use a series of dipoles *with the same name*. In this case, elegant automatically turns off interior edge effects. This is true when the dipole elements directly follow one another or are separated by a MARK element.

Third, one can use a series of dipoles with different names. In this case, you must also use the `EDGE1_EFFECTS` and `EDGE2_EFFECTS` parameters to turn off interior edge effects.

10.99 SCATTER—A scattering element to add random values to particle coordinates.

A scattering element to add random values to particle coordinates.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
X	M	double	0.0	scattering amplitude for x
XP		double	0.0	scattering amplitude for x'
Y	M	double	0.0	scattering amplitude for y
YP		double	0.0	scattering amplitude for y'
DP		double	0.0	scattering amplitude for (p-pCentral)/pCentral
PROBABILITY		double	1	Probability that any particle will be selected for scattering.
STARTONPASS		long	0	Pass number to start on.
ENDONPASS		long	-1	Pass number to end on (inclusive). Ignored if negative.
DISTRIBUTION		STRING	gaussian	may be "gaussian" or "uniform"
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.100 SCRAPER—A collimating element that sticks into the beam from one side only. The directions 0, 1, 2, and 3 are from +x, +y, -x, and -y, respectively.

A collimating element that sticks into the beam from one side only. The directions 0, 1, 2, and 3 are from +x, +y, -x, and -y, respectively.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
XO	M	double	0.0	radiation length
ENERGY_DECAY		long	0	If nonzero, then particles will lose energy due to material using a simple exponential model.
ENERGY_STRAGGLE		long	0	Use simple-minded energy straggling model coupled with ENERGY_DECAY=1?
NUCLEAR_BREMSSTRAHLUNG		long	0	Model energy loss to nuclear bremsstrahlung? If enabled, set ENERGY_DECAY=0 to disable simpler model.
ELECTRON_RECOIL		long	0	If non-zero, electron recoil during Coulomb scattering is included (results in energy change).
Z		long	0	Atomic number
A	AMU	double	0.0	Atomic mass
RHO	KG/M^3	double	0.0	Density
PLIMIT		double	0.05	Probability cutoff for each slice
POSITION	M	double	0.0	position of edge
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
INSERT_FROM		STRING	NULL	direction from which inserted (+x, -x, x, +y, -y, y)
DIRECTION		long	-1	Deprecated. use INSERT_FROM.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The method used for material modeling is the same as that used for the `MATTER` element.

The `DIRECTION` parameter is deprecated and the more versatile `INSERT_FROM` parameter should be used instead. The values for the latter determine from which side the scraper is inserted. E.g., `INSERT_FROM="-x",POSITION=0.005` means the scraper is inserted from the negative x side and extends from $x = -\infty$ to $x = 0.005$m.

`INSERT_FROM="x"` or `INSERT_FROM="y"` means the scraper is inserted from both sides, in which case the interpretation of the position is different. E.g., `INSERT_FROM="x",POSITION=0.005` means that only the region $x : [-0.005, 0.005]$m is clear.

10.101 SCRIPT—An element that allows transforming the beam using an external script.

An element that allows transforming the beam using an external script.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	Length to be used for matrix-based operations such as twiss parameter computation.
COMMAND		STRING	NULL	SDDS-compliant command to apply to the beam. Use the sequence %i to represent the input filename and %o to represent the output filename.
USE_CSH		short	1	Use C-shell for execution (may be slower)?
VERBOSITY		short	0	Set the verbosity level.
RPN_PARAMETERS		short	0	If nonzero, then parameters from the script output file are loaded into RPN variables.
START_PASS		long	-1	Start script action on this pass. Before that, behaves like a drift space.
END_PASS		long	-1	End script action after this pass. Before that, behaves like a drift space.
PASS_INTERVAL		long	-1	Execute script only every Nth pass following START_PASS, including START_PASS. Otherwise, behaves like a drift space.
ON_PASS		long	-1	Perform script action only on this pass, overriding other pass controls. Other than that, behaves like a drift space.
DIRECTORY		STRING	NULL	Directory in which to place input and output files. If blank, the present working directory is used.
ROOTNAME		STRING	NULL	Rootname for use in naming input and output files. %s may be used to represent the run rootname.

SCRIPT continued

An element that allows transforming the beam using an external script.

Parameter Name	Units	Type	Default	Description
INPUT_EXTENSION		STRING	in	Extension for the script input file.
OUTPUT_EXTENSION		STRING	out	Extension for the script output file.
KEEP_FILES		short	0	If nonzero, then script input and output files are not deleted after use. By default, they are deleted.
DRIFT_MATRIX		short	0	If nonzero, then for non-tracking calculations the element is treated as a drift space.
USE_PARTICLE_ID		short	1	If nonzero, then the output file will supply particle IDs. Otherwise, particles are renumbered.
NO_NEW_PARTICLES		short	1	If nonzero, then no new particles will be added in the script output file.
DETERMINE_LOSSES_FROM_PID		short	1	If nonzero and if USE_PARTICLE_ID is nonzero, then particleID data from script output is used to determine which particles were lost.
SOFT_FAILURE		short	1	If output file does not exist or can't be read, consider all particles lost.
NP0		double	0.0	User-defined numerical parameter for command substitution for sequence %np0
NP1		double	0.0	User-defined numerical parameter for command substitution for sequence %np1
NP2		double	0.0	User-defined numerical parameter for command substitution for sequence %np2

An element that allows transforming the beam using an external script.

Parameter Name	Units	Type	Default	Description
NP3		double	0.0	User-defined numerical parameter for command substitution for sequence %np3
NP4		double	0.0	User-defined numerical parameter for command substitution for sequence %np4
NP5		double	0.0	User-defined numerical parameter for command substitution for sequence %np5
NP6		double	0.0	User-defined numerical parameter for command substitution for sequence %np6
NP7		double	0.0	User-defined numerical parameter for command substitution for sequence %np7
NP8		double	0.0	User-defined numerical parameter for command substitution for sequence %np8
NP9		double	0.0	User-defined numerical parameter for command substitution for sequence %np9
SP0		STRING	NULL	User-defined string parameter for command substitution for sequence %sp0
SP1		STRING	NULL	User-defined string parameter for command substitution for sequence %sp1
SP2		STRING	NULL	User-defined string parameter for command substitution for sequence %sp2
SP3		STRING	NULL	User-defined string parameter for command substitution for sequence %sp3
SP4		STRING	NULL	User-defined string parameter for command substitution for sequence %sp4

An element that allows transforming the beam using an external script.

Parameter Name	Units	Type	Default	Description
SP5		STRING	NULL	User-defined string parameter for command substitution for sequence %sp5
SP6		STRING	NULL	User-defined string parameter for command substitution for sequence %sp6
SP7		STRING	NULL	User-defined string parameter for command substitution for sequence %sp7
SP8		STRING	NULL	User-defined string parameter for command substitution for sequence %sp8
SP9		STRING	NULL	User-defined string parameter for command substitution for sequence %sp9
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element allows expanding `elegant` by using external scripts (or programs) as elements in a beamline. Here are requirements for the script:

- It must be executable from the commandline.
- It must read the initial particle distribution from an SDDS file. This file will have the usual columns that an `elegant` phase-space output file has, along with the parameter `Charge` giving the beam charge in Coulombs. The file will contain a single data page.
- It must write the final particle distribution to an SDDS file. This file should have all of the columns and parameters that appear in the initial distribution file. Additional columns and parameters will be ignored, as will all pages but the first.
- The `Charge` parameter in the file is used to determine the total beam charge; the script must ensure that this parameter is set correctly; when particles are lost or created, simply copying or retaining the value from the input file will not be correct. Normally, the charge per particle is constant in simulations. Hence, if `elegant` sees a change in charge per particle after the `SCRIPT` element, it issues a warning.

The `SCRIPT` element works best if the script accepts commandline arguments. In this case, the `COMMAND` parameter is used to provide a template for creating a command to run the script. The `COMMAND` string may contain the following substitutable fields:

1. `%i` — Will be replaced by the name of the input file to the script. (`elegant` writes the initial particle distribution to this file.)

2. `%o` — Will be replaced by the name of the output file from the script. (`elegant` expects the script to write the final particle distribution to this file.)

3. `%p` — Will be replaced by the pass number, which starts from 0.

4. `%c` — Will be replaced by the occurrence number of the element, which starts from 1.

5. `%b` — Will be replaced by 0 (1) for forward (backward) tracking.

6. `%np0`, `%np1`, ..., `%np9` — Will be replaced by the value of Numerical Parameter 0, 1, ..., 9. This can be used to pass to the script values that are parameters of the element definition. For example, if one wanted to vary parameters or add errors to the parameter, one would use this facility.

7. `%sp0`, `%sp1`, ..., `%sp9` — Will be replaced by the value of String Parameter 0, 1, ..., 9. This can be used to pass to the script values that are parameters of the element definition.

In some cases, one may wish to keep the input file delivered to the `SCRIPT` as well as the output file returned by it. This is facilitated by using the `ROOTNAME` parameter, which allows specifying the rootname for these files, as well as the `INPUT_EXTENSION` and `OUTPUT_EXTENSION` parameters. The `ROOTNAME` parameter may contain a simple string, but may also contain several substitutible fields:

- `%s` — The global rootname, which may be given by the `rootname` parameter in the `run_setup` command.

- `%p` — The pass index.

- `%ld` — The occurence number of the element.

Here's an example of a `SCRIPT COMMAND`:

```
myScript -input %i -output %o -accuracy %np0 -type %sp0
```

In this example, the script `myScript` takes four commandline arguments, giving the names of the input and output files, an accuracy requirement, and a type specifier. By default, `elegant` will choose unique, temporary filenames to use in communicating with the script. The actual command when executed might be something like

```
myScript -input tmp391929.1 -output tmp391929.2 -accuracy 1.5e-6 -type scraper
```

where for this example I've assumed `NP0=1.5e-6` and `SP0=''scraper''`.

If you have a program (e.g., a FORTRAN program) that does not accept commandline arguments, you can easily wrap it in a Tcl/Tk simple script to handle this. Alternatively, you can force `elegant` to use specified files for communicating with such a script. This is done using the `ROOTNAME`, `INPUT_EXTENSION`, and `OUTPUT_EXTENSION` parameters. So if your program was `crass` and it expected its input (output) in files `crass.in` (`crass.out`), then you'd use

```
S1: script,command=''crass'',rootname=''crass'',input_extension=''in'',&
output_extension=''out''
```

For purposes of computing concatenated transport matrices, Twiss parameters, response matrices, etc., `elegant` will perform initial tracking through the `SCRIPT` element using an ensemble of 25 particles. If this is not desirable, then set the parameter `DRIFT_MATRIX` to a non-zero value. This will force `elegant` to treat the element as a drift space for any calculations that involve transport matrices. Examples of where one might want to use this feature would be a `SCRIPT` that involves randomization (e.g., scattering), particle loss, or particle creation.

If non-zero, the `RPN_PARAMETERS` parameter directs `elegant` to load all numerical SDDS parameter values from the script output file into `rpn` variables, where they may be used for optimization. This provides the user the ability to perform script-based analysis of particle distributions and then optimize the results of that analysis. (Typically in this case the script does not actually transform the particle coordinates, but simply copies them from the input file to the output file.) The names of the variables are of the form *ElementName#N.ParameterName*, where N is the occurrence number of the script element (usually 1 if there is only one instance).

10.102 SEXT—A sextupole implemented as a matrix, up to 3rd order. Use KSEXT for symplectic tracking.

A sextupole implemented as a matrix, up to 3rd order. Use KSEXT for symplectic tracking.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K2	$1/M^3$	double	0.0	geometric strength
K1	$1/M^2$	double	0.0	geometric quadrupole strength error. See notes below!
J1	$1/M^2$	double	0.0	geometric skew quadrupole strength error. See notes below!
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE		double	0.0	fractional strength error
FFRINGE		double	0.0	Length occupied by linear fringe regions as fraction hard-edge length L.
ORDER		short	0	matrix order
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a sextupole using a matrix, up to third order.

The `K1` and `J1` parameters allow introducing normal and skew quadrupole **error** terms. The matrix expressions assume that these are weak effects and high accuracy should not be expected if this is not true. If `K1` is significant, then use of the `KQUSE` element is preferred.

10.103 SHRFDF—Simulation through space harmonics of zero length deflecting cavity.

Simulation through space harmonics of zero length deflecting cavity.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
FACTOR		double	1	A factor by which to multiply all components.
TILT	RAD	double	0.0	rotation about longitudinal axis
PERIOD_LENGTH	M	double	0.0	cavity period length, or cell length
PERIOD_PHASE	RAD	double	0.0	cavity period phase advance, or so-called working mode
V0	V	double	0.0	effective voltage of space harmonic n=0
V1	V	double	0.0	effective voltage of space harmonic n=1
V2	V	double	0.0	effective voltage of space harmonic n=2
V3	V	double	0.0	effective voltage of space harmonic n=3
V4	V	double	0.0	effective voltage of space harmonic n=4
V5	V	double	0.0	effective voltage of space harmonic n=5
V6	V	double	0.0	effective voltage of space harmonic n=6
V7	V	double	0.0	effective voltage of space harmonic n=7
V8	V	double	0.0	effective voltage of space harmonic n=8
V9	V	double	0.0	effective voltage of space harmonic n=9
PHASE0	HZ	double	0.0	Phase of space harmonic n=0
PHASE1	HZ	double	0.0	Phase of space harmonic n=1
PHASE2	HZ	double	0.0	Phase of space harmonic n=2
PHASE3	HZ	double	0.0	Phase of space harmonic n=3
PHASE4	HZ	double	0.0	Phase of space harmonic n=4
PHASE5	HZ	double	0.0	Phase of space harmonic n=5
PHASE6	HZ	double	0.0	Phase of space harmonic n=6
PHASE7	HZ	double	0.0	Phase of space harmonic n=7

Simulation through space harmonics of zero length deflecting cavity.

Parameter Name	Units	Type	Default	Description
PHASE8	HZ	double	0.0	Phase of space harmonic n=8
PHASE9	HZ	double	0.0	Phase of space harmonic n=9
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates an rf deflector with specified space harmonic parameters (voltage, phase). The thin kicks from the fundamental deflecting mode are the same as for the element RFDF. The thin kicks from the space harmonics ($n \geq 1$) are [55]

$$\begin{aligned} \Delta P_x &= -\frac{\partial(\mathbf{H}-H_0)}{\partial x} \\ &= \sum_{n=1}^{\infty} -q\bar{V}_n \cdot \sin(k_n z + \phi_n) \cdot (\tfrac{1}{2}\alpha_n + \tfrac{1}{16}\alpha_n^3 \cdot (3x^2 + y^2)) \end{aligned} \tag{159}$$

$$\begin{aligned} \Delta P_z &= -\frac{\partial(\mathbf{H}-H_0)}{\partial z} \\ &= \sum_{n=1}^{\infty} -q\bar{V}_n \cdot k_n \cdot \cos(k_n z + \phi_n) \cdot (\tfrac{1}{2}\alpha_n \cdot x + \tfrac{1}{16}\alpha_n^3 \cdot (x^2 + y^2) \cdot x) \end{aligned} \tag{160}$$

The wave numbers k_n and α_n are listed below.

$$k_n = \frac{\varphi_0 + 2\pi n}{d} \tag{161}$$

$$\alpha_n^2 + k_n^2 = k_0^2 \tag{162}$$

where k_n is wave number of n^{th} space harmonic, n an integer number, φ_0 the phase advance per cavity period, d the cavity period length, α_n the wave number in the radial direction, m wave number (per 2π) in the angular direction.

10.104 SLICE—Performs slice-by-slice analysis of the beam for output to a file.

Performs slice-by-slice analysis of the beam for output to a file.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
N_SLICES		long	10	number of slices
START_PID		long	-1	starting particleID for particles to dump
END_PID		long	-1	ending particleID for particles to dump
INTERVAL		long	1	interval for data output (in turns)
START_PASS		long	0	pass on which to start
END_PASS		long	-1	pass on which to end (inclusive). Ignored if negative.
FILENAME		STRING		output filename, possibly incomplete (see below)
LABEL		STRING		output label
INDEX_OFFSET		long	0	Offset for file indices for sequential file naming.
REFERENCE_FREQUENCY		double	-1	If non-zero, the indicated frequency is used to define the bucket center for purposes of computing time offsets.
DISABLE		short	0	If nonzero, no output will be generated.
USE_DISCONNECT		short	0	If nonzero, files are disconnected between each write operation. May be useful for parallel operation. Ignored otherwise.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

NB: This element has very poor parallel efficiency. Hence, the `START_PASS`, `END_PASS`, and `INTERVAL` options should be used to limit the frequency of computations to the minimum needed.

10.105 SOLE—A solenoid implemented as a matrix, up to 2nd order.

A solenoid implemented as a matrix, up to 2nd order.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
KS	RAD/M	double	0.0	geometric strength, -Bs/(B*Rho)
B	T	double	0.0	field strength (used if KS is zero)
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
ORDER		short	0	matrix order
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.106 SPEEDBUMP—Simulates a semi-circular protuberance from one or both walls of the chamber.

Simulates a semi-circular protuberance from one or both walls of the chamber.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	insertion length
CHORD	M	double	0.0	z length of speed bump
DZCENTER	M	double	0.0	z center displacement of speed bump relative to middle of object
HEIGHT	M	double	0.0	height above the surrounding chamber
POSITION	M	double	0.0	position of peak relative to ideal trajectory
DX	M	double	0.0	horizontal misalignment
DY	M	double	0.0	vertical misalignment
INSERT_FROM		STRING	NULL	direction from which inserted (x, +x, -x, y, +y, -y)
SCRAPER_CONVENTION		short	0	Historically, the interpretation of the POSITION parameter was dependent on the insertion direction. Set this to 1 to make it consistent with the SCRAPER element.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a commonplace type of aperture restriction, consisting of a bump on one or both sides of a chamber. The parameters of the speedbump are illustrated in Fig. 4 It may be useful to know that the radius R of the cylinder from which the speedbump is made is

$$R = \frac{C^2 + 4h^2}{8h}, \tag{163}$$

where C is the chord length and h is the bump height. Solving for h, we have

$$h = R - \sqrt{R^2 - \left(\frac{C}{2}\right)^2}. \tag{164}$$

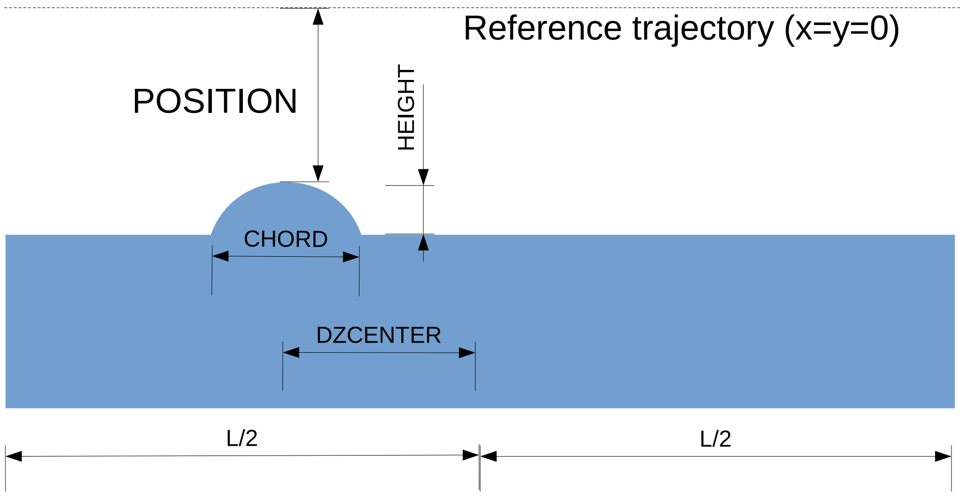

Figure 4: Illustration of the parameters used in specifying a speedbump.

10.107 SREFFECTS—Lumped simulation of synchrotron radiation effects (damping and quantum excitation) for rings.

Lumped simulation of synchrotron radiation effects (damping and quantum excitation) for rings.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
JX		double	1	x damping partition number
JY		double	1	y damping partition number
JDELTA		double	2	momentum damping partition number
EXREF	m	double	0.0	reference equilibrium x emittance
EYREF	m	double	0.0	reference equilibrium y emittance
SDELTAREF		double	0.0	reference equilibrium fractional momentum spread
DDELTAREF		double	0.0	reference fractional momentum change per turn due to SR (negative value)
PREF	$m_e c$	double	0.0	reference momentum (to which other reference values pertain)
COUPLING		double	0.0	x-y coupling
FRACTION		double	1	fraction of implied SR effect to simulate with each instance
DAMPING		long	1	include damping, less rf effects?
QEXCITATION		long	1	include quantum excitation?
LOSSES		long	1	include average losses?
CUTOFF		double	100	cutoff (in sigmas) for gaussian random numbers
INCLUDE_OFFSETS		long	1	include orbit offsets in tracking (see below)?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is intended for storage ring modeling only and provides a fast alternative to element-by-element modeling of synchrotron radiation. It should be used with care because the results will not necessarily be self-consistent. This is particularly an issue when there is dispersion at the location of the SREFFECTS element.

There are several types of storage ring simulation in which one may want to use this element:

- Simulation of instabilities or other dynamics where radiation damping or quantum excitation is important.
- Simulation of dynamics with an rf cavity when the synchronous phase is significantly different from 180 degrees, so that average radiation losses must be included.
- Computation of dynamic and momentum aperture in the presence of radiation damping.

The major parameters (JX, JY, EXREF, SDELTAREF, DDELTAREF, and PREF) can be supplied explicitly by the user, or filled in by elegant if the twiss_output command is given with radiation_integrals=1.

In explicit initialization, the user supplies the quantities EXREF, EYREF, SDELTAREF, DDELTAREF, and PREF. These are, respectively, the reference values for the x-plane emittance, y-plane emittance, fractional momentum spread, energy loss per turn, and momentum. The first four values pertain to the reference momentum. JX, JY, and JDELTA may also be given, although the defaults work for typical lattices.

In automatic initialization, the user turns on the radiation integral feature in twiss_output, causing elegant to automatically compute the above quantities. This will occur only if PREF=0. The COUPLING parameter can be used to change the partitioning of quantum excitation between the horizontal and vertical planes. Because the radiation integrals computation in twiss_output pertains to the horizontal plane only, the user must supply either EYREF or COUPLING if non-zero vertical emittance is desired.

The user may elect to turn off some aspects of the synchrotron radiation model. These should be changed from the default values with care!

- DAMPING — Default is 1. If set to 0, then no radiation damping effects will be included. More precisely, it is equivalent to setting JX=JY=JDELTA=1. Damping still occurs at any rf cavities (since elegant works in trace space).
- QEXCITATION — Default is 1. If set to 0, then no quantum excitation effects are included, which is to say that all particles will experience the same perturbation.
- LOSSES — Default is 1. If set to 0, no average energy losses are included.

There are a number of caveats that must be observed when using this element.

1. If there is dispersion at the location of the SREFFECTS element, the closed orbit will change because of the average momentum change, but it will disagree with tracking results. The reason is that in tracking SREFFECTS must displace the beam to the new equilibrium orbit, because otherwise there will be additional betatron motion excited and the wrong equilibrium emittance will be obtained. (Since the SREFFECTS element is already adding the betatron motion excitation for the entire ring, elegant is forced to offset each particle by $\Delta\delta\vec{\eta}$ to suppress any additional excitation.)

   This issue can be resolved by placing the SREFFECTS element next to the rf cavity and setting INCLUDE_OFFSETS=0. Since the average momentum change is zero from the two elements, no additional betatron motion will be generated. Optionally, one can also use many SREFFECTS elements at equivalent locations in the lattice, which will decrease the magnitude of the effect.

2. When used for dynamic aperture and momentum aperture determination, one should set `QEXCITATION=0`. Putting the rf cavity (if any) right next to the `SREFFECTS` element is a good idea to avoid spurious excitation of betatron motion.

3. Nothing prevents including this element in a lattice when doing frequency map analysis, although it probably doesn't make any sense. Only the average energy loss per turn will be included. Again, putting an rf cavity right after `SREFFECTS` is a good idea.

4. In versions 19.0 and later, `elegant` includes the effect of `SREFFECTS` on the closed orbit. This presents a dilemna when automatic initialization is used, because in order to perform automatic initialization, `elegant` has to compute the optics functions. However, it must determine the closed orbit to compute the optics functions. The solution to this is for the user to pre-compute the twiss parameters and radiation integrals using `twiss_output` with `output_at_each_step=0`. The user is free to subsequently give `twiss_output` with `output_at_each_step=1` to obtain the results on the closed orbit.

5. Computation of Twiss parameters does not fully include the effects of synchrotron radiation losses when these are imposed using `SREFFECTS` elements. If `PREF=0` (automatic initialization), these effects are completely missing. If `PREF` is non-zero, then `elegant` will use the `DDELTAREF` parameter to compute the energy offset from the element, and thus its effect on the beam trajectory.

10.108 STRAY—A stray field element with local and global components. Global components are defined relative to the initial beamline direction.

A stray field element with local and global components. Global components are defined relative to the initial beamline direction.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
LBX	T	double	0.0	local Bx
LBY	T	double	0.0	local By
GBX	T	double	0.0	global Bx
GBY	T	double	0.0	global By
GBZ	T	double	0.0	global Bz
ORDER		long	0	matrix order
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates stray fields. These fields are considered perturbations, in that they change the trajectory (or orbit), but not the floor coordinates. Local stray fields (`LBX` and `LBY`) are referenced to the local coordinate system. Global stray fields (`GBX`, `GBY`, `GBZ`) are referenced to the global coordinate system, which coincides with the local coordinate system only at the start of the beamline (unless there is no bending, in which case the two systems are identical).

10.109 TAPERAPC—A tapered aperture that is a section of a circular cylinder.

A tapered aperture that is a section of a circular cylinder.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
RSTART	M	double	0.0	radius at the start
REND	M	double	0.0	radius at the end
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
STICKY	$NULL$	short	0	final aperture holds downstream until next TAPERAPC, TAPERAPE, TAPERAPR, or MAXAMP
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.110 TAPERAPE—A tapered elliptical aperture.

A tapered elliptical aperture.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
ASTART	M	double	0.0	horizontal semi-axis at the start
AEND	M	double	0.0	horizontal semi-axis at the end
BSTART	M	double	0.0	vertical semi-axis at the start
BEND	M	double	0.0	vertical semi-axis at the end
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
TILT	RAD	double	0.0	misalignment
RESOLUTION	M	double	1e-06	z resolution of finding intersection
XEXPONENT	$NULL$	short	2	super-elliptical exponent (even number)
YEXPONENT	$NULL$	short	2	super-elliptical exponent (even number)
STICKY	$NULL$	short	0	final aperture holds downstream until next TAPERAPC, TAPERAPE, TAPERAPR, or MAXAMP
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.111 TAPERAPR—A tapered rectangular aperture.

A tapered rectangular aperture.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
XSTART	M	double	0.0	horizontal half-aperture at the start
XEND	M	double	0.0	horizontal half-aperture at the end
YSTART	M	double	0.0	vertical half-aperture at the start
YEND	M	double	0.0	vertical half-aperture at the end
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
TILT	RAD	double	0.0	misalignment
STICKY	$NULL$	short	0	final aperture holds downstream until next TAPERAPC, TAPERAPE, TAPERAPR, or MAXAMP
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.112 TFBDRIVER—Driver for a turn-by-turn feedback loop

Driver for a turn-by-turn feedback loop
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
ID		STRING	NULL	System identifier
STRENGTH		double	0.0	Strength factor
KICK_LIMIT		double	0.0	Limit on applied kick; in radians for transverse plane or fraction of momentum in longitudinal plane.
FREQUENCY	Hz	double	0.0	Resonant frequency of the unloaded kicker cavity.
DRIVE_FREQUENCY	Hz	double	0.0	Drive frequency. If zero, defaults to resonant frequency of the loaded cavity.
CLOCK_FREQUENCY	Hz	double	0.0	Clock frequency used for timing of the changes to generator current. Typically the rf or bunch frequency is used.
CLOCK_OFFSET	s	double	0.0	Offset of the generator current change relative to clock tick. Clock tick is nominally aligned to the bunch center.
PHASE	Deg	double	0.0	Phase of the applied voltage relative to the bunch center, with 0 being on-crest.
RAOVERQ	Ohm	double	0.0	Shunt impedance, Ra/Q=V$\hat{2}$/(P*Q).
QLOADED		double	0.0	Loaded Q of the cavity.
OUTPUT_FILE		STRING	NULL	File for logging filter output and driver output
GAIN_FACTOR_FILE		STRING	NULL	File providing gain factors for individual bunches.
GAIN_FACTOR_COLUMN		STRING	NULL	Column from GAIN_FACTOR_FILE containing gain factors.
DELAY		long	0	Delay (in turns)
A0		double	1	Filter coefficient
A1		double	0.0	Filter coefficient

Driver for a turn-by-turn feedback loop

Parameter Name	Units	Type	Default	Description
A2		double	0.0	Filter coefficient
A3		double	0.0	Filter coefficient
A4		double	0.0	Filter coefficient
A5		double	0.0	Filter coefficient
A6		double	0.0	Filter coefficient
A7		double	0.0	Filter coefficient
A8		double	0.0	Filter coefficient
A9		double	0.0	Filter coefficient
A10		double	0.0	Filter coefficient
A11		double	0.0	Filter coefficient
A12		double	0.0	Filter coefficient
A13		double	0.0	Filter coefficient
A14		double	0.0	Filter coefficient
A15		double	0.0	Filter coefficient
A16		double	0.0	Filter coefficient
A17		double	0.0	Filter coefficient
A18		double	0.0	Filter coefficient
A19		double	0.0	Filter coefficient
A20		double	0.0	Filter coefficient
A21		double	0.0	Filter coefficient
A22		double	0.0	Filter coefficient
A23		double	0.0	Filter coefficient
A24		double	0.0	Filter coefficient
A25		double	0.0	Filter coefficient
A26		double	0.0	Filter coefficient
A27		double	0.0	Filter coefficient
A28		double	0.0	Filter coefficient
A29		double	0.0	Filter coefficient
UPDATE_INTERVAL		long	0	Interval in units of pickup update interval for sampling pickup data and updating filter output.
OUTPUT_INTERVAL		long	1024	Number of samples to buffer between writing output file updates.
START_PASS		long	-1	If positive, first pass on which to drive beam.

Driver for a turn-by-turn feedback loop

Parameter Name	Units	Type	Default	Description
END_PASS		long	-1	If positive, last pass on which to drive beam.
LONGITUDINAL		short	0	If non-zero, kick is in the longituidinal plane. KICK_LIMIT is in fractional momentum deviation.
BUNCHED_BEAM_MODE		short	1	If non-zero, run in bunched beam mode.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is used together with the TFBPICKUP element to simulate a digital turn-by-turn feedback system. Each TFBDRIVER element must have a unique identification string assigned to it using the ID parameter. The same identifier must be used on a TFBPICKUP element. This is the pickup from which the driver gets its signal. Each pickup may feed more than one driver, but a driver can use only one pickup.

A 30-term FIR filter can be defined using the A0 through A29 parameters. The output of the filter is simply $\sum_{i=0}^{29} a_i P_i$, where P_i is the pickup filter output from $i*U$ turns ago, where U is the UPDATE_INTERVAL value specified for the pickup. The output of the filter is optionally delayed by the number of update intervals given by the DELAY parameter.

To some extent, the DELAY is redundant. For example, the filter $a_0 = 0, a_1 = 1$ with a delay of 0 is equivalent to $a_0 = 1, a_1 = 0$ with a delay of 1. However, for long delays or delays combined with many-term filters, the DELAY feature must be used.

The output of the filter is multiplied by the STRENGTH parameter to get the kick to apply to the beam. The KICK_LIMIT parameter provides a very basic way to simulate saturation of the kicker output.

The plane that the TFBDRIVER kicks is determined by the PLANE parameter on the corresponding TFBPICKUP element, and additionally by the LONGITUDINAL parameter, as described in Table 3

Note: The OUTPUT_FILE will produce a file with missing data at the end of the buffer if the OUTPUT_INTERVAL parameter is not a divisor of the number of passes.

The FREQUENCY and PHASE parameters may be used to specify the resonant frequency of the driving cavity and its phase relative to the center of the bunch. If the frequency is not specified, the kicker is assumed to kick all particles in a bunch by the same amount.

For longitudinal feedback only, a more sophicated approach is available using a circuit model developed by T. Berenc (APS) may be employed to simulate driving the cavity resonance. To invoke this, the user must provide the loaded Q of the cavity using the QLOADED parameter, the (R_a/Q) using RAOVERQ, and the resonant frequency of the unloaded cavity using FREQUENCY. Optionally, the drive frequency may be specified using DRIVE_FREQUENCY; it defaults to the unloaded resonant

TFBPICKUP PLANE	TFBDRIVER LONGITUDINAL	coordinate kicked	note
x	0	x'	
x	1	δ	pickup should have $\eta_x \neq 0$
y	0	$y\prime$	
y	1	δ	pickup should have $\eta_y \neq 0$
delta	0	-	invalid
delta	1	δ	

Table 3: Correspondence between PLANE parameter of TFBPICKUP, LONGITUDINAL parameter of TFBDRIVER, and action of feedback loop.

frequency.

Typically one should choose the resonant frequency to be $(n \pm \frac{1}{4})f_b$, where f_b is the bunch frequency and n is an integer. This will ensure that the kick to one bunch from the residual voltage from the previous bunch (both beam-loading and generator terms), is approximately minimized. Checking the ResidualVoltage column in the output file to confirm this is advised.

In addition to the resonant and drive frequencies, one must specify a clock frequency with CLOCK_FREQUENCY and a clock offset with CLOCK_OFFSET. The clock used used to determine when the drive current changes, which happens at regular intervals. The clock offset is used to ensure that the change does not occur during passage of the bunch. If the clock offset is too small and the bunch length too long, this will happen and results in an error. The phase shift that results from the clock offset is automatically compensated.

Beam loading is not included in the model, but can be superimposed by inserting an RFMODE element with matching parameters.

See Section 7.2.14 of *Handbook of Accelerator Physics and Engineering* (Chao and Tigner, eds.) for a discussion of feedback systems.

10.113 TFBPICKUP—Pickup for a turn-by-turn feedback loop

Pickup for a turn-by-turn feedback loop
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
ID		STRING	NULL	System identifier
PLANE		STRING	x	"x", "y", "delta", or "phase"
RMS_NOISE	M	double	0.0	RMS noise to add to position readings.
A0		double	0.0	Filter coefficient
A1		double	0.0	Filter coefficient
A2		double	0.0	Filter coefficient
A3		double	0.0	Filter coefficient
A4		double	0.0	Filter coefficient
A5		double	0.0	Filter coefficient
A6		double	0.0	Filter coefficient
A7		double	0.0	Filter coefficient
A8		double	0.0	Filter coefficient
A9		double	0.0	Filter coefficient
A10		double	0.0	Filter coefficient
A11		double	0.0	Filter coefficient
A12		double	0.0	Filter coefficient
A13		double	0.0	Filter coefficient
A14		double	0.0	Filter coefficient
A15		double	0.0	Filter coefficient
A16		double	0.0	Filter coefficient
A17		double	0.0	Filter coefficient
A18		double	0.0	Filter coefficient
A19		double	0.0	Filter coefficient
A20		double	0.0	Filter coefficient
A21		double	0.0	Filter coefficient
A22		double	0.0	Filter coefficient
A23		double	0.0	Filter coefficient
A24		double	0.0	Filter coefficient
A25		double	0.0	Filter coefficient
A26		double	0.0	Filter coefficient
A27		double	0.0	Filter coefficient
A28		double	0.0	Filter coefficient
A29		double	0.0	Filter coefficient
UPDATE_INTERVAL		long	0	Interval in turns for sampling data and updating filter output.

Pickup for a turn-by-turn feedback loop

Parameter Name	Units	Type	Default	Description
START_PASS		long	-1	If positive, first pass on which to perform computations.
END_PASS		long	-1	If positive, last pass on which to perform computations.
REFERENCE_FREQUENCY		double	0.0	Reference frequency for computing phase offsets.
DX	M	double	0.0	Horizontal offset (subtracted from pickup signal).
DY	M	double	0.0	Vertical offset (subtracted from pickup signal)
BUNCHED_BEAM_MODE		short	1	If non-zero, run in bunched beam mode.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is used together with the `TFBDRIVER` element to simulate a digital turn-by-turn feedback system. Each `TFBPICKUP` element must have a unique identification string assigned to it using the `ID` parameter. This is used to identify which drivers get signals from the pickup.

A 30-term FIR filter can be defined using the `A0` through `A29` parameters. The input to the filter is the turn-by-turn beam centroid at the pickup location. The output of the filter is simply $\sum_{i=0}^{29} a_i C_i$, where C_i is the centroid from $i * U$ turns ago, where U is the value specified by the `UPDATE_INTERVAL` parameter. Note that $\sum_{i=0}^{29} a_i$ should generally be zero. Otherwise, the system will attempt to correct the DC orbit. The output of the filter is the input to the driver element(s).

The `PLANE` parameter can take four values: "x", "y", "delta", and "phase", specifying what centroid property of the beam is measured by the pickup. The "delta"-mode pickup is nonphysical, but could have applications to simulations where is not convenient to put a pickup in a high-dispersion area.

See Section 7.2.14 of *Handbook of Accelerator Physics and Engineering* (Chao and Tigner, eds.) for a discussion of feedback systems.

10.114 TMCF—A numerically-integrated accelerating TM RF cavity with spatially-constant fields.

A numerically-integrated accelerating TM RF cavity with spatially-constant fields.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
FREQUENCY	HZ	double	2856000000	frequency
PHASE	S	double	0.0	phase
TIME_OFFSET	S	double	0.0	time offset (adds to phase)
RADIAL_OFFSET	M	double	1	not recommended
TILT	RAD	double	0.0	rotation about longitudinal axis
ER	V	double	0.0	radial electric field
BPHI	T	double	0.0	azimuthal magnetic field
EZ	V	double	0.0	longitudinal electric field
ACCURACY		double	0.0001	integration accuracy
X_MAX	M	double	0.0	x half-aperture
Y_MAX	M	double	0.0	y half-aperture
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
N_STEPS		long	100	number of steps (for nonadaptive integration)
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, non-adaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.115 TRCOUNT—An element that defines the point from which transmission calculations are made.

An element that defines the point from which transmission calculations are made.
Parallel capable? : no
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
DUMMY		long	0	
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.116 TRFMODE—A simulation of a beam-driven TM dipole mode of an RF cavity.

A simulation of a beam-driven TM dipole mode of an RF cavity.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
RA	Ohm/m	double	0.0	shunt impedance, Ra=V2̂/P
RS	Ohm/m	double	0.0	shunt impedance (Rs=Ra/2)
Q		double	0.0	cavity Q
FREQ	Hz	double	0.0	frequency
CHARGE	C	double	0.0	Deprecated—use CHARGE element
BETA		double	0.0	normalized load impedance
BIN_SIZE	S	double	0.0	bin size for current histogram
N_BINS		long	20	number of bins for current histogram
INTERPOLATE		long	0	if non-zero, interpolate voltage within bins
PLANE		STRING	both	x, y, or both
SAMPLE_INTERVAL		long	1	passes between output to RECORD file
PER_PARTICLE_OUTPUT		long	0	If non-zero, then in BINLESS mode, provides per-particle output of RECORD data.
RECORD		STRING	NULL	output file for cavity data
SINGLE_PASS		long	0	if nonzero, don't accumulate field from pass to pass
RIGID_UNTIL_PASS		long	0	don't affect the beam until this pass
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
XFACTOR		double	1	factor by which to multiply shunt impedances
YFACTOR		double	1	factor by which to multiply shunt impedances
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the impedance to full strength.
BINLESS		long	0	If nonzero, use algorithm that doesn't requiring binning. Best for few particles, widely spaced.

A simulation of a beam-driven TM dipole mode of an RF cavity.

Parameter Name	Units	Type	Default	Description
RESET_FOR_EACH_STEP		long	1	If nonzero, voltage and phase are reset for each simulation step.
LONG_RANGE_ONLY		long	0	If nonzero, induced voltage from present turn does not affect bunch. Short range wake should be included via TRWAKE or ZTRANSVERSE element.
N_CAVITIES		long	1	effect is multiplied by this number, simulating N identical cavities
BUNCHED_BEAM_MODE		long	1	If non-zero, then do calculations bunch-by-bunch.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a beam-driven dipole mode cavity using the fundamental theorem of beam loading and phasor rotation.

Normally, the field dumped in the cavity by one particle affects trailing particles in the same turn. However, if one is also using a `TRWAKE` or `ZTRANSVSE` element to simulate the short-range wake of the cavity, this would be double-counting. In that case, one can use `LONG_RANGE_ONLY=1` to suppress the same-turn effects of the `RFMODE` element.

10.117 TRWAKE—Transverse wake specified as a function of time lag behind the particle.

Transverse wake specified as a function of time lag behind the particle.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
INPUTFILE		STRING	NULL	name of file giving Green functions
TCOLUMN		STRING	NULL	column in INPUTFILE containing time data
WXCOLUMN		STRING	NULL	column in INPUTFILE containing x Green function
WYCOLUMN		STRING	NULL	column in INPUTFILE containing y Green function
CHARGE	C	double	0.0	Deprecated—use CHARGE element
FACTOR		double	1	factor by which to multiply both wakes
XFACTOR		double	1	factor by which to multiply x wake
YFACTOR		double	1	factor by which to multiply y wake
N_BINS		long	0	number of bins for current histogram
INTERPOLATE		long	0	interpolate wake?
SMOOTHING		long	0	Use Savitzky-Golay filter to smooth current histogram?
SG_HALFWIDTH		long	4	Savitzky-Golay filter halfwidth for smoothing
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
TILT	RAD	double	0.0	rotation about longitudinal axis
X_DRIVE_EXPONENT		long	1	Exponent applied to x coordinates of drive particles
Y_DRIVE_EXPONENT		long	1	Exponent applied to y coordinates of drive particles
X_PROBE_EXPONENT		long	0	Exponent applied to x coordinates of probe particles
Y_PROBE_EXPONENT		long	0	Exponent applied to y coordinates of probe particles

Transverse wake specified as a function of time lag behind the particle.

Parameter Name	Units	Type	Default	Description
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the wake to full strength.
BUNCHED_BEAM_MODE		long	1	If non-zero, then do calculations bunch-by-bunch.
START_BUNCH		long	-1	In bunched beam mode, if non-negative, starting bunch number for computations
END_BUNCH		long	-1	In bunched beam mode, if non-negative, ending bunch number for computations
ACAUSAL_ALLOWED		long	0	If non-zero, then an acausal wake is allowed.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The input file for this element gives the transverse-wake Green functions, $W_x(t)$ and $W_y(t)$, versus time behind the particle. The units of the wakes are V/C/m, so this element simulates the integrated wake of some structure (e.g., a cell or series of cells). If you have, for example, the wake for a cell and you need the wake for N cells, then you may use the `FACTOR` parameter to make the appropriate multiplication. The values of the time coordinate should begin at 0 and be equi-spaced, and be expressed in seconds. A positive value of time represents the distance behind the exciting particle. Time values must be equally spaced.

The sign convention for W_q (q being x or y) is as follows: a particle with $q > 0$ will impart a positive kick ($\Delta q' > 0$) to a trailing particle following t seconds behind if $W_q(t) > 0$. A physical wake function should be zero at $t = 0$ and also be initially positive as t increases from 0. Causality requires that $W_q(t) = 0$ for $t < 0$. Acasual wakes are supported, provided the user sets `ACAUSAL_ALLOWED=0`. The data file must contain a value of $W(t)$ at $t = 0$, and should have equal spans of time to the negative and positive side of $t = 0$.

Use of the `CHARGE` parameter on the `TRWAKE` element is disparaged. It is preferred to use the `CHARGE` element as part of your beamline to define the charge.

Setting the `N_BINS` paramater to 0 is recommended. This results in auto-scaling of the number of bins to accomodate the beam. The bin size is fixed by the spacing of the time points in the wake.

The default degree of smoothing (`SG_HALFWIDTH=4`) may be excessive. It is suggested that users vary this parameter to verify that results are reliable if smoothing is employed (`SMOOTHING=1`).

The `XFACTOR` and `YFACTOR` parameters can be used to adjust the strength of the wakes if the location at which you place the `TRWAKE` element has different beta functions than the location at which the object that causes the wake actually resides.

The X_DRIVE_EXPONENT and Y_DRIVE_EXPONENT parameters can be used to change the dependence of the wake on the x and y coordinates, respectively, of the particles. Normally, these have the value 1, which corresponds to an ordinary dipole wake in a symmetric chamber.

If you have an asymmetric chamber, then you will have a transverse wake kick even if the beam is centered. (Of course, you'll need a 3-D wake code like GdfidL or MAFIA to compute this wake.) This part of the transverse wake is modeled by setting X_DRIVE_EXPONENT=0 and Y_DRIVE_EXPONENT=0. It will result in an orbit distortion, but conceivably could have other effects, such as emittance dilution. In this case, the units for the x and y wake must be V/C. A negative value of the wake corresponds to a kick toward negative x (or y).

In addition, a quadrupole wake can be modeled by setting X_DRIVE_EXPONENT=0, Y_DRIVE_EXPONENT=0, X_PROBE_EXPONENT=1, and Y_PROBE_EXPONENT=1. The kick to a particle now depends on *it's* displacement, not on the displacement of the leading particles. In this case, the units for the wakes must be $V/C/m$.

Bunched-mode application of the short-range wake is possible using specially-prepared input beams. See Section 6 for details. The use of bunched mode for any particular TRWAKE element is controlled using the BUNCHED_BEAM_MODE parameter

10.118 TSCATTER—An element to simulate Touschek scattering.

An element to simulate Touschek scattering.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
DUMMY		long	0	
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.119 TUBEND—A special rectangular bend element for top-up backtracking.

A special rectangular bend element for top-up backtracking.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	arc length
ANGLE	RAD	double	0.0	bend angle
FSE		double	0.0	fractional strength error
OFFSET		double	0.0	horizontal offset of magnet center from arc center
MAGNET_WIDTH		double	0.0	horizontal width of the magnet pole
MAGNET_ANGLE		double	0.0	angle that the magnet was designed for
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.120 TWISS—Sets Twiss parameter values.

Sets Twiss parameter values.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
BETAX	M	double	1	horizontal beta function
ALPHAX		double	0.0	horizontal alpha function
ETAX	M	double	0.0	horizontal eta function
ETAXP		double	0.0	slope of horizontal eta function
BETAY	M	double	1	vertical beta function
ALPHAY		double	0.0	vertical alpha function
ETAY	M	double	0.0	vertical eta function
ETAYP		double	0.0	slope of vertical eta function
FROM_BEAM		short	0	compute transformation from tracked beam properties instead of Twiss parameters?
FROM_0VALUES		short	0	if non-zero, transformation is from the "0" values provided in the element definition
COMPUTE_ONCE		short	0	compute transformation only for first beam or lattice functions?
APPLY_ONCE		short	1	apply correction only on first pass through for each beam?
VERBOSE		short	0	if non-zero, print extra information about transformations
DISABLE		short	0	if non-zero, element is ignored
BETAX0	M	double	1	initial horizontal beta function (if FROM_0VALUES nonzero)
ALPHAX0		double	0.0	initial horizontal alpha function (if FROM_0VALUES nonzero)
ETAX0	M	double	0.0	initial horizontal eta function (if FROM_0VALUES nonzero)

Sets Twiss parameter values.

Parameter Name	Units	Type	Default	Description
ETAXP0		double	0.0	initial slope of horizontal eta function (if FROM_0VALUES nonzero)
BETAY0	M	double	1	initial vertical beta function (if FROM_0VALUES nonzero)
ALPHAY0		double	0.0	initial vertical alpha function (if FROM_0VALUES nonzero)
ETAY0	M	double	0.0	initial vertical eta function (if FROM_0VALUES nonzero)
ETAYP0		double	0.0	initial slope of vertical eta function (if FROM_0VALUES nonzero)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This elements allows transformation of the twiss parameters of a beam with a first-order matrix. The matrix is computed in various ways based on initial and final twiss parameters. Depending on how you set it up, the final twiss parameters for your beam may not be the twiss parameters you specify.

The twiss parameter values `BETAX`, `BETAY`, etc. specified in the element definition specify the target values of the transformation. To completely specify the transformation, one must know the initial values as well.

Lattice-Function-Based Transformation

If `FROM_BEAM` is zero, which is the default, then the initial values are taken from the incoming lattice functions computed by `twiss_output`. This provides a way to transform the lattice functions between two parts of a transport line without designing intervening optics. A beam that is matched at the beginning of the transport line will remain matched. A beam that is mismatched at the beginning of the transport line *will not* be matched after the `TWISS` element.

By default, each time the twiss parameters are recomputed, the transformation is updated to maintain the desired lattice functions at the exit of the `TWISS` element. Setting `COMPUTE_ONCE` to a non-zero value specifies that `elegant` should compute the transformation matrix only once, i.e., for the first set of computed lattice functions.

By default, the transformation is applied to the beam only the first time it passes the element. Setting `APPLY_ONCE` to a zero will result in application of the transformation at each pass.

Beam-Ellipse-Based Transformation

If `FROM_BEAM` is non-zero, the the initial values for the transformation are computed from a beam. This provides a way to transform the beam ellipse to the desired twiss parameters irrespective of the lattice. The results from `twiss_output` will not necessarily be matched downstream of this element. Only if the beam ellipse and lattice ellipse are the same will this occur.

By default, each time a new beam is generated, the transformation will be updated to maintain the desired beam ellipse at the exit of the TWISS element. Setting COMPUTE_ONCE to a non-zero value specifies that **elegant** should compute the transformation matrix only once, i.e., for the first beam it sees.

By default, the transformation is applied to the beam only the first time it passes the element. Setting APPLY_ONCE to a zero will result in application of the transformation at each pass. This would make sense, for example, if the TWISS element was filling in for a section of a ring. It wouldn't make sense if the TWISS element was being used to match the beam from a transport line to a ring.

10.121 TWLA—A numerically-integrated first-space-harmonic traveling-wave linear accelerator.

A numerically-integrated first-space-harmonic traveling-wave linear accelerator.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
FREQUENCY	HZ	double	2856000000	frequency
PHASE	RAD	double	0.0	phase
TIME_OFFSET	S	double	0.0	time offset (adds to phase)
EZ	V/M	double	0.0	electric field
B_SOLENOID	T	double	0.0	solenoid field
ACCURACY		double	0.0001	integration accuracy
X_MAX	M	double	0.0	x half-aperture
Y_MAX	M	double	0.0	y half-aperture
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
BETA_WAVE		double	1	(phase velocity)/c
ALPHA	$1/M$	double	0.0	field attenuation factor
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
N_STEPS		long	100	number of steps (for nonadaptive integration)
FOCUSSING		long	1	include focusing effects?
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, non-adaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")
CHANGE_P0		long	0	does element change central momentum?
SUM_BN2		double	0.0	sum of squares of amplitudes of n!=0 space harmonics
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.122 TWMTA—A numerically-integrated traveling-wave muffin-tin accelerator.

A numerically-integrated traveling-wave muffin-tin accelerator.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
FREQUENCY	HZ	double	2856000000	frequency
PHASE	RAD	double	0.0	phase
EZ	V/M	double	0.0	electric field
ACCURACY		double	0.0001	integration accuracy
X_MAX	M	double	0.0	x half-aperture
Y_MAX	M	double	0.0	y half-aperture
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
KX	$1/M$	double	0.0	horizontal wave number
BETA_WAVE		double	1	(phase velocity)/c
BSOL		double	0.0	solenoid field
ALPHA	$1/M$	double	0.0	field attenuation factor
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
N_STEPS		long	100	number of kicks
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, non-adaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.123 TWPL—A numerically-integrated traveling-wave stripline deflector.

A numerically-integrated traveling-wave stripline deflector.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
RAMP_TIME	S	double	1e-09	time to ramp to full strenth
TIME_OFFSET	S	double	0.0	offset of ramp-start time
VOLTAGE	V	double	0.0	maximum voltage between plates due to ramp
GAP	M	double	0.01	gap between plates
STATIC_VOLTAGE	V	double	0.0	static component of voltage
TILT	RAD	double	0.0	rotation about longitudinal axis
ACCURACY		double	0.0001	integration accuracy
X_MAX	M	double	0.0	x half-aperture
Y_MAX	M	double	0.0	y half-aperture
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
N_STEPS		long	100	number of steps (for nonadaptive integration)
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, non-adaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.124 UKICKMAP—An undulator kick map (e.g., using data from RADIA).

An undulator kick map (e.g., using data from RADIA).
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FIELD_FACTOR		double	1	Factor by which to multiply the magnetic fields.
XY_FACTOR		double	1	Factor by which to multiply the x and y values in the input file.
YAW		double	0.0	Yaw angle of the device. Meaningful only if N_KICKS is not 1.
INPUT_FILE		STRING	NULL	Name of SDDS file with undulator kickmap data.
N_KICKS		long	1	Number of kicks into which to split the element.
PERIODS		long	0	Number of periods (for radiation integral computations only).
KREF		double	0.0	Reference value of undulator parameter. K=KREF*FIELD_FACTOR is used for radiation integral calculations only assuming period=L/PERIODS.
KACTUAL		double	0.0	Value of undulator parameter, used for radiation integral calculations only assuming period=L/PERIODS.
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?

An undulator kick map (e.g., using data from RADIA).

Parameter Name	Units	Type	Default	Description
YAW_END		short	0	-1=Entrance, 0=Center, 1=Exit
SINGLE_PERIOD_MAP		short	0	if non-zero, the map file is for a single period. L still pertains to the full device. Set N_KICKS to the number of periods.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element provides simulation of undulators using kick maps [27]. For general-purpose kickmaps, use the KICKMAP element.

A script (km2sdds) is provided with the elegant distribution to translate RADIA [28] output into SDDS for use by elegant.

The input file has the following columns:

- x — Horizontal position in meters.
- y — Vertical position in meters.
- xpFactor — Horizontal kick factor C_x in T^2m^2. This factor is defined by equation (5a) in [27]. In particular, $\Delta x' = C_x/H^2$, where H is the beam rigidity in T^2m^2.
- ypFactor — Vertical kick factor C_y in T^2m^2. This factor is defined by equation (5b) in [27]. In particular, $\Delta y' = C_y/H^2$, where H is the beam rigidity in T^2m^2.

The values of x and y must be laid out on a grid of equispaced points. It is assumed that the data is ordered such that x varies fastest. This can be accomplished with the command

```
% sddssort -column=y,increasing -column=x,increasing input1.sdds input2.sdds
```

where input1.sdds is the original (unordered) file and input2.sdds is the new file, which would be used with UKICKMAP.

The data file is assumed to result from integration through a full device. If instead it results from integration through just a single period of a full device, one should set the SINGLE_PERIOD_MAP parameter to 1 and N_KICKS equal to the number of periods. (One can also use the FIELD_FACTOR parameter to get the same result, but this is confusing and is discouraged.)

elegant performs radiation integral computations for UKICKMAP and can also include radiation effects in tracking. This feature has limitations, namely, that the radiation integral computations assume the device is horizontally deflecting. However, in tracking, no such assumption is made. To obtain synchrotron radiation integral effects (e.g., in output from twiss_output), the KREF and

PERIODS parameters must be given. Care must be taken when using the FIELD_FACTOR parameter in this case, particularly if it is adjusted to account for using a single-period kickmap multiple times. To obtain synchrotron radiation effects in tracking, the SYNCH_RAD and/or ISR flags must additionally be used.

N.B.: at present this element is *not* included in beam moments computations via the moments_output command (the CWIGGLER element is an option for that).

The YAW and YAW_END parameters can be used in the simulation of canted IDs. Normally, steering magnets are used to create an angle between the devices. The devices are thus oriented in the reference coordinate system, meaning the beam tranverses the IDs at an angle. If it is desirable to align the IDs to the beam, the IDs can be yawed. A positive yaw will tilt the ID so that it is colinear with a beam that has been kicked by a positive horizontal steering angle. The YAW_END parameter defines which end of the ID is held fixed when the yaw is applied.

This element was requested by W. Guo (BNL), who also assisted with the implementation and debugging.

10.125 VKICK—A vertical steering dipole implemented as a matrix, up to 2nd order. Use EVKICK for symplectic tracking.

A vertical steering dipole implemented as a matrix, up to 2nd order. Use EVKICK for symplectic tracking.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
KICK	RAD	double	0.0	kick strength
TILT	RAD	double	0.0	rotation about longitudinal axis
B2	$1/M^2$	double	0.0	normalized sextupole strength (kick = KICK*(1+B2*y$\hat{2}$))
CALIBRATION		double	1	strength multiplier
EDGE_EFFECTS		short	0	include edge effects?
ORDER		short	0	matrix order
STEERING		short	1	use for steering?
SYNCH_RAD		short	0	include classical, single-particle synchrotron radiation?
ISR		short	0	include incoherent synchrotron radiation (quantum excitation)?
LERAD		double	0.0	if L=0, use this length for radiation computations
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

10.126 VMON—A vertical position monitor, accepting a rpn equation for the readout as a function of the actual position (y).

A vertical position monitor, accepting a rpn equation for the readout as a function of the actual position (y).
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
WEIGHT		double	1	weight in correction
TILT		double	0.0	rotation about longitudinal axis
CALIBRATION		double	1	calibration factor for readout
SETPOINT	M	double	0.0	steering setpoint
ORDER		short	0	matrix order
READOUT		STRING	NULL	rpn expression for readout (actual position supplied in variable y)
CO_FITPOINT		short	0	If nonzero, then closed orbit value is placed in variable <name>#<occurence>.yco
STORE_TURN_BY_TURN		short	0	If nonzero, then turn-by-turn vertical position readout and number of particles are placed in variables <name>#<occurence>.y/n.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

If `STORE_TURN_BY_TURN` is nonzero, then the computed vertical BPM reading is stored in *ElementName#ElementOccurence*.y during tracking. This can be used in the `expression` in `modulate`elements— to create position-triggered changes to elements.

10.127 WAKE—Longitudinal wake specified as a function of time lag behind the particle.

Longitudinal wake specified as a function of time lag behind the particle.
Parallel capable? : yes
GPU capable? : yes
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
INPUTFILE		STRING	NULL	name of file giving Green function
TCOLUMN		STRING	NULL	column in INPUTFILE containing time data
WCOLUMN		STRING	NULL	column in INPUTFILE containing Green function
CHARGE	C	double	0.0	Deprecated—use CHARGE element
FACTOR		double	1	factor by which to multiply wake
N_BINS		long	0	number of bins for current histogram
INTERPOLATE		long	0	interpolate wake?
SMOOTHING		long	0	Use Savitzky-Golay filter to smooth current histogram?
SG_HALFWIDTH		long	4	Savitzky-Golay filter half-width for smoothing
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing
CHANGE_P0		long	0	change central momentum?
ALLOW_LONG_BEAM		long	0	allow beam longer than wake data?
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the wake to full strength.
BUNCHED_BEAM_MODE		long	1	If non-zero, then do calculations bunch-by-bunch.
START_BUNCH		long	-1	In bunched beam mode, if non-negative, starting bunch number for computations
END_BUNCH		long	-1	In bunched beam mode, if non-negative, ending bunch number for computations
ACAUSAL_ALLOWED		long	0	If non-zero, then an acausal wake is allowed.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The input file for this element gives the longitudinal Green function, $W(t)$ versus time behind the particle. The units of the wake are V/C, so this element simulates the integrated wake of some structure (e.g., a cell or series of cells). If you have, for example, the wake for a cell and you need the wake for N cells, then you may use the `FACTOR` parameter to make the appropriate multiplication. The values of the time coordinate should begin at 0 and be equi-spaced, and be expressed in seconds. A positive value of time represents the distance behind the exciting particle.

A positive value of $W(t)$ results in energy *loss*. A physical wake function should be positive at $t = 0$. Causality requires that $W(t) = 0$ for $t < 0$. Acasual wakes are supported, provided the user sets `ACAUSAL_ALLOWED=0`. The data file must contain a value of $W(t)$ at $t = 0$, and should have equal spans of time to the negative and positive side of $t = 0$.

Use of the `CHARGE` parameter on the `WAKE` element is disparaged. It is preferred to use the `CHARGE` element as part of your beamline to define the charge.

Setting the `N_BINS` paramater to 0 is recommended. This results in auto-scaling of the number of bins to accomodate the beam. The bin size is fixed by the spacing of the time points in the wake.

The default degree of smoothing (`SG_HALFWIDTH=4`) may be excessive. It is suggested that users vary this parameter to verify that results are reliable if smoothing is employed (`SMOOTHING=1`).

The algorithm for the wake element is as follows:

1. Compute the arrival time of each particle at the wake element. This is necessary because `elegant` uses the longitudinal coordinate $s = \beta ct$.

2. Find the mean, minimum, and maximum arrival times (t_{mean}, t_{min}, and t_{max}, respectively). If $t_{max} - t_{min}$ is greater than the duration of the wakefield data, then `elegant` either exits (default) or issues a warning (if `ALLOW_LONG_BEAM` is nonzero). In the latter case, that part of the beam that is furthest from t_{mean} is ignored for computation of the wake.

3. If the user has specified a fixed number of bins (not recommended), then `elegant` centers those bins on t_{mean}. Otherwise, the binning range encompasses $t_{min} - \Delta t$ to $t_{max} + \Delta t$, where Δt is the spacing of data in the wake file.

4. Create the arrival time histogram. If any particles are outside the histogram range, issue a warning.

5. If `SMOOTHING` is nonzero, smooth the arrival time histogram.

6. Convolve the arrival time histogram with the wake function.

7. Multiply the resultant wake by the charge and any user-defined factor.

8. Apply the energy changes for each particle. This is done in such a way that the transverse momentum are conserved.

9. If `CHANGE_P0` is nonzero, change the reference momentum of the beamline to match the average momentum of the beam.

Bunched-mode application of the short-range wake is possible using specially-prepared input beams. See Section 6 for details. The use of bunched mode for any particular `WAKE` element is controlled using the `BUNCHED_BEAM_MODE` parameter.

10.128 WATCH—A beam property/motion monitor–allowed modes are centroid, parameter, coordinate, and fft.

A beam property/motion monitor–allowed modes are centroid, parameter, coordinate, and fft.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : yes

Parameter Name	Units	Type	Default	Description
FRACTION		double	1	fraction of particles to dump (coordinate mode)
START_PID		long	-1	starting particleID for particles to dump
END_PID		long	-1	ending particleID for particles to dump
INTERVAL		long	1	interval for data output (in turns)
START_PASS		long	0	pass on which to start
END_PASS		long	-1	pass on which to end (inclusive). Ignored if negative.
FILENAME		STRING		output filename, possibly incomplete (see below)
LABEL		STRING		output label
MODE		STRING	coordinates	coordinate, parameter, centroid, or fft. For fft mode, you may add a space and a qualifer giving the window type: hanning (default), parzen, welch, or uniform.
X_DATA		short	1	include x data in coordinate mode?
Y_DATA		short	1	include y data in coordinate mode?
LONGIT_DATA		short	1	include longitudinal data in coordinate mode?
EXCLUDE_SLOPES		short	0	exclude slopes in coordinate mode?
FLUSH_INTERVAL		long	100	file flushing interval (parameter or centroid mode)
SPARSE_INTERVAL		long	1	interval for particle output (coordinate mode)
DISABLE		short	0	If nonzero, no output will be generated.
USE_DISCONNECT		short	0	If nonzero, files are disconnected between each write operation. May be useful for parallel operation. Ignored otherwise.

A beam property/motion monitor–allowed modes are centroid, parameter, coordinate, and fft.

Parameter Name	Units	Type	Default	Description
INDEX_OFFSET		long	0	Offset for file indices for sequential file naming.
REFERENCE_FREQUENCY		double	-1	If non-zero, the indicated frequency is used to define the bucket center for purposes of computing time offsets.
AUTO_REFERENCE		short	0	If nonzero, uses the highest-frequency RFCA or RFCW element to determien the reference frequency.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The output filename may be an incomplete filename. In the case of the `WATCH` point element, this means it may contain one instance of the string format specification "%s" and one occurence of an integer format specification (e.g., "%ld"). `elegant` will replace the format with the rootname (see `run_setup`) and the latter with the element's occurrence number. For example, suppose you had a repetitive lattice defined as follows:

```
W1: WATCH,FILENAME=''%s-%03ld.w1''
Q1: QUAD,L=0.1,K1=1
D: DRIFT,L=1
Q2: QUAD,L=0.1,K1=-1
CELL: LINE=(W1,Q1,D,2*Q2,D,Q1)
BL: LINE=(100*CELL)
```

The element `W1` appears 100 times. Each instance will result in a new file being produced. Successive instances have names like "*rootname*-001.w1", "*rootname*-002.w1", "*rootname*-003.w1", and so on up to "*rootname*-100.w1". (If instead of "%03ld" you used "%ld", the names would be "*rootname*-1.w1", "*rootname*-2.w1", etc. up to "*rootname*-100.w1". This is generally not as convenient as the names don't sort into occurrence order.)

The files can easily be plotted together, as in

```
% sddsplot -column=t,p *-???.w1 -graph=dot -separate
```

They may also be combined into a single file, as in

```
% sddscombine *-???.w1 all.w1
```

In passing, note that if `W1` was defined as

```
W1: WATCH,FILENAME=''%s.w1''
```

or

```
W1: WATCH,FILENAME=''output.w1''
```

only a single file would be produced, containing output from the last instance only.

Notes:

1. Confusion sometimes occurs about some of the quantities related to the `s` coordinate in this file when in parameter mode. Please see Section 4 above.

2. This element can adversely affect parallel efficiency. Use of the `START_PASS`, `END_PASS`, `INTERVAL`, and `FLUSH_INTERVAL` options can help reduce the impact. Also, particle output is the most expensive, by far.

10.129 WIGGLER—A wiggler or undulator for damping or excitation of the beam.

A wiggler or undulator for damping or excitation of the beam.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
RADIUS	M	double	0.0	Peak bending radius. Ignored if K or B is non-negative.
K		double	0.0	Dimensionless strength parameter.
B	T	double	0.0	Peak vertical magnetic field. Ignored if K is non-negative
DX		double	0.0	Misaligment.
DY		double	0.0	Misaligment.
DZ		double	0.0	Misaligment.
TILT		double	0.0	Rotation about beam axis.
POLES		long	0	Number of wiggler poles
FOCUSING		short	1	If 0, turn off vertical focusing (this is unphysical!)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a wiggler or undulator. There are two aspects to the simulation: the effect on radiation integrals and the vertical focusing. Both are included as of release 15.2 of elegant.

If the number of poles should be an odd integer, we include half-strength end poles to match the dispersion, but only for the radiation integral calculation. For the focusing, we assume all the poles are full strength (i.e., a pure sinusoidal variation). If the number of poles is an even integer, no special end poles are required, but we make the unphysical assumption that the field at the entrance (exit) of the device jumps instantaneously from 0 (full field) to full field (0).

The radiation integrals were computed analytically using Mathematica, including the variation of the horizontal beta function and dispersion. For an odd number of poles, half-strength end-poles are assumed in order to match the dispersion of the wiggler. For an even number of poles, half-length end poles are assumed (i.e., we start and end in the middle of a pole), for the same reason.

The vertical focusing is implemented as a distributed quadrupole-like term (affecting ony the vertical, unlike a true quadrupole). The strength of the quadrupole is (see Wiedemann, *Particle*

Accelerator Physics II, section 2.3.2)

$$K_1 = \frac{1}{2\rho^2}, \tag{165}$$

where ρ is the bending radius at the center of a pole. The undulator is focusing in the vertical plane.

The wiggler field strength may be specified either as a peak bending radius ρ (RADIUS parameter) or using the dimensionless strength parameter K (K parameter). These are related by

$$K = \frac{\gamma \lambda_u}{2\pi\rho}, \tag{166}$$

where γ is the relativistic factor for the beam and λ_u is the period length.

10.130 ZLONGIT—A simulation of a single-pass broad-band or functionally specified longitudinal impedance.

A simulation of a single-pass broad-band or functionally specified longitudinal impedance.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
CHARGE	C	double	0.0	Deprecated—use CHARGE element
BROAD_BAND		long	0	broad-band impedance?
RA	Ohm	double	0.0	shunt impedance, Ra=V$\hat{2}$/P
RS	Ohm	double	0.0	shunt impedance (Rs=Ra/2)
Q		double	0.0	cavity Q
FREQ	Hz	double	0.0	frequency (BROAD_BAND=1)
ZREAL		STRING	NULL	<filename>=<x>+<y> form specification of input file giving real part of impedance vs f (BROAD_BAND=0)
ZIMAG		STRING	NULL	<filename>=<x>+<y> form specification of input file giving imaginary part of impedance vs f (BROAD_BAND=0)
BIN_SIZE	S	double	0.0	bin size for current histogram (use 0 for autosize)
N_BINS		long	128	number of bins for current histogram
MAX_N_BINS		long	0	Maximum number of bins for current histogram
WAKES		STRING	NULL	filename for output of wake
WAKE_INTERVAL		long	1	interval in passes at which to output wake
WAKE_START		long	0	pass at which to start to output wake
WAKE_END		long	9223372036854775807	pass at which to stop to output wake
AREA_WEIGHT		long	0	use area-weighting in assigning charge to histogram?
INTERPOLATE		long	0	interpolate wake?
SMOOTHING		long	0	Use Savitzky-Golay filter to smooth current histogram?
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing
SG_HALFWIDTH		long	4	Savitzky-Golay filter halfwidth for smoothing

A simulation of a single-pass broad-band or functionally specified longitudinal impedance.

Parameter Name	Units	Type	Default	Description
REVERSE_TIME_ORDER		long	0	Reverse time-order of particles for wake computation?
FACTOR		double	1	Factor by which to multiply impedance.
START_ON_PASS		long	0	The pass on which the impedance effects start.
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the impedance to full strength.
HIGH_FREQUENCY_CUTOFF0		double	-1	Frequency at which smoothing filter begins. If not positive, no frequency filter smoothing is done. Frequency is in units of Nyquist (0.5/binsize).
HIGH_FREQUENCY_CUTOFF1		double	-1	Frequency at which smoothing filter is 0. If not given, defaults to HIGH_FREQUENCY_CUTOFF0.
BUNCHED_BEAM_MODE		long	1	If non-zero, then do calculations bunch-by-bunch.
START_BUNCH		long	-1	In bunched beam mode, if non-negative, starting bunch number for computations
END_BUNCH		long	-1	In bunched beam mode, if non-negative, ending bunch number for computations
ALLOW_LONG_BEAM		long	0	Allow beam longer than covered by impedance data?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element allows simulation of a longitudinal impedance using a "broad-band" resonator or an impedance function specified in a file. The impedance is defined as the Fourier transform of the wake function

$$Z(\omega) = \int_{-\infty}^{+\infty} e^{-i\omega t} W(t) dt \qquad (167)$$

where $i = \sqrt{-1}$, $W(t) = 0$ for $t < 0$, and $W(t)$ has units of V/C.

For a resonator impedance, the functional form is

$$Z(\omega) = \frac{R_s}{1 + iQ(\frac{\omega}{\omega_r} - \frac{\omega_r}{\omega})}, \tag{168}$$

where R_s is the shunt impedance in $Ohms$, Q is the quality factor, and ω_r is the resonant frequency.

When providing an impedance in a file, the user must be careful to conform to these conventions. In addition, the units of the frequency column must be Hz, while the units of the impedance components must be Ohms. At present, `elegant` does not check the units for correctness.

Other notes:

1. The frequency data required from the input file is *not* ω, but rather $f = \omega/(2\pi)$.
2. The default smoothing setting (`SG_HALFWIDTH=4`), may apply too much smoothing. It is recommended that the user vary this parameter if smoothing is employed.
3. Impedance data can be created from a wake function using the script `wake2impedance`, which is supplied with `elegant`. This script also illustrates how to scale the data with the frequency spacing. The script uses `sddsfft`, which produces a folded FFT ($f \geq 0$) from a real function. The folded FFT representation involves multiplying the non-DC terms by 2. `elegant` expects this and internally multiplies the DC term by 2 as well.
4. Using the broad-brand resonator model can often result in a very large number of bins being used, as `elegant` will try to resolve the resonance peak and achieve the desired bin spacing. This can result in poor performance, particularly for the parallel version.
5. Wake output is available only in the serial version.

Bunched-mode application of the impedance is possible using specially-prepared input beams. See Section 6 for details. The use of bunched mode for any particular `ZLONGIT` element is controlled using the `BUNCHED_BEAM_MODE` parameter.

Explanation of <filename>=<x>+<y> format: Several elements in `elegant` make use of data from external files to provide input waveforms. The external files are SDDS files, which may have many columns. In order to provide a convenient way to specify both the filename and the columns to use, we frequently employ <filename>=<x>+<y> format for the parameter value. For example, if the parameter value is `waveform.sdds=t+A`, then it means that columns `t` and `A` will be taken from file `waveform.sdds`. The first column is always the independent variable (e.g., time, position, or frequency), while the second column is the dependent quantity.

10.131 ZTRANSVERSE—A simulation of a single-pass broad-band or functionally-specified transverse impedance.

A simulation of a single-pass broad-band or functionally-specified transverse impedance.
Parallel capable? : yes
GPU capable? : no
Back-tracking capable? : no

Parameter Name	Units	Type	Default	Description
CHARGE	C	double	0.0	Deprecated—use CHARGE element
BROAD_BAND		long	0	broad-band impedance?
RS	Ohm/m	double	0.0	shunt impedance (Rs=Ra/2=V$\hat{2}$/(2*P))
Q		double	0.0	cavity Q
FREQ	Hz	double	0.0	frequency (BROAD_BAND=1)
INPUTFILE		STRING	NULL	name of file giving impedance (BROAD_BAND=0)
FREQCOLUMN		STRING	NULL	column in INPUTFILE containing frequency
ZXREAL		STRING	NULL	column in INPUTFILE containing real impedance for x plane
ZXIMAG		STRING	NULL	column in INPUTFILE containing imaginary impedance for x plane
ZYREAL		STRING	NULL	column in INPUTFILE containing real impedance for y plane
ZYIMAG		STRING	NULL	column in INPUTFILE containing imaginary impedance for y plane
BIN_SIZE	S	double	0.0	bin size for current histogram (use 0 for autosize)
INTERPOLATE		long	0	interpolate wake?
N_BINS		long	128	number of bins for current histogram
MAX_N_BINS		long	0	Maximum number of bins for current histogram
SMOOTHING		long	0	Use Savitzky-Golay filter to smooth current histogram?
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing
SG_HALFWIDTH		long	4	Savitzky-Golay filter halfwidth for smoothing

ZTRANSVERSE continued

A simulation of a single-pass broad-band or functionally-specified transverse impedance.

Parameter Name	Units	Type	Default	Description
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
FACTOR		double	1	Factor by which to multiply x and y impedances.
XFACTOR		double	1	Factor by which to multiply x impedance.
YFACTOR		double	1	Factor by which to multiply y impedance.
WAKES		STRING	NULL	filename for output of wake
WAKE_INTERVAL		long	1	interval in passes at which to output wake
WAKE_START		long	0	pass at which to start to output wake
WAKE_END		long	9223372036854775807	pass at which to stop to output wake
START_ON_PASS		long	0	The pass on which the impedance effects start.
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the impedance to full strength.
HIGH_FREQUENCY_CUTOFF0		double	-1	Frequency at which smoothing filter begins. If not positive, no frequency filter smoothing is done. Frequency is in units of Nyquist (0.5/binsize).
HIGH_FREQUENCY_CUTOFF1		double	-1	Frequency at which smoothing filter is 0. If not given, defaults to HIGH_FREQUENCY_CUTOFF0.
X_DRIVE_EXPONENT		long	1	Exponent applied to x coordinates of drive particles
Y_DRIVE_EXPONENT		long	1	Exponent applied to y coordinates of drive particles
X_PROBE_EXPONENT		long	0	Exponent applied to x coordinates of probe particles

A simulation of a single-pass broad-band or functionally-specified transverse impedance.

Parameter Name	Units	Type	Default	Description
Y_PROBE_EXPONENT		long	0	Exponent applied to y coordinates of probe particles
BUNCHED_BEAM_MODE		long	1	If non-zero, then do calculations bunch-by-bunch.
START_BUNCH		long	-1	In bunched beam mode, if non-negative, starting bunch number for computations
END_BUNCH		long	-1	In bunched beam mode, if non-negative, ending bunch number for computations
ALLOW_LONG_BEAM		long	0	Allow beam longer than covered by impedance data?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element allows simulation of a transverse impedance using a "broad-band" resonator or an impedance function specified in a file. The impedance is defined as the Fourier transform of the wake function

$$Z(\omega) = \int_{-\infty}^{+\infty} e^{-i\omega t} W(t) dt \tag{169}$$

where $i = \sqrt{-1}$, $W(t) = 0$ for $t < 0$, and $W(t)$ has units of $V/C/m$. Note that there is no factor of i in front of the integral. Thus, in `elegant` the transverse impedance is simply the Fourier transform of the wake. This makes it easy to convert data from a program like ABCI into the wake formalism using `sddsfft`.

For a resonator impedance, the functional form is

$$Z(\omega) = \frac{-i\omega_r}{\omega} \frac{R_s}{1 + iQ(\frac{\omega}{\omega_r} - \frac{\omega_r}{\omega})}, \tag{170}$$

where R_s is the shunt impedance in $Ohms/m$, Q is the quality factor, and ω_r is the resonant frequency.

When providing an impedance in a file, the user must be careful to conform to these conventions. In addition, the units of the frequency column must be Hz, while the units of the impedance components must be Ohms/m. At present, `elegant` does not check the units for correctness.

Other notes:

1. The frequency data required from the input file is *not* ω, but rather $f = \omega/(2\pi)$.
2. The default smoothing setting (`SG_HALFWIDTH=4`), may apply too much smoothing. It is recommended that the user vary this parameter if smoothing is employed.

3. Impedance data can be created from a wake function using the script `trwake2impedance`, which is supplied with `elegant`. This script also illustrates how to scale the data with the frequency spacing. The script uses `sddsfft`, which produces a folded FFT ($f \geq 0$) from a real function. The folded FFT representation involves multiplying the non-DC terms by 2. `elegant` expects this and internally multiplies the DC term by 2 as well.

4. Using the broad-brand resonator model can often result in a very large number of bins being used, as `elegant` will try to resolve the resonance peak and achieve the desired bin spacing. This can result in poor performance, particularly for the parallel version.

5. Wake output is available only in the serial version.

Bunched-mode application of the impedance is possible using specially-prepared input beams. See Section 6 for details. The use of bunched mode for any particular `ZTRANSVERSE` element is controlled using the `BUNCHED_BEAM_MODE` parameter.

11 Examples

Example runs and post-processing files are available in a separate tar file. The examples are intended to demonstrate program capabilities with minimal work on the user's part. However, they don't pretend to cover all the capabilities.

Each demo is (typically) invoked using a command (usually a C-shell script) that can both run `elegant` and post-process the output. The post-processing is often handled by a lower-level script that is called from the demo script. These lower-level scripts are good models for the creation of customized scripts for user applications.

The examples are organized into a number of directories and subdirectories. In each area, the user will find a "Notebook" file (a simple ASCII file) that describes the example and how to run it.

Many examples for storage ring simulations reside in the `PAR` subdirectory. The PAR (Particle Accumulator Ring) is a small storage ring in the APS injector that is good for quick examples because of its size.

Here's a helpful tip in searching the examples on UNIX/LINUX systems: suppose one wants to find an example of the `frequency_map` command. One can search all the elegant command files very quickly with this command:

```
find . -name '*.ele' | xargs fgrep frequency_map
```

Similarly, to find all examples that use `CSBEND` elements, one could use

```
find . -name '*.lte' | xargs fgrep -i csbend
```

- `acceptance` — Use of the acceptance feature when tracking collections of particles.
  - `energyScan1` — Tracking a FODO line with various apertures, with variation of the initial momentum offset.
  - `fodoScan1` — Tracking a FODO line with various apertures, with scanning of the quadrupole strengths.
  - `transportLineAcceptance` — Determine transverse and momentum acceptance of a transport line using tracking. Example by M. Borland (ANL).
- `alphaMagnet` — Optimization of the strength of an alpha magnet to compress the beam from a thermionic rf gun.
- `APSRing` — Examples for the APS storage ring
  - `beamMoments` — 6D beam moments calculation with errors
  - `ibsAndTouschekLifetime` — Compute touschek lifetime with IBS-inflated emittances
  - `ibsVsEnergy` — Compute IBS as a function of energy.
  - `ionEffects1` — Basic simulation of ion effects.
- `beamBasedAlignment` — Determines quadrupole offsets based on simulated beam-based alignment procedure.
- `beamBreakup` — Example of simulating beam-driven deflecting rf mode in a simple linac.
- `bendErrors` — Analysis of the effect of errors on the matrix elements for a four-dipole bunch compression chicane.
- `boosterRamp` — Examples of simulating ramping in a booster.
  - `elementByElement` — Example of simulating ramping in a booster, using the NSLS-II booster lattice (R. Fliller).
  - `ILMATRIX` — Example of ramping using ILMATRIX for faster tracking.
- `bpmOffsets1` — Example of loading BPM offsets from an external file and then correcting the orbit with those offsets.
- `bunchCompression` — Examples of using a four-dipole chicane for bunch compression.
  - `backtrack-bunchCompCSRLSCWake` — Simluation of bunch compression with CSR, LSC, and wakes. Both forward and backward tracking are performed.
  - `bunchComp` — Four examples revolving around a four-dipole chicane bunch compressor. Simulations include basic compression, sensitivity to timing, phase, and beam energy.
  - `bunchCompJitter` — Simulation of a linac with a bunch compressor, including phase and voltage errors in the linac.
  - `bunchCompJitter2` — Simulation of a linac with a bunch compressor, including phase and voltage errors in the linac. In this case, the errors are generated externally.
  - `bunchCompLSC` — Inclusion of longitudinal space charge in simulation of a linac with a bunch compressor.

- bunchCompOptimize — Example of using tracking to optimize a linac and bunch compressor including a 4th-harmonic linearizer.

- chromaticAmplitudes — Example of minimizing chromatic amplitude functions in a simple beamline.
- chromaticResponse — Example of computing the chromatic transfer functions R16(s) and R26(s) as described in P. Emma and R. Brinkmann, SLAC-PUB-7554.
- constructOrbitBump1 — Illustration of how to make an orbit bump using BPM offsets and the orbit correction algorithm.
- coupling — Examples of coupling calculation and correction.
  - couplingCorrection1 — Scripts to perform coupling correction for the APS ring, emulating what is done in APS operations. These scripts are now part of the elegant distribution.
  - couplingCorrection2 — Example of using cross-plane response matrix and vertical dispersion to correct the coupling.
- customBeamDistributions — Examples of making custom beam distributions for tracking with elegant.
  - customLongitudinal — Example of making customized longitudinal distributions.
  - doubleBeam1 — Example of how to make a double-gaussian time distribution using two runs. The resultant beam would be used in a subsequent run using the sdds_beam command.
  - example1 — Gaussian energy distribution, linearly-ramped time distribution, and uniform transverse distributions.
  - parabolic — Gaussian longitudinal distribution combined with parabolic transverse distributions.
- cwiggler — Examples of using the CWIGGLER element.
  - cwig+kickmap — Example of simulating a simple wiggler with CWIGGLER, making a kickmap from trackings, then validating the kickmap.
  - cwiggler1 — A simple example of dynamic aperture with a set of sinusoidal wigglers, using the CWIGGLER element.
  - cwiggler2 — An simple example of dynamic aperture with a set of two-component horizontal wigglers, using the CWIGGLER element.
- DATuneScan — Performs a scan of the tunes in a storage ring and determines the variation in dynamic aperture.
- defeatLinkage — Example of how to defeat the automatic link between the gradient and other multipoles in a dipole and the strength of the dipole itself.
- ellipseComparison — Example of comparing beam ellipse from tracking to ellipse implied by the twiss parameters.

- `emitProc` — Various applications of the program sddsemitproc, which processes quad-scan emittance measurements.
  - `emitProc1` — Simple example with constant measurement errors.
  - `emitProc2` — Measurement errors are supplied in the data file.
  - `emitProc3` — Includes the presence of dispersion, with constant measurement errors.
  - `emitProc4` — Quadrupole scan values are supplied from an external source.
  - `emitProc5` — Includes acceleration as part of the beamline.
- `fiducialization` — Examples for fiducializaton of a beamline.
  - `fiducial1` — Example of fiducialization with a fiducial bunch and a perturbed bunch. The system in question is a linac with 50 structures, a four dipole chicane, then 50 more structures
- `followIndividualParticles` — Tracking a bunch of particles, then extracting and plotting the trajectories of a few particles.
- `full457MeV` — Tracking of the APS linac with a PC gun beam, up to the entrance of the LEUTL undulator.
- `GENESIS2.0` — Example of running SDDS-compliant GENESIS 1.3 with output from elegant for LCLS.
- `geneticOptimizer1` — Illustration of using the geneticOptimizer script together with elegant.
- `ILMatrixFromTracking` — Determination of the values for ILMATRIX based on analysis of tracking data.
- `injRingMatch` — Matching of a transport line to a storage ring.
  - `injRingMatch1` — Illustration of finding the periodic solution for a ring, then matching a transport line to that solution.
  - `injRingMatch2` — Illustration of finding the periodic solution for a ring, then matching a transport line to that solution. In this case, a single run is used.
  - `movingElements` — Example of matching a transport line to a ring with movable quadrupoles but fixed total length.
- `LCLS` — LCLS-I tracking example from P. Emma, November 2007.
  - `wakes` —
- `linacDispersion1` — Example of determining the initial dispersion error in a linac.
- `LongitudinalSpaceCharge` — Examples related to longitudinal space charge.
  - `LSCOscillationExample` — Example of longitudinal space charge oscillations in a drift space.
- `lsrMdltr` — Various examples of using the LSRMDLTR (Laser Modulator) element

  - `example1` — Simple example using LCLS-I-like parameters
  - `example2` — Includes a time-profile on the laser.
  - `example3` — Simulation of laser slicing for a storage ring.

- `matching` — Various examples of lattice matching and optimization.
  - `beamSizeMatch1` — Example of adjusting the initial beam parameters to match the measured beam sizes at a set of diagnositcs.
  - `betaMatching` — A simple two-stage matching example.
  - `IDCompensation` — Example of compensating for insertion device focusing effects.
  - `linacMatching1` — Example of three-part matching of a linac with a bunch compressor.
  - `linearize2` — Example of reducing nonlinearities in phase space using the REMCOR element to remove linear correlations first.
  - `matchMeasuredBetas` — Optimization of lattice quadrupoles to create a model that reproduces measured beta functions.
  - `matchTwoEnergies` — Example of matching beams with two different initial energies in a linac. The beams are affected by common quadrupoles, but also by quadrupoles unique to each beam.
  - `multiPartMatching1` — Complex example of multi-part matching for a linac with several splice points.
  - `multiPartMatching2` — Example of storage ring matching with three types of cells.
  - `spectrometer1` — Optimizes a simple spectrometer to maximize energy resolution.

- `MBALatticeDAWithErrors` — Example of performing DA vs momentum offset tracking when the lattice has strong sextupoles that make the orbit difficult to correct.

- `multibunchCollectiveEffects` — Examples of multi-bunch collective effects for APS storage ring and other cases.
  - `APS-24Bunch-CBI` — Includes main and harmonic cavities, beamloading, rf feedback, beam feedback, and short-range impedance.
  - `ILMatrixFromTracking` — Example of using tracking to set up the ILMATRIX element for fast tracking. This is useful for increasing the speed of collective effects simulations.
  - `linacBunchTrain1` — Includes main linac cavities, dipole HOMs, and monopole HOMs for a simple linac, showing beam breakup.
  - `linacWithHOMs` — Includes main linac cavities, dipole HOMs, monopole HOMs, and wakes for part of LCLS. The number of bunches is varied using templates.
  - `transientBeamLoading` — Includes main and harmonic cavities, beamloading, rf feedback, beam feedback, and short-range impedance. 48 bunches are grouped into four short trains to show the effect of transient beamloading.

- `multiStepErrors1` — Example of multi-step addition and correction of errors for a storage ring.

- `NSLS-II-GirderMisalignment` — Simulation of girder misalignment for NSLS-II, by S. Kramer (BNL) and M. Borland (ANL).

- `outboardTrajCorr` — Examples of using the response matrix computed by elegant to perform trajectory correction with a script.
  - `outboardTrajCorr1` — Compares trajectory correction inside elegant to correction performed with an external script.
  - `outboardTrajCorr2` — Compares trajectory correction inside elegant to correction performed with an external script. Includes BPM offsets.
- `PAR` — Numerous examples using the small APS Particle Accumulator Ring.
  - `accumulate` — Simulates adding particles to an already-stored beam.
  - `alphaExpansion` — Example of computing momentum compaction (alpha) to higher order using tracking.
  - `broadBandImpedance` — Example of using ZLONGIT, ILMATRIX, and SREFFECTS to simulate a broad-band impedance in a storage ring.
  - `bunchLengthening` — Simulation of a passive bunch-lengthening cavity using the RFMODE element.
  - `chromCorrection` — Simple chromaticity correction with two families. Also illustrates saving and loading correction results.
  - `chromTracking` — Illustration of using tracking to determine variation of tune with momentum.
  - `chromTracking2` — Similar to chromTracking, but includes determination of the momentum dependence of the beta functions.
  - `CSR` — Example of tracking with APS Particle Accumulator Ring with a Coherent Synchrotron Radiation impedance.
  - `DANormSigma` — Determination of dynamic aperture in terms of beam size.
  - `daOpt` — Example of optimization of dynamic acceptance.
  - `dynamicAperture` — Determination of dynamic aperture for a series of momentum errors.
  - `dynamicApertureWithSynchMotion` — Example of dynamic aperture with radiation damping and synchrotron motion.
  - `ejectionOptimization` — Tuning of a multi-turn extraction system using several kickers.
  - `elasticScatteringTracking` — Tracking to determine elastic scattering lifetime and loss distribution.
  - `emittanceOptimization` — Direct optimization of the emittance using linear optics tuning.
  - `fineDynamicAperture` — High-resolution dynamic aperture including a map of where particles are lost.
  - `fixedLVsRegularOrbit` — Illustration of the difference between orbits computed with fixed path length (fixed rf frequency) and fixed beam energy (variable rf frequency).
  - `fmaWithDispersiveOrbit` — Performs frequency map analysis including the momentum-dependence of the horizontal orbit.

- `frequencyMap` — Example of frequency map analysis
- `frequencyMap-x-delta` — Example of frequency map analysis for (x, delta)
- `gasScatteringLifetime` — Simple computation of gas scattering lifetime using a fixed pressure and gas mixture.
- `gasScatteringLifetimePresFile` — Computation of gas scattering lifetime using a file giving the pressure around the ring.
- `ILMatrixScan` — Set up ILMATRIX element, then scan the tune.
- `inelasticScatteringTracking` — Tracking to determine inelastic scattering lifetime and loss distribution.
- `moments` — Computes 6D beam moments with coupling errors.
- `momentumAperture` — Computes the s-dependent momentum aperture without errors.
- `offMomentumDA` — Another computation of off-momentum dynamic aperture
- `offMomentumTwiss` — Computation of off-momentum twiss parameters.
- `offMomentumTwiss2` — Computation of off-momentum twiss parameters vs s.
- `quadScan` — Computation of twiss parameters as quadrupoles are varied according to an external table.
- `randomMultipoles` — Dynamic aperture including random multipole errors in the quadrupoles and sextupoles.
- `synchrotronTune` — Simple example of tracking with synchrotron motion.
- `tracking` — Visualization of motion in x-x' and y-y' phase space.
- `trajOrbitCorrect` — Correct the first-turn trajectory, then correct the orbit.
- `tswa` — Example of obtaining amplitude-dependence of tunes from tracking.
- `TSWATracking` — Uses tracking and post-processing to determine tune variation with amplitude.
- `tuneExcitation` — Use a swept kick to excite the horizontal tune, observing excitation of the synchrotron tune as well.
- `tuneOptimization` — Correct the tunes and chromaticities.
- `twissCalculation` — Simple calculation of the twiss parameters
- `twoCavityMoments` — Calculation of 6D beam moments in the presence of main and harmonic rf cavities.

• `parallel` — Various runs illustrating a few features of the parallel version.

- `DA` — Dynamic aperture calculation.
- `FMA` — Frequency map analysis.
- `LMA` — Local momentum aperture calculation.
- `swarmOptimizer` — Simple example of using the particle-swarm optimizer.

• `pepperPot` — Examples of using the PEPPER_POT element

- `basic` — Basic example of simulating a pepper-pot plate.

- `pepperPotScan` — Example of simulating a pepper-pot plate with emittance analysis.

- `periodicTwissRFCA` — Demonstration that one can't have periodic beta functions in a FODO cell array with linac structures.
- `pulsedSextInjection` — Illustration of optimizing the sextupoles of pulsed sextupole kickers for injection into a storage ring.
- `rampTunesWithBeam` — Example of ramping tunes while tracking beam. In this case, we ramp the tunes across the difference coupling resonance.
- `rfDeflectingCavity` — A simple example of using a traveling wave rf deflector (RFDF).
- `RFTMEZ0` — Tracking through a TM-mode rf cavity based on an off-axis expansion starting from Ez(z) at r=0.
- `scanParameters` — Examples of scanning parameters of beamline elements.
  - `scanParameters1` — Scan two quadrupoles together.
  - `scanParameters2` — Scan the phase of an rf cavity and look at synchrotron oscillations.
- `scriptElement` — Examples of using the SCRIPT element
  - `elegantShower` — Use of the SCRIPT element to execute the electron-gamma shower simulation code SHOWER as part of an elegant run.
  - `mergeBeams` — Using the SCRIPT element to merge several beams into a simulation that already has a beam.
  - `slitArray` — Simulation of an array of slits using the SCRIPT element.
- `sddsoptimizeExample` — Example of using the program sddsoptimize to optimize the results of elegant simulations. In this example, we vary a strength fudge factor for a set of quadrupoles in a transport line in order to attempt to match measured H and V response matrices.
- `serverExample` — Example of using elegant in server mode to update lattice functions when magnet strengths change.
- `SPEAR3` — Various examples using an early SPEAR3 lattice
  - `dynamicAperture` — Compute DA for several error seeds, including multipole errors.
  - `latticeErrors` — Compute variation in lattice functions with errors, including correction of the orbit, tunes, and chromaticities.
- `staticPlusDynamicErrors` — Example of combining static and dynamic errors in one simulation.
- `storageRingRfNoise` — Example of including rf phase and amplitude noise in a tracking simulation.
- `straightDipoleModels` — Examples of setting up models of transverse gradient or longitudinal gradient dipoles with fringe effects.

  - `ccbend1` — Example of using a generalized gradient expansion to create a symplectic model of a transverse gradient dipole with soft fringe effects using CCBEND.
  - `lgbend1` — Example of using a generalized gradient expansion to create a symplectic model of a 5-segment lonitudinal gradient dipole with soft fringe effects using CCBEND.

- `transportLineHigherOrderDispersion` — Determine higher-order dispersion in a transport line using tracking.
- `transportLineSteering` — Examples of steering transport lines.
  - `coupledTransportLineSteering` — A simple example of a strongly-coupled transport line.
  - `twoPlaneSteering` — A simple example of trajectory correction in a transport line.
- `twissDerivatives` — Example of how to compute slopes of beta, alpha, and dispersion as a function of initial momentum for a transport line.
- `twoBunchPhasing` — Example of putting two bunches through a linac with the linac phased to the first bunch.
- `varyPlotExample` — Example of varying a beamline parameter and computing beam properties, then plotting those properties vs s.
- `wakesAndImpedances` — Examples of wakes and impedances.
  - `csrImpedance` — Comparison of using CSR impedance (from csrImpedance) and CSRCSBEND.
  - `transverse1` — Compare use of transverse wake and impedance methods for a damped oscillator.

12 The rpn Calculator

The program `rpn` is a Reverse Polish Notation programmable scientific calculator written in C. It is incorporated as a subprogram into `elegant`, and a number of the SDDS programs. It also exists as a command-line program, `rpnl`, which executes its command-line arguments as `rpn` operations and prints the result before exiting. Use of `rpn` in any of these modes is extremely straightforward. Use of the program in its stand-alone form is the best way to gain familiarity with it. Once one has entered `rpn`, entering "help" will produce a list of the available operators with brief summaries of their function. Also, the `rpn` definitions file `rpn.defns`, distributed with `elegant`, gives examples of most `rpn` operation types.

Like all RPN calculators, `rpn` uses stacks. In particular, it has a numeric stack, a logical stack, and a string stack. Items are pushed onto the numeric stack whenever a number-token is entered, or whenever an operation concludes that has a number as its result; items are popped from this stack by operations that require numeric arguments. Items are pushed onto the logical stack whenever a logical expression is evaluated; they are popped from this stack by use of logical operations that require logical arguments (e.g., logical ANDing), or by conditional branch instructions. Items enclosed in double quotes are pushed onto the string stack; items are popped from this stack by use of operations that require string arguments (e.g., formatted printing).

`rpn` supports user-defined memories and functions. To create a user-defined memory, one simply stores a value into the name, as in "1 sto unity"; the memory is created automatically when `rpn` detects that it does not already exist. To create a user-defined function, enter the "udf" command; `rpn` will prompt for the function name and the text that forms the function body. To invoke a UDF, simply type the name.

A file containing `rpn` commands can be executed by pushing the filename onto the string stack and invoking the "@" operator. `rpn` supports more general file I/O through the use of functions that mimic the standard C I/O routines. Files are identified by integer unit numbers, with units 0 and 1 being permanently assigned to the terminal input and terminal output, respectively.

13 Change Log

13.1 Highlights of What's New in Version 2023.3

Here is a summary of what's changed since release 2023.2. Historical change logs are collected in Section 13.

13.1.1 Bug Fixes for Elements

- The SPEEDBUMP element and SCRAPER elements share a common parameter POSITION, which indicates the position of the surface closest to the reference trajectory. The SCRAPER element correctly interprets the sign of the POSITION parameter to indicate whether the surface extends across the reference trajectory. For example, with INSERT_FROM=''-x'' and POSITION=0.010, the scraper extends from the negative side 10 mm across the reference trajectory. SPEEDBUMP, on the other hand, interprets the POSITION parameter as with different conventions depending on the direction of insertion. The SCRAPER_CONVENTION parameter was added to SPEEDBUMP to change the behavior to the SCRAPER approach. By default, the old behavior is used.

13.1.2 Bug Fixes for Commands

- The computation of trajectory and orbit response matrices had a bug when link_elements was used to link various steering elements. The response was computed without the links.
- The insert_elements command previously did not warn the user when nothing was inserted, which could have surprising consequences. This is now flagged as an error, with an option to issue a warning instead.
- The linear acceptance calculations provided by twiss_output were invalid when the nominal trajectory was outside the aperture. This was been fixed.
- A bug was corrected in the computation of the orbit response matrix when multiple steering_element commands were used in combination with use_response_from_computed_orbits=1 in the correct or correction_matrix_output command. The bug would cause generation of an incorrect matrix or even a crash.

13.1.3 New and Modified Elements

- The CSBEND element now provides a new symplectic, nonlinear fringe-field implementation that is similar to the one available for the CCBEND element, based on theoretical work by R. Lindberg (APS), who also did most of the coding. A companion program, curvedDipoleFringeCalc, computes the required seven integrals for the entrance and exit fringes.
- The MRFDF element now supports the START_PASS, END_PASS, START_PID, and END_PID parameters, per a request from forum user Seb_Wilkes.
- The CCBEND, CSBEND, KQUAD, and LGBEND elements have a new parameter, SR_IN_ORDINARY_MATRIX which allows including the effect of classical synchrotron radiation in ordinary matrix computations (e.g., for twiss_output or matrix_output). This would allow, for example, estimating the optical effects of energy losses around the circumference of a storage ring. (The moments_output provides a fully self-consistent treatment.) This is in response to a request from forum user esmahmadi.

- When RFCA or RFCW elements are used with CHANGE_T=1, a warning will be issued if any other time-dependent elements are present. This is to remind the user that such a combination may yield wrong results or even crashes.
- The CORGPLATES element was added, providing the longitudinal wake for a pair of corrugated plates [63].
- The ALPH element, which simulates transport through an alpha magnet [5], was improved to support a separate element for momentum filtration, which allows visualizing the effect more easily.

13.1.4 New and Modified Commands

- The correction_matrix_output command can now output response and inverse matrices for coupled, i.e., unified, trajectory correction.
- The function of the interrupt-file feature of the optimize command was improved, allowing control of the time interval between checks.
- The load_parameters command can now optionally load the same page from a file repeatedly. This can be used to reassert a baseline configuration prior to other changes, for example.
- The insert_elements command now supports defining the region of insertion by bracking with two named elements.
- The twiss_output command now provides the third-order momentum compaction. However, users are cautioned that the analytical matrices for most elements are limited to second-order, so using tracking-derived matrices is necessary where supported. This was added by M. Wallbank (FNAL).
- The CPICKUP and CKICKER elements, used to model optical stochastic cooling, were improved thanks to work by M. Wallbank (FNAL). The main improvement is that incoherent mode can now be used in the parallel version.
- The run_setup command now offers the ability to set the initial s coordinate, using the s_start parameter. This was requested by V. Sajaev (ANL).

13.1.5 Other Changes

- Performance improvements were made to serial and parallel versions in matrix memory handling, numerical precision, and integrator-based elements (CSBEND, CCBEND, LGBEND, etc.). Speed-ups vary between 5% and 70%, with 30-50% being typical. Tracking results will be different from previous versions at machine precision level. This work was done by N. Kuklev (APS).
- Maximum multipole order in input files is now limited to 19 (previously any order was valid). Please let authors know if this is too low.

13.1.6 Changes Specific to the MPI Parallel Version

- None

13.1.7 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.1.8 Changes to Related Programs and Files

The `elegant` distribution includes many programs and scripts that perform computations with `elegant` output data, provide interfaces with other programs, or prepare data for use with `elegant`. These are listed in Section 8.

Changes to these tools in this release include

- None.

13.2 Highlights of What's New in Version 2023.2

Here is a summary of what's changed since release 2023.1. Historical change logs are collected in Section 13.

13.2.1 Bug Fixes for Elements

- A memory leak was fixed in the `WAKE` element for multi-bunch mode. This was reported by forum user `Seb_Wilkes`.

13.2.2 Bug Fixes for Commands

- A bug was fixed in the `obstruction_data` command that would cause a crash if a second command was given.

13.2.3 New and Modified Elements

- The `PARTICLE_OUTPUT_FILE` file from `BRAT` elements now includes the direction cosines.

13.2.4 New and Modified Commands

- The `ion_effects` command now supports use of a Poisson solver for computation of fields from ions. This was implemented by J. Calvey (APS), with assistance from Y. P. Sun and M. Borland (both APS). It uses the widely-available FFTW library.
- The `bpm_centroids` output from the `run_setup` command how includes two columns giving the BPM weights.
- The `optimization_variable` command now accepts step sizes as a function of the allowed range, using the `fractional_step_size` parameter.
- The `optimization_setup` command supports "resetting" of optimization variables upon restarts of the algorithm based on a threshold. This helps prune variables that have a small effect on the outcome.

- The loss output file from touschek_scatter now contains several parameters to help more easily identify the scattering location. This was inspired by a question from forum user blanco-garcia.
- The vary_element command now has a disable parameter. If set to a non-zero value, the command is ignored.

13.2.5 Other Changes

- None

13.2.6 Changes Specific to the MPI Parallel Version

- None

13.2.7 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.2.8 Changes to Related Programs and Files

The elegant distribution includes many programs and scripts that perform computations with elegant output data, provide interfaces with other programs, or prepare data for use with elegant. These are listed in Section 8.

Changes to these tools in this release include

- Fixed a bug in ibsEmittance that resulted in the emittance ratio being forced to 1 when the -emityInput option was used.
- ibsEmittance has a new option, -fixEmity, that allows forcing the vertical emittance to a constant value when finding equilibrium parameters in a ring. This was requested by forum user Seb_Wilkes.

13.3 Highlights of What's New in Version 2023.1

Here is a summary of what's changed since release 2022.2.

13.3.1 Bug Fixes for Elements

- The CCBEND element had a bug in the path-length correction feature that affected runs in which the optimization of fractional strength error or transverse offset was repeated. This could result in a mismatch between the rf frequency and path-length, for example. M. Venturini (LBNL) reported the issue.
- Z. Duan (IHEP) reported an issue with misalignment methods 1 and 2 for CSBEND elements when when TILT was nonzero, which he identified as an error in the code that incorrectly used TILT and ETILT in combination. This was fixed, but in the process it proved advisable to change the order of operations to conform to the original one in elegant; in particular,

mislignments now occur in the untilted frame in all cases except CCBEND elements. In addition, we added the ability to select either the linearized or exact Venturini algorithm by adding two new modes. In particular,

  - Method 0 — This selects the original method, which was the only one available before version 2021.1. The misalignment is referenced to the entrance face. Yaw and pitch errors are not supported.
  - Method 1 — This selects a linearized method based on M. Venturini's work [58], with misalignment referenced to the entrance face. Yaw and pitch errors are supported.
  - Method 2 — This selects a linearized method based on M. Venturini's work [58], with misalignment referenced to the magnet center. Yaw and pitch errors are supported.
  - Method 3 — This selects an exact method based on M. Venturini's work [58], with misalignment referenced to the entrance face. Yaw and pitch errors are supported.
  - Method 4 — This selects an exact method based on M. Venturini's work [58], with misalignment referenced to the magnet center. Yaw and pitch errors are supported.

- The C matrix component for MATR elements was ignored, with the exception of the C[4] (path-length) component.

13.3.2 Bug Fixes for Commands

- None.

13.3.3 New and Modified Elements

- CCBEND now has a parameter, REFERENCE_CORRECTION that permits turning on correction of residual trajectory errors due to limited accuracy in optimization of the reference trajectory.
- RFCA and RFCW now optionally include body focusing terms for a pure sinusoidal traveling wave. This deficiency was pointed out by B. Simona (PSI).
- The CKICKER element, used as the kicker for simulation of optical stochastic cooling, was improved by the main author, A. J. Dick. Improvements include
  - Corrected the upper bound of integration in XYZ.
  - Corrected bugs in computation of incoherent effects.
  - The envelope function is now applied correctly to both coherent and incoherent kicks.
  - Corrected some mistakes in the documentation.
- The LGBEND element now supports yaw and pitch misalignments.

13.3.4 New and Modified Commands

- The touschek_scatter command was interpreting ignored_portion values less than or equal to 10^{-6} as zero, under the assumption that typically $\sim$1 million particles would be used. This cut-over now occurs at 10^{-9}. Forum user Skamarokha reported this issue.

13.3.5 Other Changes

- None

13.3.6 Changes Specific to the MPI Parallel Version

- None

13.3.7 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.3.8 Changes to Related Programs and Files

The `elegant` distribution includes many programs and scripts that perform computations with `elegant` output data, provide interfaces with other programs, or prepare data for use with `elegant`. These are listed in Section 8.

Changes to these tools in this release include

- The `straightDipoleFringeCalc` program now allows the user to specify the arc length of the `CCBEND` element, rather than relying on the internal calculation, which may not be accurate.

13.4 Highlights of What's New in Version 2022.2

Here is a summary of what's changed since release 2022.0.

13.4.1 Bug Fixes for Elements

- The open-side feature of the `MAXAMP` element behaved incorrectly when the dimension in one plane was not set, as reported by Z. Duan (IHEP).
- When using `moments_output` computations for a beamline that contained `SBEN` elements, a less accurate model was used compared to the same computation with `CSBEND` elements. V. Sajaev (ANL) helped find this bug.
- The pole-factor parameters of `CWIGGLER` gave strange results when the number of integration steps per half period was odd. This was reported by forum user Skamarokha. It was fixed by forcing `STEPS_PER_PERIOD` to be $4n$, where n is an integer.
- The `START_PID` and `END_PID` parameters of `WATCH` did not permit selecting a single particle. This issue was identified with the help of C. Li (DESY).

13.4.2 Bug Fixes for Commands

- The `correct` command had a bug related to threaded and one-to-one trajectory correction if removal of pegged correctors was not invoked.

13.4.3 New and Modified Elements

- The `CWIGGLER` element can now be split using the `divide_elements` command or the `element_divisions` parameter of the `run_setup` command. In addition, multiple `CWIGGLER` elements that are separated only by `MARK` or `WATCH` elements are considered part of the same element. This was suggested by forum user `Skamarokha`.

- The LGBEND element now allows the user to provide z-dependent aperture data via an SDDS file.

- The SPEEDBUMP element now provides information for logging of global coordinates of lost particles.

- The HMON, VMON, and MONI elements now provide the number of particles seen on the most recent turn for use in optimization. These are, in essence, virtual BPM sum signals.

- The BMXYZ element now allows specifying the range over which the optional constant field is applied.

- Added ability for BMXYZ and BRAT elemente to use sections of field maps with defined symmetry, which reduces storage requirements.

- The BMXYZ element now applies apertures defined by MAXAMP, APCONTOUR (with STICKY=1), and aperture_data in the interior.

- The APCONTOUR element now accepts multiple contours combined via user-defined logic.

- The WAKE, TRWAKE, ZLONGIT, and ZTRANSVERSE elements now allow limiting which bunches are subjected to short-range impedance effects, using the START_BUNCH and END_BUNCH parameters. This was suggested by forum user Seb_Wilkes and can provide significant reduction in run time for some types of simulations.

- The RFDF element now allows restricting effects based on particle ID values. This was also suggested by forum user Seb_Wilkes.

- The KAverage and pAverage values were wrong in output files for WATCH elements for the parallel version. This was pointed out by C. Li (DESY).

13.4.4 New and Modified Commands

- The correct command (orbit and trajectory correction) now supports Tikhonov regularization of the singular value spectrum in creating the inverse matrices, following some ideas of V. Sajaev (ANL).

- The optimization_term command now recognizes two new quantities: sMaxTransmitted and sMaxTransmittedMonitor, giving the maximum s coordinate to which particles are transmitted to any point or to a monitor, respectively. This can be used for threading of beam through a transport line.

- Added 1d-scan and RCDS [62] methods to the optimize command.

- Added the slope_limit and coord_limit parameters to the global_settings command. These allow controlling the limiting values for particle motion, beyond which particles are considered lost.

13.4.5 Other Changes

- None

13.4.6 Changes Specific to the MPI Parallel Version

- None

13.4.7 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.4.8 Changes to Related Programs and Files

The `elegant` distribution includes many programs and scripts that perform computations with `elegant` output data, provide interfaces with other programs, or prepare data for use with `elegant`. These are listed in Section 8.

Changes to these tools in this release include

- The `straightDipoleFringeCalc` program now allows the user to specify the arc length of the `CCBEND` element, rather than relying on the internal calculation, which may not be accurate.

13.5 Highlights of What's New in Version 2022.0

Here is a summary of what's changed since release 2021.4. Historical change logs are collected in Section 13.

13.5.1 Bug Fixes for Elements

- A bug was fixed in `CSBEND` that resulted in incorrect recording of global loss coordinates.
- An bug was fixed with `CSBEND` in that the matrix-based computations were sensitive to the `EDGE_ORDER` parameter whereas the tracking was not for `EDGE_EFFECTS1` or `EDGE_EFFECTS2` equal to 3 or 4. The code now uses a more consistent approach. This issue was pointed out by L. Nadolski (SOLEIL).
- A bug was fixed in `IBSCATTER` that gave incorrect results when `PARALLEL_INTEGRATION` was nonzero.
- A bug was fixed in the `FTRFMODE` element that resulted in the `N_CAVITIES` parameter being ignored. This was pointed out by forum user `Siwei_Wang`.
- A bug was fixed in the `CWIGGLER` element that resulted in the `FIELD_OUTPUT` file failing to reflect the values of `POLE_FACTOR1`, `POLE_FACTOR2`, and `POLE_FACTOR3`, even though these were included in the beam dynamics. This was pointed out by forum user `Skamarokha`.

13.5.2 Bug Fixes for Commands

- A bug was fixed that resulted in incorrect reporting of the element definition in informational messages issued by the `insert_elements` command.

- The sdds_beam command previously accepted negative values for duplicate_stagger[4] (the time offset between duplicated bunches. This doesn't make sense and can cause problems with multi-bunch wakes. A warning is now issued. Forum user Siwei_Wang pointed out the issue.

- A bug was fixed for touschek_scatter in that the scattering was not performed relative to the closed orbit. This was reported by G. Penn (LBNL).

- A bug was fixed for error_element that caused Pelegant to crash when several error_element commands assigned errors to the same quantity.

13.5.3 New and Modified Elements

- The CCBEND element has a new symplectic fringe modeling capability based on theory and developed by R. Lindberg (APS).

- The LGBEND element was added, supporting modeling of segmented longitudinal-gradient dipole magnets with fringe effects. It is based on the same theory as CCBEND.

- Added the ability to sum two field maps for BRAT. This uses the new ADDITIONAL_FILENAME, MAIN_FACTOR, and ADDITIONAL_FACTOR parameters.

- Added the ability to apply constant field components inside the hard edge boundary of the magnet to BMXYZ.

- The FSE parameter on CSBEND can now be set to any value. Previously, any value smaller than -1 was treated as -1.

13.5.4 New and Modified Commands

- The global_settings command has a new parameter, warning_limit, the controls how many similar warnings will be shown in detail before such warnings are suppressed. The default is 10.

- Added the element_occurence parameter to the change_start command, which allows changing the starting location in a beamline.

- Added the change_end command, which allows changing the end of the beamline.

- Added the include_commands command, which allows including commands from another file.

- The apertures defined by the obstruction_data command are now reflected inside CCBEND elements.

- The correct_tunes and chromaticity commands now allow saving the response and correction matrices to SDDS files.

13.5.5 Other Changes

- The unified warning system has now been completed. All warnings from commands go through this system and are collected in the summary at the end of the run.
- The type of the `particleID` field in beam output files was changed from `long` (typically a 32-bit signed integer) to `ulong64` (a 64-bit signed integer). Internally, only 52 bits are used. This is a step toward increasing the number of particles that can be tracked, which is presently limited to 2^{31}.

13.5.6 Changes Specific to the MPI Parallel Version

- A bug was fixed in the `sdds_beam` command that resulted in a crash when multiple beam input files were given.

13.5.7 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.5.8 Changes to Related Programs and Files

The `elegant` distribution includes many programs and scripts that perform computations with `elegant` output data, provide interfaces with other programs, or prepare data for use with `elegant`. These are listed in Section 8.

Changes to these tools in this release include

- The program `straightDipoleFringeCalc` was added. It performs fringe integral and other computations needed for setting up the fringe models in `CCBEND` and `LGBEND`.
- `touschekLifetime` was modified so that the `FN` and `FP` columns are normalized to the circumference, so that integration vs `s` will give the total loss rate. The units of the columns were updated.
- `TFBFirSetup`, which generates filter coefficients for transverse feedback, had several bugs as pointed out by Siwei Wang on the forum. These were corrected.
- `computeRBGGE`, which generates generalized gradient expansions in rectangular regions, now has auto-tuning options to require a minimum number of multipoles or derivatives.

13.6 Highlights of What's New in Version 2021.4

Here is a summary of what's changed since release 2021.3.

13.6.1 Bug Fixes for Elements

- A memory leak was fixed in the `FMULT` element.

- As pointed out by P. Anisimov, there were problems with the IBSCATTER element when more than one was inserted in the beamline. This was fixed. In addition, the code now detects negative growth rates (typically in the transverse planes) and correctly refuses to operate in "random" mode, since that mode is only able to increase the emittances.
- As pointed out by G. Penn (LBNL), the KSEXT element gave incorrect results for synchrotron radiation if K1 or J1 was significant. This was fixed. This change resulted in some reorganization of common routines used by KQUAD, KSEXT, KOCT, and CCBEND, so small differences in results may be seen.
- The CALIBRATION factor on HKICK, VKICK, EHKICK, and EVKICK and the HCALIBRATION and VCALIBRATION factors on HVKICK and EHVKICK was applied twice in the computation of the response matrices for trajectories. That is, the actual calibration factor would be the square of the given factor.
- A memory bug was found in CSBEND that would sometimes cause a crash when REFERENCE_CORRECTION=1 if the moments_output command was given. This was reported by G. Penn (LBNL).

13.6.2 Bug Fixes for Commands

- As pointed out by G. Penn (LBNL), the lengthError parameter in the closed_orbit output file was always zero, even when fixed_length=0. This was fixed.

13.6.3 New and Modified Elements

- Added the FACTOR parameter for the FMULT element, allowing all multipole components to be scaled by a common factor. Also added the UNTILTED_MATRIX parameter, which can be used to force elegant to numerically compute the matrix for the untilted element; see the manual page for discussion.
- Added the PARALLEL_INTEGRATION parameter to IBSCATTER, which permits disabling the default parallelization of s-dependent integrals for intrabeam scattering.
- The WATCH element has a new parameter AUTO_REFERENCE that allows automatically determining the reference frequency from the rf cavities.

13.6.4 New and Modified Commands

- The rf_setup command now does a better job of setting the rf frequency in the presence of a closed orbit, which may have a different length than the nominal orbit. It also includes the ability to offset the rf phase, which allows the user to compensate for small differences between the calculation of the energy loss per turn from radiation integrals and tracking. These improvements were inspired by questions from G. Penn (LBNL).
- The save_lattice command can now write the entire lattice as a single beamline, if output_seq=2. This was requested by forum user blanco-garcia.

13.6.5 Changes Specific to the MPI Parallel Version

- Added the multiply_np_by_cores parameter to bunched_beam and bunched_beam_moments. If non-zero, then the number of particles generated is n_particles_per_bunch times the number of working cores.

13.6.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.6.7 Changes to Related Programs and Files

The `elegant` distribution includes many programs and scripts that perform computations with `elegant` output data, provide interfaces with other programs, or prepare data for use with `elegant`. These are listed in Section 8.

Changes to these tools in this release include

- None.

13.7 Highlights of What's New in Version 2021.3

Here is a summary of what's changed since release 2021.2.

13.7.1 Bug Fixes for Elements

- The voltage setpoint adjustment feature of `RFMODE` will now work in the serial version.
- There was a bug in the I/Q feedback for `RFMODE` that essentially added the beamloading voltage to the setpoint. This was pointed out by Z. Duan (IHEP), who also provided a correction to the code.

13.7.2 Bug Fixes for Commands

- Fixed a lattice lattice parser bug that would cause a crash when a single reflected element appeared with a quoted name.
- A bug was fixed in the `ramp_elements` and `modulate_elements` commands that would cause incorrect values to be asserted when multiple elements of the same name were targetted in the default differential mode.
- The `chromaticity` command was ignoring the `update_orbit` parameter and not fully updating the twiss parameters as the correction was performed. In cases with large orbits and strong sextupoles, this would cause a discrepancy between the claimed corrected chromaticity and the actual chromaticity (as reported by the `twiss_output` command). This was reported by V. Sajaev (ANL).

13.7.3 New and Modified Elements

- Thanks to A. J. Dick and P. Piot (NIU), two new elements were added that allow simulation of Optical Stochastic Cooling. The `CPICKUP` element is used to record information from the pickup undulator, while the `CKICKER` element is used to impart corresponding kicks to the beam. See the manual pages for more detail.
- The `RFMODE` element now checks that the `A0` coefficient is nonzero and normalizes other coefficients to it, as implied by the manual.

- The REFERENCE_CORRECTION feature of CSBEND was not handling non-zero EPITCH or EYAW values, as reported by G. Penn (LBNL).

13.7.4 New and Modified Commands

- The ignore_elements command no longer results in a printout of all the ignored elements.
- Several improvements were made to the ion_effects command: (a) Changed binning for multi-function fits, so that only ions that are in range in both planes are counted. (b) Normalize kicks by the charge in the fit region, rather than the total charge. (c) Fixed some bugs with single Gaussian fit method. These changes were made by J. Calvery (APS).

13.7.5 Changes Specific to the MPI Parallel Version

- None.

13.7.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.7.7 Changes to Related Programs and Files

The elegant distribution includes many programs and scripts that perform computations with elegant output data, provide interfaces with other programs, or prepare data for use with elegant. These are listed in Section 8.

Changes to these tools in this release include

- The weightedBunch script now allows users to control the random number seed.
- The computeRBGGE and computeCBGGE programs now provide units for the generalized gradients.

13.8 Highlights of What's New in Version 2021.2

Here is a summary of what's changed since release 2021.1. Historical change logs are collected in Section 13.

13.8.1 Bug Fixes for Elements

- Beam moments calculations with the moments_output command now work correctly for CSBEND, CSRCSBEND, KQUAD, KSEXT, QUAD, SBEND, and SEXT elements in the presence of misalignments. Previously, if the misalignments were large the results would be unreliable.
- Beam moments computations for BGGEXP have been improved to use a tracking-based method for determination of the diffusion matrix. This can be turned off by setting tracking_based_diffusion_matrix_particles to 0 in the moments_output command.

- Radiation calculations for BGGEXP have been changed to correct the dependence of energy loss and energy spread on energy offset, as well as to correctly model direct effects on transverse momenta when the symplectic integrator is used.
- The MAPSOLENOID element contained a global sign error in using the magnetic fields. To restore the previous (incorrect) behavior, multiply the FACTOR parameter by -1.

13.8.2 Bug Fixes for Commands

- The bunched_beam and bunched_beam_moments commands previously did not use the centroids from moments_output when use_moments_output_values=1. This was fixed.
- The -pipe commandline option was broken, as reported by V. Sajaev (APS). This was fixed.
- The correct command no longer complains about diverging orbits and no longer adjusts the iteration fraction if the orbit is smaller than the accuracy requirement.
- The matrix_output command now respects the full_matrix_only setting for SDDS output. Also, the Step parameter is now set correctly.

13.8.3 New and Modified Elements

- The BEAMBEAM element was added. At present, it provides beam-beam kicks from a rigid opposing beam with a defined charge, center, and sizes, using either a gaussian or uniform ellipsoidal charge distribution. This was requested by A. Blednykh and M. Blaskiewicz (BNL).
- The BGGEXP element has five new parameters, FACTOR0, FACTOR1, FACTOR2, FACTOR3, and FACTOR4 which permit scaling the solenoidal, dipolar, quadrupolar, sextupolar, and octupolar field components independently.
- The BGGEXP and BMXYZ elements have three new parameters, BXFACTOR, BYFACTOR, and BZFACTOR, which allow multiplying the indicated field components by the given factors. This may be unphysical, but can provide helpful diagnostics.

13.8.4 New and Modified Commands

- The global_settings command has a new parameter, allow_overwriting which controls whether elegant is allowed to overwrite existing files. By default, it is allowed.
- The moments_output command now provides control of how many tracking-based matrices can be stored for reuse, via the tracking_based_matrices_store_limit parameter.
- The closed_orbit command has a new parameter, immediate, which allows requesting immediate computation of the closed orbit rather than the default computation as part of the major action.

13.8.5 Changes Specific to the MPI Parallel Version

- None.

13.8.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.8.7 Changes to Related Programs and Files

The `elegant` distribution includes many programs and scripts that perform computations with `elegant` output data, provide interfaces with other programs, or prepare data for use with `elegant`. These are listed in Section 8.

Changes to these tools in this release include

- `computeCBGGE` can now compute the expansion for solenoidal fields if provided with B_z data on a cylinder. Ryan Lindber (APS) provided the theoretical basis and helped with debugging.

13.9 Highlights of What's New in Version 2021.1

Here is a summary of what's changed since release 2020.5. Historical change logs are collected in Section 13.

13.9.1 Bug Fixes for Elements

- The `EDRIFT` element was being modeled as a regular `DRIFT` element when the `concat_order` parameter of `run_setup` was set to a non-zero value. This was reported by M. Venturini (LBNL).
- The `STICKY` parameter on the `APCONTOUR` element was ignored and always set to 1.

13.9.2 Bug Fixes for Commands

- A bug was fixed for the `touschek_scatter` command that resulted in a crash when there was a `RECIRC` element in the beamline. This was reported by M. Jebramcik (DESY). A. Xiao (ANL) helped uncover the cause.
- The `ramp_elements` and `modulate_elements` commands now work correctly in conjunction with `load_parameters` with `change_defined_values=0`.
- The `final` file, which is requested from `run_setup`, now contains additional parameters giving the minimum and maximum values of the particle coordinates at the end of the system; e.g., `xpMaximum` gives the maximum x coordinate. These quantities, like (almost) all quantities in the `final` file, are available for use in `optimization_term` expressions.

13.9.3 New and Modified Elements

- Several elements have a new parameter, `N_SLICES`, that replaces the inconsistently-used and misleadingly-named `N_KICKS` parameter. The `N_KICKS` parameter is still available for use, but is deprecated.
  - `CCBEND`, `CSBEND`, `CSRCSBEND`, `FMULT`, `MULT`: The `N_KICKS` parameter was poorly named. It actually gave the desired number of full integrator steps, or "slices."

- KQUAD, KSEXT, KQUSE: The N_KICKS parameter was actually the number of total integrator substeps. E.g., for the fourth-order integrator, dividing N_KICKS by 4 gave the number of slices.

- CSBEND, KQUAD, KSEXT, and KOCT have a parameter MALIGN_METHOD that permits invoking new misalignment methods based on work of M. Venturini [58]. Gregg Penn (LBNL) helped verify the new methods. Based on the value of this parameter, misalignment calculations are changed as follows:
  - MALIGN_METHOD=0: use the existing method. The new YAW and PITCH (or EYAW and EPITCH for CSBEND elements) parameters are ignored.
  - MALIGN_METHOD=1: use M. Venturini's method, with misalignment parameters understood to be relative to the magnet entrance. The new YAW and PITCH (or EYAW and EPITCH for CSBEND elements) parameters become active. This is presently incompatible with the moments_output command.
  - MALIGN_METHOD=2: use M. Venturini's method, with misalignment parameters understood to be relative to the magnet center. The new YAW and PITCH (or EYAW and EPITCH for CSBEND elements) parameters become active. This is presently incompatible with the moments_output command.
- The HMON, VMON, and MONI elements can be used to store turn-by-turn BPM readings by setting the STORE_TURN_BY_TURN parameter to 1. This can be used, for example, to create position-based triggers using modulate_elements.
- The GKICKMAP element was added, which provides a generalized kickmap that is not specific to undulators or wigglers (unlike UKICKMAP).
- The SCATTER element now supports uniform distributions in addition to the default gaussian distribution using the new DISTRIBUTION parameter.
- The CSBEND matrix no longer requires use of the relatively slow tracking-based matrix option when the steering parameters are used. The element also supports the XSTEERING and YSTEERING parameters to allow individual control of whether the element is used for steering.
- The individual-element steering parameters for KSEXT elements are no longer ignored.

13.9.4 New and Modified Commands

- The parameters output of the run_setup command now includes string quantities in the ParameterValueString column. Numerical quantities are stored in the ParameterValue column, as before.
- The run_setup command has a new parameter suppress_parameter_defaults. If set to a nonzero value, the data stored in the parameters output file will not contain data that match the default values. This can result in much smaller files and faster loading, with the downside that future changes to the defaults would impact the ability to reproduce a run using a saved parameter file.
- The load_parameters command now makes consistent use of the ParameterValueString column in any input files. If the ParameterValue column is present, it is used for numerical quantities, while the ParameterValueString column is used for string quantities only. If only

ParameterValueString is present, elegant will attempt to scan the string values as needed for numerical values; this is not the preferred approach as it will degrade performance.

- The losses output of the run_setup command now provides the global coordinate angle in the horizontal plane when losses_include_global_coordinates=1. One can also now control the range of s coordinates for recorded particles using the losses_s_limit array.
- The correct_tunes and chromaticity commands now accept lists giving lower and upper strength limits for each family. In addition, the strength_log files are now compatible with load_parameters.
- The closed_orbit and correct commands now have a control that allows accepting a closed orbit results that exceeds the accuracy target, rather than considering this an error. This will allow correction or other computations to proceed in spite of poor convergence of the closed orbit.

13.9.5 Changes Specific to the MPI Parallel Version

- A problem was fixed with the particle-swarm optimizer that caused it to abort or return invalid results when a function evaluation yielded an invalid result. This was in response to issues raised by forum user marlibgin.

13.9.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.9.7 Changes to Related Programs and Files

The elegant distribution includes many programs and scripts that perform computations with elegant output data, provide interfaces with other programs, or prepare data for use with elegant. These are listed in Section 8.

Changes to these tools in this release include

- The program computeRBGGE, which allows computation of a generalized gradient expansion[50] for use with BGGEXP based on field data on a rectangular boundary [57], was improved to support automatic tuning of the number of gradients and multipoles. In addition, output of the implied field map is supported.
- The program computeCBGGE was added, which allows computation of a generalized gradient expansion based on field data on a circular-cylinder boundary [50]. It replaces the script computeGeneralizedGradients, providing better performance and features.
- The script generateBunchTrain was added. It allows generating particle input files for a fairly arbitrary multi-train bunch pattern.

13.10 Highlights of What's New in Version 2020.5

Here is a summary of what's changed since release 2020.4. Historical change logs are collected in Section 13.

13.10.1 Bug Fixes for Elements

- The SCATTER element had a bug in the implementation of the ENDONPASS parameter, which was being ignored if the value was 0.
- The CCBEND element had several misalignment-related issues. Most significantly, the ETILT parameter was ignored. In addition, the sense of DX and DY was inverted when when the ANGLE was negative. As a result of these fixes, misalignment effects from CCBEND will change. Setting ETILT=0 and DX_DY_SIGN=-1 will cause the code to revert to the old behavior.

13.10.2 Bug Fixes for Commands

- Frequency map analysis would crash in some circumstances when there was a CHARGE in the beamline. This bug was reported by G. Penn (LBNL).

13.10.3 New and Modified Elements

- The BGGEXP element now supports both normal and skew components. R. Lindberg (APS) implemented the symplectic integrator option for this.
- The BMAPXYZ element now has a FSE (Fractional Strength Error) parameter.
- The RADIAL_ORDER parameter of RFTMEZ0 can now be set to 0 to turn off radial dependence of fields.
- The POLYSERIES element was released, which allows transforming the beam using a set of arbitrary polynomials in the canonical coordinates. This existed in the code but was not documented. It was developed by L. Emery (APS) and inspired by work of Y. P. Sun (APS).
- The ETILT_SIGN parameter for CSBEND, CSRCSBEND, RBEN, and SBEN now defaults to 1. The previous default of -1, for backward compatibility, was potentially confusing.

13.10.4 New and Modified Commands

- Added the change_start command, which permits changing the starting location in a lattice. This was inspired by a request from Duan Zhe (IHEP).
- The obstruction_data command now supports multiple vertical planes of data.

13.10.5 Changes Specific to the MPI Parallel Version

- None.

13.10.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.10.7 Changes to Related Programs and Files

- The computeRBGGE program was added, allowing computation of a generalized gradient expansion[50] for use with BGGEXP based on field data on a rectangular boundary [57]. The code was developed by R. Lindberg (APS) with help from R. Soliday (APS) and M. Borland (APS).

13.11 Highlights of What's New in Version 2020.4

Here is a summary of what's changed since release 2020.3.

13.11.1 Bug Fixes for Elements

- The TAPERAPC element would mark all particles as lost when the RSTART and REND parameters had the same value. This was reported by forum user Cai_Meng.
- The FMULT element had a bug that produced *very* invalid results with the FSE parameter was non-zero, as reported by A. Xiao (APS). Also, the FSE and other parameters were not influencing matrix-related computations if changed during a run.
- Use of the XREADOUT and YREADOUT features of the MONI element resulted in a message about undefined rpn variables.

13.11.2 Bug Fixes for Commands

- The load_parameters command produced incorrect results in multi-step runs (i.e., n_steps¿1 in run_control) when a multi-page parameter file was used with multiple differential-mode load instructions for the same element. This was reported by V. Sajaev (APS).
- The HKICK, VKICK, and HVKICK elements were not responding properly to the ramp_elements and modulate_elements commands. This was reported by V. Sajaev (APS).
- A bug was fixed in ion_effects that sometimes caused unphysical sigma values in the bi- and tri-gaussian fits. The bug was found and fixed by J. Calvey (APS).
- A bug was fixed that resulted in a crash when coupled=1 was set for trajectory response matrix output from correction_matrix_output.
- A bug was fixed that resulted in a crash if a beamline definition contained unbalanced quotation marks. This was reported by X. Huang (APS).

13.11.3 New and Modified Elements

- The CCBEND, CSBEND, FMULT, KOCT, KSEXT, and KQUAD now support sixth-order symplectic integrators. The coefficients were provided by Y.P. Sun (APS), who also assisted in testing. The value of the N_KICKS parameter may be reduced by a factor of ∼5 if the sixth-order integrator is used in place of the fourth-order integrator, with essentially identical results but a ∼30% reduction in run time.
- The APCONTOUR element has two new parameters: STICKY and CANCEL. The STICKY parameter results in the aperture contour being applied inside subsequent CCBEND, CSBEND, CSRCSBEND, KQUAD, KSEXT, verb—KOCT—, and KQUSE elements, as well as at the end of other downstream

elements. This continues until another APCONTOUR element asserts a new contour, or uses CANCEL=1 to cancel the feature. This improvement was inspired by forum user dondreka.

13.11.4 New and Modified Commands

- Added bpm_centroid parameter to the run_setup command. This provides a facility similar to the centroid parameter, but instead of giving the centroids at all elements, it gives them at the BPMs only. This was requested by X. Huang (APS).
- The moments_output command now provides additional quantities for optimization at MARK locations. See the documentation for the MARK element.
- The modulate_elements command now provides the ability to use the pass number to compute the time, which is helpful in simulations where the time is offset by CHANGE_T=1 on RFCA elements.
- Several improvements were made to the ion_effects command by J. Calvey (APS). The gaussianfit option was added for the field_calculation_method parameter; this provides a gaussian fit to model the ion fields, as an alternative to using gaussian statistics or more complex fitting functions. Also, a new parameter, ion_output_interval, was added that allows increasing the interval between logging of ion data.
- The correct command accepts a new value, coupled, for the method parameter when mode is trajectory. This allows trajectory correction in strongly-coupled transport lines.
- The bunched_beam_moments command was added. This command is virtually identical to the venerable bunched_beam command, but instead of specifying the beam dimensions in terms of emittances, beta functions, etc., the user provides beam moments (e.g., beam size, divergence, etc.).

13.11.5 Changes Specific to the MPI Parallel Version

- The population_log file produced by parallel_optimization_setup erroneously recorded the values of the optimization variables in the wrong in an offset fashion.

13.11.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.11.7 Changes to Related Programs and Files

- None.

13.12 Highlights of What's New in Version 2020.3

Here is a summary of what's changed since release 2020.2.

13.12.1 Bug Fixes for Elements

- The KQUAD and KSEXT elements had a bug when computing beam moments with moments_output if the XKICK or YKICK values were nonzero. This was reported by G. Penn (LBNL).
- The MALIGN was affecting floor coordinates, which actually doesn't make much sense. This was changed, but the prior behavior can be restored by setting FLOOR=1.
- Lost particle coordinates inside CSBEND elements were recording the wrong value of the longitudinal coordinate. The reported coordinate was a mixture of the central path length and the individual particle's path length, whereas the correct coordinate should be just the central path length.
- The FRFMODE and TFRFMODE elements had a bug that caused them to use up all the available file descriptors on a system when many such elements were inserted. This was reported by J. England (SLAC).

13.12.2 Bug Fixes for Commands

- Issues were resolved with the computation of the s coordinate for sigma and centroid output files when invoking backtracking mode from run_setup.
- The ion_effects command had a bug in the automatic bin size selection code that could result in pathological changes in the bin size over several bunches. This was found and fixed by J. Calvey (APS).
- When saving a lattice with output_seq=1, RFCA and other elements with the PHASE_REFERENCE parameter would have this parameter set to very large values. This bug was reported by Z. Duan (IHEP).

13.12.3 New and Modified Elements

- The SCRIPT element can now be used in backtracking mode. See the manual page for details. This was requested by Y. Park (UCLA).
- The MATTER element now accepts PRESSURE and TEMPERATURE values, which are used to compute the density according to the ideal gas law, allowing easier simulation of scattering from gasses.
- The CSBEND element now supports steering fields via the XKICK and YKICK parameters. The FSE and FSE_DIPOLE parameters can also be used for horizontal steering.
- The CCBEND element now supports horizontal steering via the XKICK parameter. One can also use the FSE and FSE_DIPOLE parameters.
- The CSBEND element now supports a new symplectic nonlinear edge model, developed by R. Lindberg (APS).
- The BRAT and BMXYZ elements, which propagate particles through 3D field maps for bending and non-bending elements, respectively, now offer higher-order 2D interpolation as an alternative to the detail multi-linear interpolation. The also support storing the field data in single-precision arrays to reduce memory requirements.

- The BRAT and BMXYZ elements now supports testing of particle locations against the global-coordinate obstruction contours specified by the obstruction_data command.
- The BRAT element now supports floor coordinate transformations.
- The TFBDRIVER element now supports individual gain factors for each bunch. This was requested by M. Venturini (LBNL).
- The BOFFAXE element now supports a high-order off-axis expansion for z-dependent sextupole fields.

13.12.4 New and Modified Commands

- The twiss_output command can now compute lattice functions for a half periodic cell, i.e., a cell with mirror symmetry, by setting matched=-1. This was inspired by a forum question from mcarla.
- The obstruction_data command now supports periodic replication of the obstructions for rings, as well as a global cap on the minimum and maximum vertical coordinates.
- The tune_shift_with_amplitude command now uses an improved 2D polynomial fitting routine in all the tracking-based modes. Discussions with Y.P. Sun (APS) motivated this change.
- The corrector-pegging feature of the trajectory/orbit correction command correct now works better. In particular, when a corrector is pegged, the iteration stops until the response matrix can be recomputed. Printouts and other output now reflect this.

13.12.5 Changes Specific to the MPI Parallel Version

- None.

13.12.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.12.7 Changes to Related Programs and Files

- None.

13.13 Highlights of What's New in Version 2020.2

Here is a summary of what's changed since release 2020.1. Historical change logs are collected in Section 13.

13.13.1 Bug Fixes for Elements

- The CSRDRIFT element was not including longitudinal space charge as requested when CSR effects were turned off. This was reported by Y. Park (UCLA).

13.13.2 Bug Fixes for Commands

- The parameters file created by the run_setup command had negative values for certain drift lengths when backtracking was used. This was reported by Y. Park (UCLA).

13.13.3 New and Modified Elements

- Added MIN_NORMAL_ORDER, MAX_NORMAL_ORDER, MIN_SKEW_ORDER, and MAX_SKEW_ORDER to the KQUAD, KSEXT, and CCBEND elements. This allows easily restricting which systematic and random multipole orders are included without changing the data files.
- Added the T_REFERENCE parameter to RFCW, which is useful in backtracking in linacs.

13.13.4 New and Modified Commands

- Added the obstruction_data command, which permits specifying obstructions in global coordinates. At present, this is experimental and only enforced inside CSBEND, KQUAD, KSEXT, KOCT, and KQUSE elements, and at the end of elements.
- Added the losses_include_global_coordinates to the run_setup command, which allows requesting that the losses file contains global coordinates of lost particles (as opposed to only Frenet-Serret coordinates).
- The closed_orbit command now includes a parameter in the output file that indicates if the orbit determination has failed.
- Added the rfc_reference_output parameter to the run_setup command, which allows recording the internally-determined reference times for RFCA and RFCW elements. This can be useful in improving backtracking in linacs.
- The error_element command now supports using sampled values in an external files as the source for perturbation values.

13.13.5 Changes Specific to the MPI Parallel Version

- None.

13.13.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.13.7 Changes to Related Programs and Files

- sddsbrightness now computes the undulator linewidth (FWHM).

13.14 Highlights of What's New in Version 2020.1.1

13.14.1 Bug Fixes for Elements

- Edge effects are now correctly handled for the KQUAD element in backtracking.
- Improved backtracking for the CCBEND element (Cartesian Canonically-integrated Bend). Backtracking for this element is still doubtful for large-angle dipoles with significant gradients.
- The CCBEND element now incorporates a path-length correction to ensure that the central path length in tracking is the same as the user-defined arc length. This prevents issues with rf cavity setup and closed orbit determination. V. Sajaev (APS) reported this bug.
- The ETILT parameter of the CSBEND, CSRCSBEND, SBEN, and RBEN elements was not implemented correctly. Although the magnitude of the trajectory error was correct, it had the wrong sign compared to the dynamic effects (e.g., vertical dispersion). This bug was reported by G. Penn (LBNL), who also reported an error in adjustment of the path-length when ETILT was nonzero (no adjustment was made). A new parameter ETILT_SIGN allows changing the sign convention to match that of TILT.
- The FSE_DIPOLE and FSE_QUADRUPOLE values were being left at the user-defined values when determining the reference trajectory when REFERENCE_CORRECTION=1. This bug was reported by G. Penn (LBNL), and has been fixed.
- The handling of the DC term in ZTRANSVERSE was corrected. The code was ignoring the real part of the impedance at DC. The issue was pointed out by R. Lindberg (APS). The new script trwake2impedance illustrates how to create a transverse impedance from a wake function using sddsfft.
- The handling of the DC term in ZLONGIT was corrected. In particular, the term needed to be multipled by two internally. The new script wake2impedance illustrates how to create a longitudinal impedance from a wake function using sddsfft.
- The matrix for the MULT element, used in computation of twiss parameters and beam moments, was not being updated when parameters were changed during a run (e.g., with vary_element or error_element).
- Synchrotron radiation was computed incorrectly for KQUAD and KSEXT elements when the HKICK or VKICK steering parameters were non-zero, as reported by G. Penn (LBNL).

13.14.2 Bug Fixes for Commands

- The namelist parser was improved so that it now detects a common error, namely, a missing (or incorrectly typed) &end token.
- The bunched_beam command was giving inconsistent particle ID values between the serial and parallel versions when first_is_fiducial=1, as reported by Duan Zhe (IHEP).
- The twiss_output command would fail to deliver data to the s-dependent driving terms output file if no file was giving for twiss parameters, as reported by forum user felix_armborst. This is now flagged as an error.

13.14.3 New and Modified Elements

- Backtracking is now available for the RFCW and LSCDRIFT elements, as well as for CSRCSBEND with STEADY_STATE=1. The ENERGY element can also be used in backtracking, and is essential in some cases to ensure matching reference energy profiles between forward and backward tracking; an example is provided in the examples collection. Y. Park (UCLA) motivated the work and helped with testing.
- Work began on a mechanism for summarizing warnings at the end of a run. At present, the summary is incomplete.

13.14.4 New and Modified Commands

- Added more controls for mathematica-compatible output from the matrix_output. In particular, it is now possible to put the output in separately-named files, which makes importation into mathematica much simpler.
- Work began on a mechanism for summarizing warnings at the end of a run. At present, the summary is incomplete.

13.14.5 Changes Specific to the MPI Parallel Version

- None.

13.14.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.14.7 Changes to Related Programs and Files

- Added the scripts wake2impedance and trwake2impedance to translate wake function data into a form accepted by the ZLONGIT and ZTRANSVERSE elements, respectively.

13.15 Highlights of What's New in Version 2019.4.0

Here is a summary of what's changed since release 2019.3.0. Historical change logs are collected in Section 13.

13.15.1 Bug Fixes for Elements

- The ILMATRIX element lacked path-length terms related to the betatron amplitude, and also did not properly handle non-zero $\alpha_{x,y}$ and $\eta'_{x,y}$, as pointed out by forum user Teresia.
- A bug was fixed in back-tracking for SBEN elements with nonzero values for HGAP and FINT. This was reported by Y. Park (UCLA).

13.15.2 Bug Fixes for Commands

- The bunched_beam command was not generating particle ID values when use_moments_output_values=1, as reported by Z. Duan (IHEP).

13.15.3 New and Modified Elements

- Added SPARSE_INTERVAL parameter to the WATCH element, to allow sparsing coordinate output with regular spacing. This supplements the FRACTION parameter, which provides random sampling, and the START_PID and END_PID parameters, which provide sampling of a subset defined by particle ID.

- Added RPN_PARAMETERS parameter to the SCRIPT element, which directs the program to load SDDS parameter values from the script output file into rpn variables, where they may be used for optimization. This provides the user the ability to perform script-based analysis of particle distributions and then optimize the results of that analysis.

- The N_BINS parameter of the WAKE and TRWAKE elements now defaults to zero, which prevents some undesirable behavior when warnings are overlooked. This was requested by R. Lindberg (ANL).

- The EDRIFT, EHVCOR, EHCOR, EVCOR, CSBEND, KQUAD, KSEXT, KOCT, and UKICKMAP elements can now be used with back-tracking (see run_setup).

13.15.4 New and Modified Commands

- The chaos_map command is now available. As the name suggests, it is similar to a frequency map, but provides other measures of chaotic motion. This includes a promising new approach outlined by Y. Li *et al.* [56].

- The optimization_variable command has a new parameter differential_limits, which permits specifying that the lower and upper limits are being given relative to the initial value, rather than in absolute terms.

- The tune_footprint command has a new parameter separate_xy_for_delta, which permits specifying that tracking for the x and y momentum-dependent tunes should be done either in a combined fashion (default) or separately. The latter might be helpful if nonlinear coupling of y motion into the x plane causes the x tune to be poorly determined for small x amplitudes.

- Added egaussian mode to the ion_effects command. This mode results in computation of the fields from the electrons assuming a gaussian distribution, as normal, but determines the kick to the ions based purely on momentum conservation. This was inspired by the work of M. Blasciewicz (BNL) and implemented with J. Calvey (ANL).

- Added multiple_ionization_energy_peak and multiple_ionization_energy_rms parameters to the ion_effects command, allowing control of the peak and rms energy of ions produced by multiple ionization. This was done by J. Calvey (ANL).

13.15.5 Changes Specific to the MPI Parallel Version

- None.

13.15.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.15.7 Changes to Related Programs and Files

- None.

13.16 Highlights of What's New in Version 2019.3.0

Here is a summary of what's changed since release 2019.2.1. Historical change logs are collected in Section 13.

13.16.1 Bug Fixes for Elements

- The matrix for the `SBEN` element was incorrect when the length was negative (which is needed for back-tracking). A. Zholents (ANL) and Y. Park (UCLA) helped identify the problem.

- The integrator used for the `CSBEND` element with the expanded Hamiltonian (`EXPAND_HAMILTONIAN=1`) was very inaccurate and yieled poor results unless `N_KICKS` was large. This problem, pointed out by Z. Duan (IHEP), was fixed.

- The interpolation used for `UKICKMAP` would previously produce invalid values for particles near the upper ($y > 0$) and left ($x > 0$) edges of the grid. In essence, the interpolation assumed the kickmap was periodic in x and y.

13.16.2 Bug Fixes for Commands

- DA trimming (to make the `xClipped` and `yClipped` columns, as well as the `Area` parameter) was not working properly in the parallel version for full-plane runs. This was fixed. In addition, the algorithm was improved for both the serial and parallel versions to iterate the trimming until it converges.

- The multi-gaussian and multi-lorentzian feature of the `ion_effects` command, added in the last release, proved unreliable, as reported by B. Podobedov (BNL). Several improvements and bug fixes were implemented that should improve matters.

13.16.3 New and Modified Elements

- The `SHRFDF` element was added, which models a deflecting rf cavity using a space harmonic expansion. This was implemented by Y.P. Sun (APS). See [55] for details.

- The `LSCDRIFT` element, which models longitudinal space charge, can now have its effective length set automatically to correspond to the length of the upstream element.

13.16.4 New and Modified Commands

- The `run_setup` command has a new parameter, `back_tracking`, which allows invoking a limited back-tracking capability. See the entry for `run_setup` for more details. This is an experimental feature and users are encouraged to report problems to the forum.

- The `ion_effects` command was improved in several ways:

  - The default distribution fitting parameters were modified to give improved convergence

- The default distribution fitting criterion is now the sum of the maximum fractional absolute deviation over the histogram and the absolute fractional deviation of the ion charge. This makes it less likely that overfitting will result in large spikes in the distribution.
- The new `ion_histogram_max_bins` parameter allows restricting the maximum number of bins.
- The new `ion_histogram_min_per_bin` parameter allows setting a requirement on the minimum number of macro ions per bin.
- The new `freeze_ions_until_pass` and `freeze_electrons_until_pass` parameters allow "freezing" the motion of the ions and electrons until a specified pass number. This is useful for diagnostic purposes.
- The new `pressure_factor` parameter allows multiplying all the pressure profiles by a common factor.

- The `matrix_output` command can now print the full matrix in a form accepted by Mathematica.

13.16.5 Changes Specific to the MPI Parallel Version

- Some apparent MPI-related issues were resolved for the `ion_effects` command.

13.16.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.16.7 Changes to Related Programs and Files

- The `touschekLifetime` program now reports the value of `deltaLimit` in the output file, whether that value is given explicitly or computed via the rf voltage.
- Added the program `sdds5x5sigmaproc`, which computes the 5x5 sigma matrix (i.e., all elements except those related to the time coordinates) from a quadrupole scan.

13.17 Highlights of What's New in Version 2019.2.1

Here is a summary of what's changed since release 2019.1.1.

13.17.1 Bug Fixes for Elements

- Fixed a bug in tracking-based matrix computation for `CCBEND` that would result in the program hanging under some circumstances.
- Fixed a bug in the implementation of the expanded Hamiltonian for `MULT` elements.
- Fixed a bug in the `BMXYZ` element that caused a crash when multiple such elements were used.
- If `COUPLING` and `EYREF` were both non-zero for an `SREFFECTS` element, the `EYREF` value would be ignored, which is potentially confusing. This issue is now flagged as an error. B. Podobodov (BNL) brought the issue to our attention.

13.17.2 Bug Fixes for Commands

- The correct command, which performs trajectory or orbit correction, would fail to output corrector data for both planes in some cases. This was fixed.
- The correct command, also had a bug in reporting the "uncorrected" trajectory in the trajectory output file. Instead of giving the uncorrected trajectory, it was giving the trajectory after the penultimate correction iteration. This was reported by forum user shancai.
- The memory-efficiency of bucket assignments, invoked when using use_bunched_mode in sdds_beam, was improved, preventing crashes in some extreme cases for the serial version.

13.17.3 New and Modified Elements

- Added the APCONTOUR element, which provides an aperture or obstruction defined by an (x, y) contour in an SDDS file.
- Added the TAPERAPC element, which provides a tapered circular aperture.
- Added the TAPERAPE element, which provides a tapered elliptical aperture.
- Added the TAPERAPR element, which provides a tapered rectangular aperture.
- RFMODE now allows ignoring particles that are outside the binning region, using the ALLOW_UNBINNED_PARTICLES parameter.
- The required format for MATR (matrix from a text file) has changed slightly, as described on the manual page. The element also has a new parameter, FRACTION that allows interpolating the matrix elements with the identity matrix as one endpoint.

13.17.4 New and Modified Commands

- The ion_effects command now supports calculation of ion fields using a bi-gaussian distribution (sum of two gaussians) or bi-lorentzian distribution (sum of two lorentzians), as well as tri-gaussian and tri-lorentzian distributions. This which allows modeling the core and tails of the distribution more accurately. A number of parameters were added for control of fitting and output. J. Calvey (ANL) and R. Lindberg (ANL) co-developed this improvement.
- The sdds_beam command now offers control of which bunch is used for fiducialization of rf systems. It defaults to the first bunch (#0), which is a change from the previous (and frequently confusing) behavior of fiducializing to the entire beam.
- The aperture_search command now allows full-plane computations, i.e., computations covering both $y \geq 0$ and $y < 0$.

13.17.5 Changes Specific to the MPI Parallel Version

- None

13.17.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.17.7 Changes to Related Programs and Files

- The `touschekLifetime` program now reports the value of `deltaLimit` in the output file, whether that value is given explicitly or computed via the rf voltage.
- Added the program `sdds5x5sigmaproc`, which computes the 5x5 sigma matrix (i.e., all elements except those related to the time coordinates) from a quadrupole scan.

13.18 Highlights of What's New in Version 2019.1.1

Note that following release 35.1.0, version numbers changed to the form *year.release.minor*, where *year* is the four-digit year, *release* is the consecutive release number for the year, and *minor* is for internal APS use.

Here is a summary of what's changed since release 35.1.0. Historical change logs are collected in Section 13.

13.18.1 Bug Fixes for Elements

- The `BMXYZ` element previously would inject particles at $z = 0$ by default, which is usually not the desired behavior. Now, it injects at the start of the field map by default. The new `INJECT_AT_Z0` parameter can be used to recover the old behavior. In addition, drift spaces are now automatically included to compensate for differences between the length of the field map and the user-defined insertion length.

13.18.2 Bug Fixes for Commands

- The `load_parameters` command with `change_defined_values=0` did not work correctly when combined with `insert_elements` or `replace_elements`. This was reported by G. Penn (LBNL).
- The `insert_elements` command would sometimes fail to insert all the intended elements when `insert_before=1` when insertion between consecutive elements was required. This was reported by G. Penn (LBNL).

13.18.3 New and Modified Elements

- The `BOFFAXE` element was added. It allows integrating through a magnetic field defined by an off-axis expansion from on-axis gradients.
- Transfer matrices are now automatically computed for `BMXYZ` and `BGGEXP` elements.
- The `CSBEND`, `SBEND`, and `CCBEND` elements now support separate fractional strength errors (FSE) for the dipole and quadrupole terms.
- The `HKPOLY` element now supports and alternative, more general form for the drift Hamiltonian.
- The `UKICKMAP` element now has a flag to indicate that the kickmap is for a single period of an insertion device, which makes it easier to configure. It also has a new parameter, `KACTUAL`, for giving the K value independent of the field factor (which is applied to the kickmap).

13.18.4 New and Modified Commands

- The `global_settings` command now allows setting the default step sizes for tracking-based determination of element-by-element matrices using the new `tracking_matrix_step_size` parameter. The default values are the same as those used by the `analyze_map` command.
- The `analyze_map` command now allows changing the number of points in each dimension and the maximum fit order.
- the `correct_tunes` command now allows specifying a list of quadrupoles to be excluded from the tune knob.

13.18.5 Changes Specific to the MPI Parallel Version

- Fixed a bug that resulted in crashing of tracking-based matrix computation for certain numbers of processors.
- Fixed a bug in parallel hybrid simplex optimization, which would cause optimization to terminate prematurely if one processor encountered an invalid condition (e.g., undefined tunes).

13.18.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.18.7 Changes to Related Programs and Files

- None.

13.19 Highlights of What's New in Version 35.1.0

Here is a summary of what's changed since release 35.0.1. Historical change logs are collected in Section 13.

13.19.1 Bug Fixes for Elements

- The matrix for misaligned `FMULT` elements was incorrect. The misalignment was applied twice.
- The edge effects for `KQUAD` were broken for tracking only in version 35.0.1. This was fixed.

13.19.2 Bug Fixes for Commands

- The filter parameters (`start_occurence`, `end_occurence`, `s_start`, `s_end`, `after`, and `before`) of the `steering_element` command now work better when multiple such commands are given. In particular, overlapping intervals are detected and non-overlapping intervals are correctly implemented.

13.19.3 New and Modified Elements

- The HKPOLY element was added. It allows imparting kicks to the beam according to a Hamiltonians that are polynomial functions of (x, y) and (qx, qy). R. Lindberg (APS) helped develop the concept for this element.

13.19.4 New and Modified Commands

- The correct command has a new parameter, force_alternation that forces orbit or trajectory correction to continue with x/y alternation regardless of whether one plane appears to have converged.
- The set_reference_particle_output command was added. It allows defining a reference set of particle coordinates to which tracked coordinates will be compared for purposes of optimization.
- The optimization_setup command now allows setting the interval (in terms of function evaluations) between checks of the interrupt semaphore file. Previously, the file was checked only at the end of a simplex pass.

13.19.5 Changes Specific to the MPI Parallel Version

- None.

13.19.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.19.7 Changes to Related Programs and Files

- None.

13.20 Highlights of What's New in Version 35.0.1

Here is a summary of what's changed since release 34.4.0.

13.20.1 Bug Fixes for Elements

- The order of edge transformations for the CCBEND element was revised to make more physical sense.
- A bug was fixed in the expressions for integral-based fringe field effects in QUAD and KQUAD. The bug resulted in very small inconsistencies between the matrices when $K_1 \rightarrow -K_1$.
- Some small errors were found and fixed in the linear fringe treatment for the KQUAD and QUAD elements. Thanks to X. Huang (SLAC) for pointing out the problem.
- A bug was fixed in the NIBEND element that resulted in incorrect edge effects for ANGLE<0. This bug was apparently introduced in release 33.0.

- The SCRAPER element was not respecting changes to the DIRECTION or INSERT_FROM parameters made outside the lattice definition (e.g., using alter_elements or load_parameters). In addition, the interpretation of the INSERT_FROM=''x'' and INSERT_FROM=''y'' was incorrect, since these were supposed to correspond to a scraper inserted from both sides. Both problems were reported by forum user Youssef.
- The SPEEDBUMP element was not respecting changes to the DIRECTION or INSERT_FROM parameters made outside the lattice definition (e.g., using alter_elements or load_parameters).
- The RFCA element would bomb if no particles per present on a processor and CHANGE_T=1. This was found upon investigating a problem reported by G. Penn (ALS).
- The transport matrix for BGGEXP was being computed only to first order, which resulted in erroneous values for chromaticity, for example. This was reported by R. Linbdberg (APS).
- When CSRDRIFT elements were divided using the divide_elements command or element_divisions parameter of the run_setup command, the length was saved incorrectly to the parameters file (requested from run_setup). This was reported by Pau Gonzalez.

13.20.2 Bug Fixes for Commands

- None.

13.20.3 New and Modified Elements

- The CCBEND element now was a YAW parameter that permits changing the entrance and exit angles in a coordinated fashion. It also supports the FINT1, FINT2, and HGAP parameters for soft-fringe effects, as well as explicit multipoles from octupole to 18-pole (in addition to the existing support for systematic multipole errors). The multipoles at the entrance and exit can now be specified separately, using the EDGE1_MULTIPOLES and EDGE2_MULTIPOLES parameters.
- The FMULT element, which provides a general multipole with content specified by an SDDS file, now affects matrix-based computations (e.g., twiss parameters, chromaticities, and transfer matrix).
- The KSEXT element now provides a parameter for a normal quadrupole error, in addition to the existing skew quadrupole error. The utility of this was pointed out by Y.-P. Sun (APS) and X. Huang (SLAC).
- The BRANCH element now provides periodic branching, which permits modeling a periodic bypass, for example. This improvement was triggered by a question from forum user simone.dimitri.
- The global_settings command now has user overriding of default values, which means that whenever the user changes a value, it becomes the new default for any subsequent instances of the command in that run.
- The RFMODE element has additional features that help refine the agreement between the voltage obtained by rf feedback and the effective voltage seen by the beam.
- The WAKE and TRWAKE elements now accept acausal wakes, provided the user explicitly allows it with the ACAUSAL_ALLOWED parameter. This feature will be requested by R. Lindberg (APS).

- The LSRMDLTR and CWIGGLER elements now include *experimental* capabilities providing a transverse gradient in undulators or wigglers. In both cases, hard-to-correct residual trajectory and dispersion effects are seen, which are not yet understood. For this reason, these features are considered experimental.

13.20.4 New and Modified Commands

- The matrix_output command has two new parameters
  - print_element_data controls whether the element data is printed in addition to the matrices.
  - printout_format allows controlling the format of the printed elements.
- The analyze_map command has a new parameter, printout_format, allows controlling the format of the printed elements.
- The correct_tunes command has a new parameter, update_orbit, which allows controlling whether the orbit is updated during correction. The need for this arose from a problem encountered by I. Agapov (DESY).
- The chromaticity command has a new parameter, update_orbit, which allows controlling whether the orbit is updated during correction.
- The tracking used for matrix determination for elements, such as CCBEND, BGGEXP, and others, that rely on this, now takes advantage of parallel resources if Pelegant is used. This feature can be controlled using the newly-added parallel_tracking_based_matrices control in global_settings. The global_settings command also now offers the ability to control the number of points per phase space dimension that are used in matrix fitting, via the tracking_matrix_points parameter. The default value of this parameter has been set to 9—an increase from the minimalist value of 5 used in previous versions—in order to improve accuracy. Forum posts by J. Björklund Svensson (MAX-Lab) helped spur work on these features.
- The insert_sceffects command now supports averaging of beam size data turn-by-turn to reduce noise in transverse space charge simulation in rings, via the new averaging_factor parameter. This was suggested by V. Kornilov (GSI).

13.20.5 Changes Specific to the MPI Parallel Version

- None.

13.20.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.20.7 Changes to Related Programs and Files

- The FTABLE method for integration through 3D magnetic field maps now works in the abrat program via the -ftable commandline option. Previously, the control existed but resulted in no transformation of particles taking place.
- The program abrat now supports interpolation among multiple 2D field maps, which can be used, for example, to find the operating point in a magnet for which the field scales differently with current in different regions.
- For the longitCalcs script, the calculation of rf bucket height sometimes failed when a harmonic voltage was present; this was fixed. Also, the option to run without the GUI and put all results in a file was added.
- The makeWigglerFromBends script now includes the ability to add a gradient and specify the beam energy.
- A new program, sdds4x4sigmaproc is included that allows processing beam moments measurements from a quadrupole scan in a transport line to determine the 4x4 sigma matrix.

13.21 Highlights of What's New in Version 34.4.0

Here is a summary of what's changed since release 34.3.0.

13.21.1 Bug Fixes for Elements

- The FTABLE method for integration through 3D magnetic field maps now works in the BRAT element via the USE_FTABLE control. Previously, the control existed but resulted in no transformation of particles taking place.
- The YAW and PITCH parameters of the LTHINLENS and LMIRROR elements were overwriting the TILT parameters of the same elements.
- The B7 and B8 parameters of the CSRCSBEND elements were overwriting the B6 parameter of the same element.

13.21.2 Bug Fixes for Commands

- The share_tracking_based_matrices feature, controlled by the global_settings command, now works correctly. This can provide a considerable increase in performance when tracking-based matrices are required for many beamline elements.
- The insert_sceffects command and SCMULT element, used for space-charge simulation in rings, had a bug that caused the sign of the tune shift to be wrong for protons and positrons. This was reported by forum user hongjin.

13.21.3 New and Modified Elements

- MARK elements with FITPOINT=1 now create psix and psiy symbols for use in optimization, in addition to nux and nuy. This may be more intuitive for some users, as pointed out by forum user jgarland.

- When used for longitudinal feedback, the TFBDRIVER element now includes simulation of the feedback cavity resonance and driving circuit, using a circuit model developed by T. Berenc (APS).
- When K. Hwang's fringe model is used for the CSBEND element, automatic adjustment of the FSE value can optionally be invoked in order to null out trajectory errors that result from the fringe fields extending outside the magnet. This is obtained by setting FSE_CORRECTION=1.
- The TRACKING_MATRIX parameter of CSBEND can now be used to control the order of the tracking-based matrix, with a limit of third order. This provides an alternative to the 2nd-order analytical matrix.
- The CSBEND element now supports separate edge integral values for the entrance and exit fringes, using the FINT1 and FINT2 parameters. If not given, the FINT parameter is used as before.
- The CSBEND and CSRCSBEND elements now support a new symplectic edge effects mode, based on the linear approach of K. L. Brown. It is similar to the existing, non-symplectic default mode, but in most cases users won't see a difference.
- The CSBEND, KQUAD, KSEXT, KOCT, KQUSE, and MULT elements now support use of the expanded (to leading order) Hamiltonian by setting the EXPAND_HAMILTONIAN flag to 1. Note that no significant reduction in run time is observed with the expanded Hamiltonian.

13.21.4 New and Modified Commands

- None.

13.21.5 Changes Specific to the MPI Parallel Version

- None.

13.21.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.21.7 Changes to Related Programs and Files

- The FTABLE method for integration through 3D magnetic field maps now works in the abrat program via the -ftable commandline option. Previously, the control existed but resulted in no transformation of particles taking place.
- The program abrat now places the vertex, entry, and exit points (which are provided by the user) in the trajectory output file.

13.22 Highlights of What's New in Version 34.3.0, June 14, 2018

Here is a summary of what's changed since release 34.2.0.

13.22.1 Bug Fixes for Elements

- The CCBEND element had incorrect signs for the odd-order systematic multipoles when the bending angle was negative. There was also an issue with incorrect ordering of edge effects and coordinate transformations. R. Lindberg (APS) helped identify these problems.
- The KQUAD element had an issue with the order of the submatrices used for linear edge effects when tracking. This would cause small tune errors in tracking compared to the results of twiss_output.

13.22.2 Bug Fixes for Commands

- The beta-function scaling resulting from the twiss_scaling=1 setting for the elastic_scattering command was incorrect. Although the tracking results would be correct, this made it more difficult to optimize run time.
- The rf_setup command now handles $\alpha_c < 0$, a deficiency that was reported by P. Piot (NIU/FNAL). It also now uses $\eta = \alpha_c - 1/\gamma^2$ instead of α_c, though this rarely makes a significant difference.
- The computation of exact normalized emittances, requested with the global_settings command, had several issues that were fixed. First, the values assigned to the horizontal and vertical corrected and uncorrected emittances were permuted. Second, in the serial version, the corrected emittances (with dispersive terms removed) were computed incorrectly. This would impact the sigma and final files from the run_setup command. J. Björklund Svensson (MAX-Lab) reported problems that helped find these bugs.

13.22.3 New and Modified Elements

- CCBEND has a new parameter, EDGE_ORDER, that allows controlling the order of edge kicks.
- BRANCH has a new parameter, DEFAULT_TO_ELSE, which allows determining how the element behaves when tracking for closed orbits and the like.

13.22.4 New and Modified Commands

- The configuration command-line argument was added, which allows specifying a configuration file to be read before processing the input file. This file can also be specified with the ELEGANT_CONFIGURATION environment variable.
- The floor_coordinates command now creates additional columns in the output file, giving data on the next element in the lattice. Given that data is only provided at the end of elements, this provides an easier way to determine information at the start of elements.
- The insert_elements command now has a parameter, insert_before, that allows controlling whether elements are inserted before or after (default) the specified locations.
- The elastic_scattering command now includes data that indicates warning conditions in the file specified by log_file.

13.22.5 Changes Specific to the MPI Parallel Version

- Read buffering was re-enabled for parallel I/O to avoid performance problems on GPFS file systems. Write buffering is still disabled, since this seems to prevent data corruption on some file systems. Users may wish to configure this for their file system using the new `ELEGANT_CONFIGURATION` environment variable.

13.22.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.22.7 Changes to Related Programs and Files

- The scripts `elasticScatteringAnalysis` and `inelasticScatteringAnalysis`—which are used to analyze data from the `elastic_scattering` and `inelastic_scattering` commands—were replaced with compiled programs of the same name, giving a large reduction in run time.
- The script `longitCalcs`, which does rf calculations using an output file from `twiss_output`, now supports a commandline mode that is convenient for use in other scripts.
- The script `computeQuadFringeIntegrals` was added. It computes the fringe integrals and effective length for a quadrupole from gradient vs z data, producing a file suitable for configuring `KQUAD` elements.
- A bug was fixed in `elegant2astra` that would affect results for particles that are not highly relativistic. Forum user Biaobin reported the bug and provided the fix.
- A bug was fixed in `smoothDist6s` that resulted in strange longitudinal phase space when the average value of `t` was very large compared to the spread in `t`. Forum user `Marcello` reported the bug.

13.23 Highlights of What's New in Version 34.2.0, March 22, 2018

Here is a summary of what's changed since release 34.1. Historical change logs are collected in Section 13.

13.23.1 Bug Fixes for Elements

- None.

13.23.2 Bug Fixes for Commands

- The `frequency_map` command was incorrectly computing the diffusion rate as

$$d_r = \frac{\log_{10}\left(\Delta\nu_x^2 + \Delta\nu_y^2\right)}{N}, \tag{171}$$

instead of

$$d_r = \log_{10}\left(\frac{\Delta\nu_x^2 + \Delta\nu_y^2}{N}\right), \tag{172}$$

- The coupled_twiss_output command would sometimes crash when calculate_3d_coupling=0.

13.23.3 New and Modified Elements

- None.

13.23.4 New and Modified Commands

- The global_settings command has two new fields, mpi_io_force_file_sync and usleep_mpi_io_kludge, which can be used to solved MPI I/O problems that appear on some file systems. Z. Pan (LBNL) brought the problems to our attention.
- The floor_coodinates command now ignores MAXAMP elements when computing combined vertex points of strings of dipoles.
- The coupled_twiss_output command did not compute the tunes of the two modes, as pointed out by G. Wei (TJNAF). This was addressed with assistance from V. Sajaev (APS).

13.23.5 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.23.6 Changes to Related Programs and Files

- The program sddsbrightness now correctly includes the effect of J_x and J_y on the x and y emittances when the -coupling option is used.
- Added the script parmela2elegant, to convert PARMELA beam data (ASCII format) to a form acceptable by elegant.
- Fixed error in the atomic mass of CO_2 in the script ionTrapping.

13.24 Highlights of What's New in Version 34.1.0, 27 February 2018

Here is a summary of what's changed since release 34.0. Historical change logs are collected in Section 13.

13.24.1 Bug Fixes for Elements

- Restored the long-deprecated DIRECTION parameter for the SCRAPER element, as a convenience.
- Fixed a problem that caused the SCRIPT element to sometimes hang up in Pelegant if some processors did not have any particles after loading data from the script output file.
- The UKICKMAP element would sometimes fail to add synchrotron radiation effects during tracking even if asked; this would happen, for example, if there was no twiss_output or matrix_output command.

- The WIGGLER, UKICKMAP, CWIGGLER, and GFWIGGLER elements had an inconsistency in radiation integral computations, in that in some cases *gamma* was used when $\beta\gamma$ was intended. The differences were very small for any practical case.
- The BRAT element and the abrat commandline program for tracking particles through 3D field distributions had an error in the initial coordinate transformation, discovered by R. Lindberg (APS). In practical use, the error seems to have had a negligible effect on results. Also, the element was treated as a drift for matrix computations; now, the matrix is determined by tracking (which can be time-consuming).
- Synchrotron radiation calculations for KQUAD, KSEXT, and KOCT had a bug that resulted in only the last component being computed. For example, if steering or higher multipoles were included, those would override the effect of the main field.
- Previously, when the KQUAD element was split (with the divide_elements command or element_divisions in the run_setup command), soft-edge effects would be replicated at the interior boundaries. This was fixed.
- Soft-edge effects on the KQUAD element were not exactly symmetric. This would, e.g., introduce a slight asymmetry into an otherwise symmetric lattice. This has been fixed.

13.24.2 Bug Fixes for Commands

- The rf_setup command could not handle $\alpha_c < 0$, as discovered using files provided by P. Piot (NIU/FNAL). This was fixed.
- The analyze_map command would crash if SDDS output was not requested. This was fixed.

13.24.3 New and Modified Elements

- The CCBEND element, which integrates symplectically in Cartesian coordinates through a straight-pole combined-function bending magnet, was added.
- The BMXYZ element, which integrates particles through straight-element 3D magnetic field maps, now includes misalignment parameters. Multiple BMXYZ elements that use the same field map will share the data internally to reduce I/O and memory requirements.
- The EHKICK, EVKICK, and EHVKICK elements now include the RANDOM_MULTIPOLE_FACTOR and SYSTEMATIC_MULTIPOLE_FACTOR parameters.
- The BGGEXP element can now handle bending magnets. The non-symplectic integrator was replaced with a new method that is more accurate. R. Lindberg (APS) did most of the work on this.

13.24.4 New and Modified Commands

- During tracking, particles are no long checked against apertures after transitioning through zero-length elements that don't modify the aperture. This improves performance in lattices with many MONI, MARK, and similar elements.
- The analyze_map command can now output the matrix in SDDS format to second or third order, on request.

13.24.5 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.24.6 Changes to Related Programs and Files

- The program `sddsbrightness` now correctly includes the effect of J_x and J_y on the x and y emittances when the `-coupling` option is used.
- Added the script `parmela2elegant`, to convert PARMELA beam data (ASCII format) to a form acceptable by `elegant`.
- Fixed error in the atomic mass of CO_2 in the script `ionTrapping`.

13.25 Highlights of What's New in Version 34.0, 31 October 2017

Here is a summary of what's changed since release 33.1.1.

13.25.1 Bug Fixes for Elements

- A bug in the `IONEFFECTS` element was reported by J. Cavley (APS): when only one bunch was present, the electron beam coordinates were zeroed out.
- A bug in the `WATCH` element caused `elegant` to crash in `centroid` and `parameter` mode when the `WATCH` element was in a beamline branch that did not get executed on the first pass.
- In multi-step runs, the `STEERING_MULTIPOLES` input for the `EKICK`, `EHKICK`, and `EVKICK` elements was ignored except on the first step.

13.25.2 Bug Fixes for Commands

- A bug in the `ion_effects` command was reported by J. Cavley (APS): when only one bunch was present, the electron beam coordinates were zeroed out.
- The `center_arrival_time` feature of `sdds_beam` did not work correctly for the parallel version, as reported by Jonas Björklund.
- The `use_moments_output_values` qualifier of the `bunched_beam` command did not work for the parallel version.
- The `full_grid_output` mode of the `frequency_map` command provided incorrect results for the diffusion for particles that got lost.
- The `parameters` output file from the `run_setup` command incorrectly reported the length and angle of `CSBEND` elements when element division was invoked. This was reported by V. Sajaev (APS).
- The `amplification_factors` command now respects `link_elements` commands.
- The `tune_footprint` command now optionally runs in major action command mode. The inability to do so was pointed out by Y.-P. Sun (APS).

13.25.3 New and Modified Elements

- The long-deprecated DIRECTION parameter of the SCRAPER element has been removed; input files using the SCRAPER element will need to be updated to remove this parameter and replace it with equivalent INSERT_FROM parameter. One result is that the SCRAPER element can now support two-sided scrapers.
- Added the SYSTEMATIC_MULTIPOLE_FACTOR, RANDOM_MULTIPOLE_FACTOR, and STEERING_MULTIPOLE_FACTOR parameters to the KQUAD, KSEXT, and KOCT elements. These allow multiplying each of the indicated higher multipole contributions by a factor.
- Added YAW and YAW_END parameters to UKICKMAP element. It's useful in simulating canted insertion devices.
- Added the SPEEDBUMP element, which provides a new kind of aperture formed by a semi-circular bump protruding from one or both sides of the chamber.
- Added the DX, DY, and DZ misalignment parameters to the EHKICK, EVKICK, and EKICK elements. Also added RANDOM_MULTIPOLES parameter.

13.25.4 New and Modified Commands

- Added the inelastic_scattering command, which assists in computation of the inelastic gas scattering lifetime and the distribution of lost particles. This is only available in the parallel version.
- Added the generation_interval parameter to the ion_effects command to permit generation of ions only at every nth bunch. This was suggested by J. Calvey (APS).
- Added the ignore_elements command, which allows instructing elegant to ignore specified elements in tracking. This can reduce overhead from "do-nothing" elements like markers and monitors.
- The link_elements command can now create the source element name by editing the target name.
- The momentum_aperture command now uses resources more efficiently for the parallel version when output_mode=2. In particular, it honors the user-provided minimum δ values. In addition, the domain decomposition was revised to better equalize the workload of the processors.

13.25.5 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to check results against the serial or parallel versions and report issues to the developers.

- None.

13.25.6 Changes to Related Programs and Files

- Added the inelasticScatteringAnalysis script, a companion to the inelastic_scattering command in Pelegant. It allows computing the lifetime and local loss rates from inelastic gas scattering.

13.26 Highlights of What's New in Version 33.1.1, 25 July 2017

Here is a summary of what's changed since release 33.0.

13.26.1 Bug Fixes for Elements

- The BGGEXP element had a bug that prevented it from working when two elements used the same data file. This was fixed.
- The BGGEXP element refused to run if $m = 1$ (dipole) was the main multipole, which prevented modeling wigglers. This was reported by forum user Ji_Li and was fixed.
- The RFDF element had a bug in computing the energy-dependence of the time of flight, as reported by Daniel Marx. This was fixed. The missing phase reference feature was also implemented.
- Using the third-order matrix of the QUAD element with RADIAL=1 would result in a crash. This was fixed. Forum user meisal reported the bug.

13.26.2 Bug Fixes for Commands

- Fixed a bug in load_parameters related to the allow_missing_elements and allow_missing_parameters qualifiers. In runs with multiple load_parameters commands, only the last values of these parameters were used.
- Fixed a bug in saving parameters when elements are subdivided: the lengths of certain elements were incorrect in the saved file.

13.26.3 New and Modified Elements

- The IONEFFECTS element and the companion ion_effects command were added. These allow simulation of the interaction of the beam with residual gas ions. J. Calvey (ANL) did much of the work on these new features.
- Added SLICE element to provide turn-by-turn slice analysis.
- The CSBEND element now includes skew multipole errors up to eighth order. This involves newly-computed expressions for the fields in curvilinear coordinates, so slight numerical changes may be seen.
- The KSEXT and SEXT elements now support a skew-quad correction term. This was suggested by Z. Duan (IHEP).
- Synchrotron radiation effects were added to the BGGEXP element, so that radiation effects from essentially arbitrary fields can be included in both tracking and moments_output calculations. There are limitations as described in the manual page.
- Improvements were made to memory management for numerous elements, chiefly CSBEND, CSRCSBEND, CWIGGLER, FRFMODE, FTRFMODE, RFMODE, SLICE, TFBDRIVER, TRFMODE, ZTRANSVERSE, and ZLONGIT. This can dramatically decrease memory usage in some cases.
- The TFBPICKUP and TFBDRIVER elements (used for turn-by-turn feedback) now have start- and end-pass controls.

- The MATTER element now has start- and end-pass controls.
- To improve performance and simplify the code, the SQRT_ORDER parameter on the CSBEND, FMULT, KOCT, KQUAD, KQUSE, and KSEXT elements is now nonfunctional. The default behavior (exact square roots) is unchanged.
- The BMXYZ element now has the option for classical synchrotron radiation. It can also check the divergence and curl of the fields to assess the quality of the field solution.
- Added the BX and BY parameters to the BGGEXP element, to allow imposing a uniform "external" magnetic field.
- It is now possible to interleave zero-length LSCDRIFT elements with CSRCSBEND elements with CSR fields building up through the successive CSRCSBEND elements. This was added following a related forum post by Aaron Fetterman.

13.26.4 New and Modified Commands

- Added the elastic_scattering command, which assists in computation of the elastic gas scattering lifetime and the distribution of lost particles. This is only available in the parallel version.
- Added bpm_output option to the correct command, which provides optional output of beam position monitor readings after orbit or trajectory correction. This was suggested by V. Sajaev (APS).
- The twiss_output command now records the location of the acceptance-limiting apertures in parameters AxLocation and AyLocation.
- The track command has a new field, interrupt_file, which gives the name of a file to monitor as a semaphore to interrupt the tracking. If the file is created or updated during tracking, then tracking will terminate on completion of the next pass.

13.26.5 Changes Specific to Parallel Version

- The elastic_scattering command was added. It performs parallel tracking to determine the angular acceptance at a series of s locations. The data is intended for use with the script elasticScatteringAnalysis, which allows determination of the elastic gas scattering lifetime and loss distribution. This command is presently only available in Pelegant, due to the long runtime required.

13.26.6 Changes Specific to the GPU Version

The GPU version continues to be an alpha release and contains bugs. Users are encouraged to test results against the serial or parallel versions.

- None.

13.26.7 Changes to Related Programs and Files

- The `computeGeneralizedGradients` script (used to prepare data for the `BGGEXP` element) did not work for odd multipole orders (e.g., dipole, sextupole, ...) or fields that are odd functions of z. This was reported by forum user Ji_Li and has been fixed, with the assistance of R. Lindberg (APS).
- The program `sddsmatchmoments` was added. It allows generating a particle distribution to match the moments from the `moments_output` command.
- The `LFBFirSetup` script was added. It helps set up FIR filters for longitudinal turn-by-turn feedback.
- `touschekLifetime` can now use data from the `SLICE` element in `elegant` for slice-based lifetime computations.
- The script `removeBackDrifts` was added. It allows post-processing s-dependent files to remove negative drifts, which improves the appearance of plots and is needed for certain types of analysis.
- The program `sddsemitproc` now has the ability to specify the independent variable on the commandline. This was suggested by forum user `jan`.
- The `TFBFirSetup` script, which helps set up FIR filters for transverse turn-by-turn feedback, can now support filters with up to 30 terms.

13.27 Highlights of What's New in Version 33.0, March 3, 2017

Here is a summary of what's changed since release 32.0. Historical change logs are appended to the end of this manual.

This version includes an alpha release of GPU-enabled code. The original GPU code was developed by Tech-X corporation [51], with further work by R. Soliday (APS).

13.27.1 Bug Fixes for Elements

- The `SREFFECTS` element now correctly computes the equilibrium horizontal and vertical emittances when $J_x \neq 1$. Previously, the computation used an equation that implicitly assumes $J_x = 1$.
- The `MALIGN` element could cause spurious integer changes in the reported tunes if the `DZ` parameter was negative. This problem, reported by V. Sajaev (APS), was fixed.
- A memory management bug related to the systematic and random multipole data store was fixed. This in principle affected `KQUAD`, `KSEXT`, and other elements using the `SYSTEMATIC_MULTIPOLES` and `RANDOM_MULTIPOLES` features. In testing, no effect was in fact observed.

13.27.2 Bug Fixes for Commands

- The `correction_matrix_output` command command were ignoring the monitor calibrations (`MONI`, `HMON`, and `VMON`) values when `use_response_from_computed_orbits = 1`. This was reported by V. Sajaev (APS).

- The steering_element command no longer aborts even if the declared steering corrector appears not to kick the beam. This allows using unusual controls such as path length to steer the beam. This issue was pointed out by V. Sajaev (APS).
- The load_parameters and save_lattice commands incorrectly saved the edge angles and other edge-related quantities for bending magnets that were reflected. This issue was fixed. *Previously-saved parameter files should be modified (e.g., remove the edge parameters)* unless the magnets had the same parameters for the entrance and exit. This problem was reported by Y. Li (BNL).
- The rf_setup and moments_output commands will now run in a loop with find_aperture, momentum_aperture, and frequency_map operations, if set for per-step execution. Previously, this would only happen for the track, analyze_map, and touschek_scatter commands.

13.27.3 New and Modified Elements

- The EKICK, EHKICK, and EVKICK elements now support inclusion of multipole errors linked to the correction strength.
- The steering kicks and steering multipoles in the KQUAD element are now implemented in the body of the element, rather than at the ends.
- The WATCH element was improved so that the dt column in coordinate-logging mode and the dCt column in parameter- and centroid-logging modes are more useful. In particular, in normal cases these will now more reliably be centered on zero. One can also provide a reference frequency relative to which the reference time is defined. This improvement grew out of discussions with J. Calvey and T. Berenc (APS).
- The reported phases of the beam- and generator-induced parts of the voltage for the RFMODE element RECORD file are now computed using a method that should be more reliable. This improvement grew out of discussions with J. Calvey and T. Berenc (APS).
- The RECORD output from the RFMODE element now includes the phase of the net cavity voltage. This was requested by M. Venturini (LBNL).
- The RFMODE element now supports injection of noise into the rf source and low-level rf system. This is based on discussions with T. Berenc (APS).
- The SCRIPT element can now import particleID data from the script without attempting to use this information for lost-particle accounting. This provides better functionality when the particleID is used for other purposes, such as bunch membership.
- The TFBPICKUP and TFBDRIVER elements, used for bunch-by-bunch feedback, now allow 30-term FIR filters, up from 15 turns in earlier versions.
- The TFBDRIVER element now accepts specification of the frequency and phase of the driver cavity.
- Aperture enforcement inside KQUAD, KSEXT, KOCT, KQUSE, CSBEND, and CSRCSBEND elements has been improved. In particular, the ELLIPTICAL, EXPONENT, YEXPONENT, and OPEN_SIDE parameters of MAXAMP are now implemented. In addition, for the fourth-order integrator, the

apertures are no longer asserted at each integration step, but only after each slice (or "kick", to use the misleading terminology of the element parameters).

- Added the ALLOW_LONG_BEAM parameter to the ZLONGIT and ZTRANSVERSE elements.

13.27.4 New and Modified Commands

- The bunched_beam command can now be set to take the fully-coupled 6D bunch parameters from the calculations of the moments_output command, provided the latter is used to compute matched, equilibrium parameters. This was requested by forum user duanz.
- Added occurrence and positional filters for the steering_element command. This was requested by V. Sajaev (ANL).
- Several informational printouts for the touschekScatter command are no longer shown by default, but only if the verbosity control is set to a non-zero value. This makes short runs more efficient.
- Compared to previous versions, the lost-particle data file (losses file requested by the run_setup command) will exhibit changes in the order in which particles are recorded. This was a result of reworking the code for lost particle management.

13.27.5 Changes Specific to Parallel Version

- None.

13.27.6 Changes to Related Programs and Files

- The program madto was renamed elegantto, to more accurately reflect what it does. It will now translate elegant lattice files into MAD8 format.

13.28 Highlights of What's New in Version 32.0, 5 Jan. 2017

Here is a summary of what's changed since release 31.

13.28.1 Bug Fixes for Elements

- None.

13.28.2 Bug Fixes for Commands

- A bug was fixed in the amplification_factors command that resulted in a crash when the corrected amplification factors were requested. This was reported by S. DiMitri (ELETTRA).
- A bug was fixed for twiss_output, which was incorrectly reporting the quantities $\frac{\partial\alpha_{x,y}}{\partial\delta}$ (parameters dalphax/dp and dalphay/dp in the output file) in some cases.

13.28.3 New and Modified Elements

- Added the BRANCH element, which permits branching between parts of a beamline based on the number of passes executed.
- Apertures specified using MAXAMP or an external aperture file (using the aperture_data command) are now enforced inside CSBEND and CSRCSBEND elements. There may be small changes in, for example, momentum acceptance as a result of this, particularly when gradient dipoles are involved.
- The longitudinal location of losses inside KQUAD and KSEXT elements is now computed more accurately. Previously, it was simply the start of the element.
- Removed the non-functional FRINGE parameter of the CSBEND element.
- The BGGEXP (B-field Generalized Gradient Expansion) element now supports symplectic integration using an implicit method, implemented by R. Lindberg (APS).

13.28.4 New and Modified Commands

- Added exclude parameter to chromaticity command, allowing exclusion of some sextupoles that may match the list in the sextupole parameter.
- Added alter_at_each_step and alter_before_load_parameters parameters to the alter_elements command, allowing better control of potential conflicts with load_parameters.
- The random number generator seed is now permuted bitwise in order to add a greater level of apparent randomness. Thus, changing the seed by a small amount will now have a bigger effect on the sequences generated, making it easier to deliberately perform several runs with very distinct random values. This can be defeated using the global_settings command by setting inhibit_seed_permutation=1. This issue was pointed out by V. Sajaev (APS).

13.28.5 Changes Specific to Parallel Version

- None.

13.28.6 Changes to Related Programs and Files

- ionTrapping — Added computation of the single-ion oscillation frequency.

13.29 Highlights of What's New in Version 31.0, 1 Oct. 2016

Here is a summary of what's changed since release 30.1.

13.29.1 Bug Fixes for Elements

- The touschek_scatter command had a bug when random multipoles where used on KQUAD and KSEXT elements. In particular, these multipoles components were re-randomized for each TSCATTER element. This was discovered and fixed by A. Xiao (ANL).

- The implementation of edge effects in the KQUAD element was using x' and y' in place of q_x and q_y, and so was not symplectic. It also did not have the correct dependence on δ. These issues were reported by R. Lindberg (ANL). A similar error was fixed in the implementation of edge effects for CSBEND; this was fixed by Y.P. Sun (ANL). Practically speaking, we haven't noticed any significant change in results.
- There was a bug in the evaluation of systematic multipoles when using the second-order integrator for KQUAD and KSEXT. The default fourth-order integrator did not have this issue.
- Higher-order path-length issues were fixed for the BRAT element. This issue was reported by R. Lindberg (ANL).
- The steering kick calibration factors are no longer ignored on the KQUAD element.
- The BMXYZ and BMAPXY elements lacked dependence on the momentum deviation δ. This issue was reported by R. Lindberg (ANL).

13.29.2 Bug Fixes for Commands

- None

13.29.3 New and Modified Elements

- Added the BGGEXP element, which performs tracking through magnetic fields constructed from a generalized gradient expansion [50]. Although the integration is not symplectic, the fields satisfy Maxwell's equations exactly. A script, computeGeneralizedGradients, is provided to assist in preparing input for this element. Advice from M. Venturini (LBNL) was helpful in performing this work.
- Added separate specification of edge and body multipoles to the KQUAD and KSEXT elements.
- Added steering and steering multipoles to the KSEXT element.
- The BMXYZ element now allows independent specification of the insertion length and field map length.
- The code for the KQUAD, KSEXT, MULT, and FMULT was improved to prevent underflows that might occur in some odd cases, which would negatively affect accuracy.
- The LSRMDLR element now includes an option for a helical device. This was requested by forum user zzhang and implemented by Y.-P. Sun (ANL).
- Two additional parameters, SampledParticles and SampledCharge were added to WATCH files in coordinate mode. These are identical to Particles and Charge, respectively, except when the FRACTION parameter is < 1. In that case, the latter parameters give the values prior to sampling, while the new parameters give the parameters of the sampled fraction of the bunch. Previously, Particles and Charge changed as FRACTION was changed. *Note that scripts that use the* Particles *and* Charge *may need modification since the meaning has changed.* Y. Ding (SLAC) pointed out this issue.

13.29.4 New and Modified Commands

- The analyze_map command can now report the map using canonical variables. It also has a user-controlled accuracy parameter that can be used to eliminate spurious matrix elements. R. Lindberg (ANL) helped with the development and testing.
- The touschek_scatter command now uses averaging of the loss rate over the interval between two TSCATTER elements instead of the local value at the element, which gives more accurate estimates of the distribution of scattered particles. This change requires that TSCATTER elements be inserted at the beginning and end of the beamline, which can be done using add_at_end=1 and add_at_start=1 in the insert_elements command. This was implemented by A. Xiao (ANL).
- The modulate_elements command now offers more control over verbose printouts, to help reduce the volume of uninformative printouts. It also provides user control of the buffer flushing interval for the record output file.
- The insert_elements command now has the option to insert an element at the beginning of the beamline.

13.29.5 Changes Specific to Parallel Version

- None.

13.29.6 Changes to Related Programs and Files

- The script computeGeneralizedGradients was added to assist in preparing input for the BGGEXP element.
- The scripts elasticScatteringLifetime and bremsstrahlungLifetime now support user-specified gas composition. The Z values for carbon and oxygen were mixed up in some places in these and related scripts, as pointed out by S. Tian (IHEP); this was fixed.
- The ionTrapping script now supports user-provided factors for inflating the emittance and energy spread.

13.30 Highlights of What's New in Version 30.1, 3 Aug. 2016

Here is a summary of what's changed since release 30.0

13.30.1 Bug Fixes for Elements

- Fixed a bug in Touschek scattering simulation (TSCATTER element and touschek_scatter command) that resulted in the random multipole components of KQUAD and KSEXT elements being re-randomized for each TSCATTER element.

13.30.2 Bug Fixes for Commands

- Fixed a bug introduced in moments_output computations when CSBEND elements were present with non-zero values of ETILT. Reported by V. Sajaev (ANL).

- Fixed a bug in Touschek scattering simulation (TSCATTER element and touschek_scatter command) that resulted in the random multipole components of KQUAD and KSEXT elements being re-randomized for each TSCATTER element.

13.30.3 New and Modified Elements

- Added edge multipoles to KQUAD element. This necessitated some rearrangement of the code, so results might be slightly different even if this feature is not invoked.
- Added I/Q mode feedback to the RFMODE element.

13.30.4 New and Modified Commands

- None.

13.30.5 Changes Specific to Parallel Version

- Implemented exact normalized emittance calculations for the sigma output file of the run_setup command and in WATCH output in parameter mode. J. Bjorklund pointed out the lack of calculations in the parallel version.
- Fixed bug in assignment of particle ID values when using Halton sequences in the bunched_beam command.

13.30.6 Changes to Related Programs and Files

- The program abrat ("Asymmetric Bend RAy tracing") was added. It allows tracking electrons through 2- and 3-D magnetic field maps. It is a commandline version of the BRAT element.
- The script ionTrapping was added, providing simple ion trapping calculations for uniform bunch trains. J. Calvey (APS) helped with debugging.
- The script computeSCTuneSpread was added to allow computation of space-charge tune spread.
- The script radiationEnvelope now computes envelopes for central cone flux.

13.31 Highlights of What's New in Version 30.0, 5 July 2016

Here is a summary of what's changed since release 29.1:

13.31.1 Bug Fixes for Elements

- Fixed a memory leak in the FTABLE element.

13.31.2 Bug Fixes for Commands

- Fixed calculations of exact normalized emittance (error in equations) and implemented in parallel version. This bug impacted results in the `sigma` output file of the `run_setup` command and in `WATCH` output in `parameter` mode. J. Bjorklund pointed out the lack of calculations in the parallel version and provided an example run that helped discover the problem with the serial version.
- The `diffusionRate` output from the `frequency_map` command is now computed as $\log_{10}((\Delta\nu_x^2 + \Delta\nu_y^2)/n)$ instead of $(\log_{10}(\Delta\nu_x^2 + \Delta\nu_y^2))/n$.
- Fixed a bug in `bunched_beam` whereby the centroids for a shell-type beam were offset from zero. Reported by L. Emery (ANL).
- Fixed bug in `moments_output` when bending magnts with non-zero `ETILT` are present. When this occurs, the number of slices for moments calculation is set to 1 for those elements, to avoid numerical problems with the vertical orbit.

13.31.3 New and Modified Elements

- Added the `LEFFECTIVE` parameter for `QUAD` and `KQUAD`, which provides a convenient way to change the effective length without changing the adjacent drift spaces. Also added the ability to turn off the linear fringe field effects while keeping the nonlinear part, and to multiply the nonlinear effects by a numerical factor.
- Added the `BMXYZ` element for straightforward integration through 3D field maps for straight elements.
- Added the `BRAT` element, which is similar to `BMXYZ` but accommodates curved elements. Elements may be asymmetric, e.g., longitudinal gradient dipoles.
- Added the `FACTOR` and `THRESHOLD` options to `FTABLE`. The former allows multiplying the fields by a user-defined factor. The latter allows specifying the magnitude of the field below which it is considered zero, which can help ensure numerical stability.
- The `FTABLE` element can accept the simple-to-create input files used by the `BMXYZ` element in addition to the original input format.
- Results that depend on the transport matrix will show small changes for elements for which the matrix is determined by tracking. The tracking-based method was modified to use a larger number of sample points, increasing the accuracy.

13.31.4 New and Modified Commands

- Added the `full_grid_output` parameter to the `frequency_map` command, making it possible to display frequency maps using `sddscontour`.

13.31.5 Changes Specific to Parallel Version

- Implemented exact normalized emittance calculations for the `sigma` output file of the `run_setup` command and in `WATCH` output in `parameter` mode. J. Bjorklund pointed out the lack of calculations in the parallel version.

- Fixed bug in assignment of particle ID values when using Halton sequences in the `bunched_beam` command.

13.31.6 Changes to Related Programs and Files

- The program `abrat` ("Asymmetric Bend RAy tracing") was added. It allows tracking electrons through 2- and 3-D magnetic field maps. It is a commandline version of the `BRAT` element.
- The script `ionTrapping` was added, providing simple ion trapping calculations for uniform bunch trains. J. Calvey (APS) helped with debugging.
- The script `computeSCTuneSpread` was added to allow computation of space-charge tune spread.
- The script `radiationEnvelope` now computes envelopes for central cone flux.

13.32 Highlights of What's New in Version 29.1, 3 March 2016

Here is a summary of what's changed since release 29.0:

13.32.1 Bug Fixes for Elements

- Fixed bugs in `RECORD` output from `TRFMODE` element for multi-step, single-pass runs. This was fixed by A. Xiao (APS).

13.32.2 Bug Fixes for Commands

- The `replace_elements` command now respects quoted sequences in the new element definition.

13.32.3 New and Modified Elements

- `LRWAKE` now supports long-range quadrupole wakes. R. Lindberg (APS) provided helpful discussion in this implementation.
- `ILMATRIX` now supports second-order tune shift with amplitude as well as path-length dependence on amplitude.
- `TFBPICKUP` now supports horizontal and vertical offsets.
- Added logging of photon coordinates and angles to the `CSBEND` element. Works in serial mode only.
- `TRFMODE` now supports interpolation within bins, giving smoother results.

13.32.4 New and Modified Commands

- `alter_elements` now has a occurrence-skip parameter, which would allow for example changing every other member of a group of elements.
- `momentum_aperture` now allows specifying that `WATCH` elements remain active during momentum aperture determination.

- frequency_map was modified to include the path-length in the output file, which can be used to determine the dependence of the path length of the betatron amplitude.

13.32.5 Changes Specific to Parallel Version

- None.

13.32.6 Changes to Related Programs and Files

- The script prepareTAPAs was added, which allows processing files from twiss_output into a form that is accepted by the Android App TAPAs [46].
- The script makeSummedCsrWake was added, which allows making a CSR wake that sums up contributions from dipoles with various lengths and bending radii.
- The script TFBFirSetup was added, which allows generating FIR filters for turn-by-turn feedback using TFBDRIVER and TFBPICKUP elements.
- ibsEmittance can now perform intrabeam scattering calculations for non-gaussian longitudinal distributions.
- computeCoherentFraction now uses $\lambda/4\pi$ for the radiation emittance to be consistent with sddsbrightness.
- longitCalcs now computes the bucket-half-height even when a harmonic cavity is powered.

13.33 Highlights of What's New in Version 29.0, 15 Jan. 2016

Here is a summary of what's changed since release 28.1:

13.33.1 Bug Fixes for Elements

- Fixed a bug in the MATR element that would crop up in multi-step runs, causing a crash or lock-up. This was reported by P. Emma (SLAC).
- Fixed a bug in the RFMODE element that resulted in a few percent error between the voltage seen by the beam and the feedback-regulated voltage. T. Berenc (ANL) helped resolve this.
- The output file feature was restored for the FTRFMODE element.
- The TFBDRIVER and TFBPICKUP feedback elements can now handle changes in the number of bunches.
- The drive limit for TFBDRIVER is now imposed after application of the filter, rather than before.
- The KQUAD element now has a valid associated transfer matrix for RADIAL=1. This bug was reported by forum user libov.

13.33.2 Bug Fixes for Commands

- The `touschek_scatter` command now behaves as a regular major action command, meaning that error generation, scanning, parameter loading, etc. behave as expected.
- Fixed a bug in the `correct_tunes` command that resulted in a crash when `n_iterations=0` and would also have resulted in invalid data in the log file for mixed element types. This was reported by V. Sajaev (ANL).
- Fixed a bug in the `chromaticity` command that resulted in a crash when `n_iterations=0` and would also have resulted in invalid data in the log file for mixed element types.
- Fixed a bug related to optimization of the chromatic derivative of `alpha_x`. The value provided was actually the chromatic derivative of `betax`. A related error gave incorrect results for the `use_linear_chromatic_matrix` mode of the `track` command.
- Previous versions of this manual indicated that the `find_aperture` command provided a quantity `Area` giving the dynamic aperture area for optimization. The quantity is in fact called `DaArea`. This was reported by S. Hilbrich (TU Dortmund).
- Fixed a bug in the optimization feature that resulted in the user's weighting factors being ignored. This was pointed out by A. Zholents (ANL).
- Fixed a bug in the `alter_elements` command that caused string values not to be reflected in the output file created with `save_lattice`. This was reported by T. Pulampong (SLRI/DLS).

13.33.3 New and Modified Elements

- Added nonlinear symplectic fringe field model to CSBEND and CSRCSBEND, based on theoretical work of K. Hwang (IU) [45]. The implementation was performed by Y. Sun (APS) with assistance from K. Hwang and M. Borland.
- Added `EKICKER`, `EHKICK`, and `EVKICK`, which provide various flavors of steering correctors using an Exact model. These may be used in place of the existing `KICKER`, `HKICK`, and `VKICK` elements. The need for this was pointed out by L. Yang (BNL).
- The `MATTER` element now supports arrays of slits. This can be used, for example, to model a double-slit spoiler for producing two pulses in an FEL.
- The `ECOL` and `RCOL` collimator elements now support an `INVERT` parameter to allow simulation of an obstruction instead of an opening.
- The output files from the `WATCH` element in centroid and parameter mode now contain the beam charge, provided that a `CHARGE` element is in the beamline.
- Elements that read multipole error files (e.g., `KQUAD` and `KSEXT`) now share data internally rather than each reading the data files separately. This provides a significant speed improvement for massively parallel execution in particular.
- The `MALIGN` element was improved to allow optionally applying misalignments to only part of the beam, based on the particle ID.
- The `RFMODE` element now has a feature that allows "muting" the rf generator on a specified pass, to simulate a trip of the rf source.

- The voltage "preloading" feature of the `RFMODE` element now works even when rf feedback is used.

- In order to eliminate problems with the parallel version, the `IBSCATTER` element no longer has a separate `CHARGE` parameter. Instead, the `CHARGE` element should be used.

13.33.4 New and Modified Commands

- The `analyze_map` command can now determine the nonlinear transport matrix up to third order based on tracking data, using the method described in [4]. Parallel tracking is used for this command in `Pelegant`. Previously, the analysis was limited to the linear matrix. Also, the terminal lattice functions and their chromatic derivatives are determined from the map for both transport lines and rings. This was requested by Y. Hao (BNL) and L. Yang (BNL).

- The `correct_tunes` and `chromaticity` commands now include a weighting factor that results in minimization of the strength changes in the event that more than two familes are provided for correction. (In the future this will be replaced with an SVD-based implemenetation.)

- Added to `closed_orbit` and `correct` commands the ability to use multi-turn tracking to determine the approximate orbit. This was suggested by V. Sajaev (ANL), and is helpful when the orbit convergence is poor.

- The output in the `run_setup centroid` file now contains the beam charge, provided that a `CHARGE` element is in the beamline.

- The `run_control` command now includes a variable, `n_passes_fiducial`, that allows specifying a different number of tracking passes for fiducialization than for tracking. For ring fiducialization, this should probably always be 1.

- Most output files from `elegant` now include a parameter giving the SVN revision number of the version used to create the output.

13.33.5 Changes Specific to Parallel Version

- The `analyze_map` command, which was improved as described above, can now use parallel resources.

- A bug was fixed in the `center_on_orbit` feature of the `track` command. The bug caused the particles on each processor to be offset by different amounts related to the centroid of the local particles only. This was reported by M. Furseman (DLS).

- Fixed a bug in `FTABLE` introduced in version 26.0. The bug would cause the program to crash.

- Memory management was improved in the `touschek_scatter` command, allowing a larger number of particles to be utilized.

- The `SCRIPT` element would cause a crash when `twiss_output`, `matrix_output`, or similar commands were included but when tracking was required to determine the transfer matrix of the element. This was fixed.

- Tracking instigated via the `track` command is now more forgiving of uneven particle losses among cores. In particular, the program should no longer crash if one core has lost all of its particles or all of the particles in a particular bunch.

- The stop_tracking_particle_limit feature of the track command now works in the parallel version.
- Instead of exiting, the parallel version now simply ignores the slice_anlysis command.

13.33.6 Changes to Related Programs and Files

- The script reorganizeMmap was added to convert momentum aperture data from Pelegant in output_mode=1 into the same form as produced by elegant. This was a result of correspondence with S. Tian (IHEP).
- A bug was fixed in elegant2astra that resulting in slightly erroneous values for the longitudinal coordinate.
- The beamLifetimeCalc can now perform approximate Touschek lifetime calculations for polarized beams. This was added by A. Xiao (ANL) following an inquiry from forum user marlibgin.

13.34 Highlights of What's New in Version 28.1.0, 23 July 2015

Here is a summary of what's changed since release 28.0:

13.34.1 Bug Fixes for Elements

- The ROTATE element was not affecting the floor coordinates. This was found and fixed by A. Xiao (APS).
- The END_PASS parameter on SCATTER now works as expect, after removal of a one-pass offset.

13.34.2 Bug Fixes for Commands

- A bug was fixed that caused a crash when a 1-line aperture search was performed. This was reported by Guohui Wei (JLab).

13.34.3 New and Modified Elements

- The TFBDRIVER element now has the ability to measure the beam phase for use in longitudinal feedback. Previously, only momentum-based input was available for longitudinal feedback.

13.34.4 New and Modified Commands

- The ramp_elements and modulate_elements commands now have the ability to write a record of their output values.
- The run_setup command now has options, intended primarily for developers, to turn on memory usage and executing time monitoring during tracking.
- The units given for loss rate the output files from touschek_scatter were incorrect and were fixed. Results were not affected. (A. Xiao, ANL)

- The tune_footprint command was improved in several ways. It is now possible to ignore half-integer resonances. The upper and lower bounds of the chromatic tune footprints are now available for optimization. It's now possible to turn off either chromatic or amplitude tune footprint deterimination.
- The optimization_setup command allows suppressing particle tracking in order to improve performance in some unusual cases.
- The correct_tunes command can now utilize any element that has the K1 parameter.
- The chromaticity command can now utilize any element that has the K2 parameter.

13.34.5 Changes for Parallel Version Only

- Fixed a bug that affected tracking when orbit correction was used, start_from_centroid=1, and particle distribution was not random across processors.
- Warnings about $\rho > 10^6$ m are now issued by the parallel version, as for the serial version.
- Memory usage logging to WATCH output files now sums the memory across all cores, rather than just the master core.
- A memory leak was fixed in the ZTRANSVERSE element that sometimes caused the program to crash. This was reported by R. Lindberg (ANL).
- The output of the beam charge in the ZLONGIT wake output file was corrected; previously, it only showed the charge on one core.
- The frequency_map command now provides an estimate of the time needed to complete.

13.34.6 Changes to Related Programs and Files

- The program sddsbunchingfactor is now part of the distribution.

13.35 Highlights of What's New in Version 28.0.0, 18 June 2015

Here is a summary of what's changed since release 27.1.0:

13.35.1 Bug Fixes for Elements

- The WATCH element was improved so that the dCt column (in parameter or coordinate mode) and dt column (in coordinate mode) no longer exhibit fictitious drift due to precision limitations in simulations of rings with many turns.
- For numerical reasons, any CSBEND with $\rho > 10^6$ m is replaced with another element. In the past, an EDRIFT was used, which would produce incorrect results if the element had non-zero K_1 or K_2. This was fixed.

13.35.2 Bug Fixes for Commands

- None.

13.35.3 New and Modified Elements

- The TFBPICKUP and TFBDRIVER elements, which provide a turn-by-turn feedback capability, now support multi-bunch feedback. In addition, support was added for longitudinal feedback as well as sample/update intervals greater than one turn.
- The CSRDRIFT element can now also include longitudinal space charge, using the algorithm from the LSCDRIFT element.
- The CSBEND element has a new feature that allows suppression of spurious trajectory offsets that result from limitations of the symplectic integration routine. This feature is controlled using the REFERENCE_CORRECTION parameter.
- The input of multipole errors for KQUAD and KSEXT elements was modified so that the input columns have more transparent names. Previously, the names caused some confusion. Files that worked with previous versions are still accepted.
- The MARK element with FITPOINT=1 now stores the emittances of the three modes as e1m, e2m, and e3m for optimization if moments_output is invoked. This deficiency was pointed out by forum user marlibgin.

13.35.4 New and Modified Commands

- The transmute_elements command now does a better job of copying common parameters between the old and new element types. In the past, only the length was preserved. A. Zholents (ANL) reported this issue.
- The floor_coordinates command has a new parameter, store_vertices, which allows requesting that dipole vertex points be stored for use in optimization.
- The twiss_output command now stores the acceptances Ax and Ay for use in optimization.

13.35.5 Changes for Parallel Version Only

- None

13.35.6 Changes to Related Programs and Files

- None.

References

[1] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, N.J., second edition, 1988.

[2] H. Grote, F. C. Iselin, "The MAD Program–Version 8.1," CERN/SL/90-13(AP), June 1991.

[3] K. L. Brown, R. V. Servranckx, "First- and Second-Order Charged Particle Optics," SLAC-PUB-3381, July 1984.

[4] M. Borland, "A High-Brightness Thermionic Microwave Electron Gun," SLAC-Report-402, February 1991, Stanford University Ph.D. Thesis.

[5] H. A. Enge, "Achromatic Mirror for Ion Beams," Rev. Sci. Inst., 34(4), 1963.

[6] M. Borland, private communication.

[7] W. H. Press, *et al*, *Numerical Recipes in C*, Cambridge University Press, Cambridge, 1988.

[8] M. Borland, "A Self-Describing File Protocol for Simulation Integration and Shared Postprocessors," Proc. 1995 PAC, May 1-5, 1995, Dallas, Texas, pp. 2184-2186 (1996).

[9] M. Borland, "A Universal Postprocessing Toolkit for Accelerator Simulation and Data Analysis," Proc. 1998 ICAP Conference, Sept. 14-18, 1998, Monterey, California, to be published.

[10] T. P. Green, "Research Toward a Heterogeneous Networked Computer Cluster: The Distributed Queuing System Version 3.0," SCRI Technical Publication, 1994.

[11] M. Borland *et al*, "Start-to-End Jitter Simulation of the LCLS," Proceedings of the 2001 Particle Accelerator Conference, Chicago, 2001.

[12] M. Borland and L. Emery, "Tracking Studies of Top-Up Safety for the Advanced Photon Source,", Proceedings of the 1999 Particle Accelerator Conference, New York, 1999, pg 2319-2321.

[13] M. Xie, "Free Electron Laser Driven by SLAC LINAC".

[14] S. Reiche, *NIM* A 429 (1999) 242.

[15] J.D. Bjorken, S.K. Mtingwa, "Intrabeam Scattering," Part. Acc. Vol. 13, 1983, 115-143.

[16] K. Halbach, "First Order Perturbation Effects in Iron-Dominated Two-Dimensional Symmetrial Multipoles", NIM **74-1**, 1969, 147-164.

[17] J. D. Jackson, *Clasical Electrodynamics*, second edition.

[18] G. Ripken, DESY Report No. R1-70/04, 1970 (unpublished).

[19] *Handbook of Accelerator Physics and Engineering*, A. Chao and M. Tigner eds., 1998.

[20] Ya. S. Derbenev, J. Rossbach, E. L. Saldin, V. D. Shiltsev, "Microbunch Radiative Tail-Head Interaction," September 1995, TESLA-FEL 95-05.

[21] A. Xiao *et al.*, "Direct Space-Charge Calculation in `elegant` and its Application to the ILC Damping Ring," Proc. PAC2007, 3456-3458.

[22] Z. Huang *et al.*, Phys. Rev. ST Accel. Beams **7** 074401 (2004).

[23] A. Piwinski, " The Touschek effect in strong focusing storage rings," DESY-98-179, Nov 1998.

[24] A. Xiao *et al.*, "Touschek Effect Calculation and its Application to a Transport Line," Proc. PAC07, 3453-3455 (2007).

[25] W. Warnock, "Shielded Coherent Synchrotron Radiation and Its Effect on Very Short Bunches," SLAC-PUB-5375, 1990.

[26] T. Agoh and K. Yokoya, "Calculation of coherent synchrotron radiation using mesh," Phys. Rev. ST Accel. Beams 7, 054403 (2004).

[27] P. Elleaume, "A New Approach to Electron Beam Dynamics in Undulators and Wigglers," Proc. EPAC 1992, 661-663.

[28] http://www.esrf.eu/Accelerators/Groups/InsertionDevices/Software/Radia

[29] J. Bengtsson, "The Sextupole Scheme for the Swiss Light Source (SLS): An Analytic Approach," SLS Note 9/97, March 7, 1997. (Corrections to several typos were supplied by W. Guo, NSLS.)

[30] K. Flöttmann, Astra User Manual, http://www.desy.de/ mpyflo/Astra_dokumentation/

[31] J. Qiang *et al.*, J. Comp. Phys. 163, 434 (2000).

[32] V. N. Aseev *et al.*, Proc. PAC05, 2053-2055 (2005); ASCII version 39 from B. Mustapha.

[33] H. Chi *et al.*, Mathematics and Computers in Simulation **70** (2005) 9-21.

[34] D. Zhou *et al.*, "Explicit maps for the fringe field of a quadrupole," Proc. IPAC10.

[35] J. Irwin *et al.*, "Explicit soft fringe maps of a quadrupole," Proc. PAC95.

[36] C. X. Wang, "Explicit Formulas for 2nd-order Driving Terms due to Sextupoles and Chromatic Effects of Quadrupoles," ANL/APS/LS-330, March 10, 2012.

[37] J. Bengtsson and J. Irwin, "Analytical Calculations of Smear and Tune Shift," SSC-232, Feb. 1990.

[38] K. Bane, "Corrugated Pipe as a Beam Dechirper," SLAC-PUB-14925, April 2012.

[39] Y.S. Tsai, Rev. Mod. Phys. 46, 815 (1974)

[40] A. Wrulich, CERN Accelerator School 94-01, Vol. 1, 409 (1994).

[41] J. LeDuff, NIM A 239 (1985) 83-101.

[42] A. Franchi *et al.*, Phys. Rev. ST Accel. Beams 17, 074001 (2014).

[43] . K. Floettmann, Phys. Rev. ST Accel. Beams 6, 034202 (2003).

[44] T. Berenc, M. Borland, and R. R. Lindberg, "Modeling RF Feedback in Elegant for Bunch-Lengthening Studies for the Advanced Photon Source Upgrade," Proc. of IPAC15, MOPMA006 (2015).

[45] K. Hwang and S. Y. Lee, "Dipole fringe field thin map for compact synchrotrons", Phys. Rev. ST Accel. Beams **18**, 122401, 2015; K. Hwang, "On intrinsic nonlinear particle motion in compact synchrotrons," Indiana University Ph. D. Thesis, 2015.

[46] M. Borland, "Android application for accelerator physics and engineering calculations," Proc. of PAC 2013, 1364-1366.

[47] T. Nakamura *et al.*, "Transverse bunch-by-bunch feedback system for the SPRing-8 Storage Ring," Proc. of EPAC 2004, 2649.

[48] A. Chao *et al.*, "Tune shifts of bunch trains due to resistive vacuum chambers without circular symmetry," Phys. Rev. ST Accel. Beams, **5**, 111001 (2002).

[49] Y. Bacconier and G. Brianti, CERN/SPS/80-2 (1980).

[50] M. Venturini and A. Dragt, "Accurate computation of transfer maps from magnetic field data," NIM A 427 (1999) 387-392.

[51] J. R. King, I. V. Pogorelov, M. Borland, R. Soliday, K. Amyx, "Current status of the GPU-Accelerated version of elegant," Proc. IPAC15, 623 (2015).

[52] M. Bassetti, G. Erskine, CERN ISR TH/80-06 (1980).

[53] B. Buesing, private communication, 2018.

[54] R. Lindberg, private communication, 2018.

[55] Y.-P. Sun and C.-Y. Yao, "Tracking With Space Harmonics in ELEGANT Code," Proc. NA-PAC19.

[56] Y. Li *et al.*, "Fast dynamic aperture optimization with reversal integration," arXiv:1912.00121, 30 November 2019.

[57] C. E. Mitchell, "Calculation of Realistic charged-particle transfer maps." Ph.D. thesis, University of Maryland, College Park, 2007. http://www.physics.umd.edu/dsat/

[58] M. Venturini, "Particle dynamics and misaligned lattice elements," ALS Technical Note ALSU-AP-2021-001, Feb. 2021.

[59] M. Bassetti and G. Erskine, CERN ISR TH/80-06 (1980).

[60] M. Furman, "Compact Complex Expressions for the Electric Field of 2-D Elliptical Charge Distributions," LBL-34682 (2007).

[61] M. Borland, "Symplectic integration in `elegant`," ANL/APS/LS-356, Rev. 1, Dec. 2021.

[62] X. Huang, "An algorithm for online optimization of accelerators," NIM A 726, 77-83, 2013.

[63] Z. Zhang *et al.*, "Electron beam energy chirp control with a rectangular corrugated structure at the Linac Coherent Light Source," Phys. Rev. ST Accel. Beams 18, 010702 (2015).