User's Guide to Program `clinchor`

L. Emery

July 1, 2016

1 Introduction

The program `clinchor` calculates the growth rates of longitudinal and transverse coupled bunch modes in an electron storage ring. This program combines features that are found separately in other programs such as ZAP[1], BBI[2] and PC-BBI[3]. The effort in writing another coupled bunch program is justified further by the opportunity to implement more flexible input and output methods. The input command file is written entirely in namelist commands, and the input and output data files are in SDDS format. The postprocessing and graphics are done outside of `clinchor` using the `SDDS` toolkits.

The method of calculation in `clinchor` is based on the normal mode analysis of K. Thompson and R. Ruth[4]. The bunches are treated as point charges and therefore only rigid bunch modes are calculated, which means Landau damping and coherent damping from short-term wake-fields is not considered. The effect of bunch length on the strength of the wake fields generated is take into account, however.

For those with inquiring minds, the name of the program comes from the expression "**c**alculation of **l**ongitudinal and transverse coupled bunch **in**stability due to **c**avity **h**igher-**or**der mode resonators."

The rest of the guide is organized in three sections. The general features are presented in the next section. In section 3, the input commands are explained in more detail. The HOM definition file format is explained in section 4.

2 General Features

The calculation features of the program are:

1. treatment of general bunch distributions plus randomization of charge,
2. inclusion of bunch form factor, which reduces the effect of impedance on the beam,
3. shifting of one or more HOM frequencies to the closest dangerous resonant frequency,
4. sweeping of one cavity higher-order mode (HOM) frequencies,
5. randomization of the cavity higher-order mode (HOM) frequencies for Monte Carlo simulations, i.e representing changes that might occur during operations,
6. frequency-stepping (staggering) of the HOM frequencies across cavities,
7. effect of the beta function at the RF cavities for transverse motion.

The implementation features of the program are:

1. `C` language coding,
2. input file with namelist commands,
3. HOM input data stored in convenient `SDDS`-format files, along with flags for number of cavities and setting to resonance,
4. output files compatible with `SDDS` toolkits,

These items are briefly explained below.

2.1 General Bunch Distribution

Programs like ZAP and BBI, and the theory upon which they are based can only treat symmetric bunch distribution. Thompson and Ruth[4] developed a simple theory to calculate modes of irregularly spaced bunches. Their formalism also allows the possibility of bunches of different charge. In order to get the flexibility of having the most general bunch patterns, program `clinchor` has three commands to generate a symmetric bunch pattern, a bunch train, and a general bunch pattern. See namelist commands `symmetricBunchPattern`, `bunchTrain`, and `generalBunchPattern` in section 3. The bunch pattern can be read from a file using `bunchPatternFromFile`. Namelist command `randomizeBunchCurrent` generates a random component to the bunch distributions after the beam is defined.

2.2 Bunch Form Factor

Bunches with non-zero length have power spectral densities that decay with frequency. This will decrease the effect of the impedances, which improves stability expectations. The program can read a file of rms bunch lengths as a function of current for determining the form factors for each bunch in the growth rate calculations. It is assumed that the longitudinal distribution of the bunches is Gaussian.

2.3 Shifting of the HOM Frequencies to the Closest Dangerous Resonant Frequency

A monopole HOM impedance can contribute the maximum growth rate to a symmetric longitudinal mode when the HOM frequency falls on a revolution harmonic upper-synchrotron sideband. Similarly, a dipole HOM impedance can contribute the maximum growth rate to a symmetric transverse mode when the HOM frequency falls on a revolution harmonic lower-betatron sideband. An optional data column may be included in the HOM definition file (see section 4) as a flag to force selected HOM frequencies onto dangerous resonances. Because of the relatively low probability for HOM frequencies to hold exactly these resonant values, this option is mostly useful in checking the calculated growth rates against those obtained in hand calculations.

2.4 Sweeping of the HOM Frequencies

The setup command `sweepFrequency` allows one to sweep the frequency of a selected HOM through some range. With a little postprocessing, a plot of the growth rate as a function of that HOM frequency may be obtained. This is a feature borrowed from BBI. The results can be used to cross-check with formulas of single HOM.

2.5 Randomization of the HOM Frequencies

This optional feature (borrowed from PC-BBI) allows one to make a Monte Carlo simulation of the CBM growth rates using the HOM frequencies as random numbers. One can then determine the probability of instability for the set of HOMs defined. See namelist command `randomizeHOMFrequencies`.

2.6 Staggering of the HOM Frequencies

The frequency stepping (staggering) of the HOM frequencies in an RF system reduces the probability that two or more HOM impedances will contribute to the same coupled bunch mode. The staggering is accomplished by varying the cavity dimensions in uniform steps from cavity to cavity. The staggering is expected to vary with HOM type. Staggering steps per cavity are entered as an optional column in the HOM definition file.

2.7 Beta Functions at the RF Cavities

This concerns only the transverse coupled bunch mode calculations. All previous calculations of coupled bunch motion that I know (ZAP, BBI) assume a constant beta function around the ring (smooth focusing approximation). There is no reason not to use the correct beta function values (it is not that complicated), therefore `clinchor` uses and requires values of the beta functions at the RF cavities.

2.8 `C` Language

The usual advantages of a `C` code versus Fortran code applies here (as of the writing in 1992; Fortran may catch up in future years). In `C`, one can dynamically allocate memory for structure arrays and matrices as the calculation requires them. The memory is freed when it is no longer needed. This results in a smaller executable file. `C` code, in general, is easily ported to other types of computers. (So far the program has been ported to various Unix, Linux, and Windows OSs since 1992.) The code is written in a verbose style (i.e., variable names are based on un-abbreviated physical quantity names), which makes it easy for anyone to understand the code.

2.9 Command Input File

To run `clinchor` one needs are least two input files. One containing the namelist commands, and one for the HOMs definitions. The input file containing the namelist commands is specified on the shell command line as in this example,

```
clinchor input.cb
```

where `input.cb` is the input file. The complete list of namelist commands and variables is given in section 3.

Macros (tag-value pairs) can be defined on the command line for substituting strings that appear in the input file between "<" and ">" characters. For example,

```
clinchor input.cb -macro=totalCurrent=102,dipoleHOMs=dipole.sdds
```

where tags <`totalCurrent`> and <`dipoleHOMs`> might appear in the input file.

2.10 HOM Definition File

The HOM data is contained in a SDDS format file rather than appearing explicitly in the input command file. The choice of the SDDS format facilitates the organization of the HOM data.

2.11 Output files

The main output file is directed to the standard output by default. The file echoes the namelist commands and contains some secondary physical quantities, which the user should verify to ensure that no data entry error occurred.

Other output files are generated for specific calculations.

3 List of Namelist Commands

The namelist commands in the command input file define the whole problem: the ring parameters, the bunch distribution of the beam, the cavity HOM impedances, the optional instructions to setup the calculation, and the calculation itself. Each namelist commands use variables which further specify the command.

The namelist commands are processed by subroutines from object libraries made available publicly by M. Borland.

Below, the namelist commands in `clinchor` are listed and grouped according to their function:

Basic ring parameter	`ringParameters`
Cavity HOM definitions	`monopoleHOMs`
	`dipoleHOMs`
Beam definition	`symmetricBunchPattern`
	`bunchTrain`
	`generalBunchPattern`
	`bunchPatternFromFile`
	`randomizeBunchCurrent`
Option setup commands	`randomizeHOMFrequencies`
	`sweepFrequency`
Calculation commands	`doLongitudinalMotion`
	`doTransverseMotion`

The following pages describe all the namelist commands. For each command, one will find:

- the type and function of the command,
- a command definition listing, and
- a detailed explanation of the variables of the command.

The command definition listing is of the form

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

The component `<variable-type>` can be one of three types:

- `double` — for a double-precision type, used for most physical quantity variables,
- `long` — for an integer type, used for some physical quantity variables, such as the harmonic number, but most often for the logical flag variables, where 0 mean false and 1 means true.
- `STRING` — for a character string enclosed in double quotes, used most often for a filename variable.

Note that the component `<variable-type>` appears in these pages only as a place-holder, and shouldn't be used literally in an actual input file.[1] The namelist in an input file should look like this:

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

The square brackets denotes an optional component. Not all variables need to be defined – the defaults may be sufficient. Those that do need to be defined are noted in the detailed explanations. The only variables that don't have default values in general are string variables.

Array variables take a list of values with the first one assigned to the array element `index`. If the index value isn't given, then the namelist processor assumes the first array element is number zero. The case of the letters in all namelist and variable names is important.

One will note that quantities which aren't traditionally described in MKS units have their units appearing explicitly in their names. This convention is used to avoid confusion about which non-MKS units to use.

Whenever a namelist is read, it is written back to the standard output file, which is the screen device unless the output is redirected.

In general, the namelist commands can be repeated as many times as one wishes. The reason for repeating may be to change a variable value between calculation commands, to assemble HOM definitions in a modular way, or to construct a more complex bunch pattern.

[1]macro place-holders in the input file would have explicit angle brackets enclosing a tag, as explained above.

3.1 ringParameters

- type: Setup command.
- function: Set basic ring parameters necessary to calculate synchrotron frequency, bunch length, and so on. Once the namelist is processed and written to the standard output file, a list of dependent quantities used in the growth rate calculations are written to the standard output file.

```
&ringParameters
    STRING twissFile = NULL;
    double energyGeV = 0.0
    double circumference = 0.0
    double energyLossPerTurnMeV = 0.0
    double rfVoltageMV = 0.0
    long harmonicNumber = 1
    double momentumCompaction = 0.0
    double relativeEnergySpread = 0.0
    double bunchLengtheningFactor = 1.0
    STRING bunchLengthTableFile = NULL;
    STRING bunchLengthUsedFile = NULL;
    double longDampingTime = 0.0
    double transDampingTime = 0.0
    double horizontalTune = 0
    double verticalTune = 0
    double betaxAtRFCavities = 0
    double betayAtRFCavities = 0
    STRING parameters = NULL;
&end
```

- `twissFile` — If not blank, then parameters of the lattice are taken from the named file, which is assumed to be an `elegant` Twiss output file. You must turn on the radiation-integral calculation in `elegant` or `clinchor` will complain of missing data. The data taken from the file are energy, circumference, energy loss per turn, momentum compaction, relative energy spread, damping times, tunes, and beta functions at the cavities. For the last item, `clinchor` looks for elements of type `RFCA` and `RFCW`. It averages the beta functions over all elements found. If you wish to override the values in the file, simply give a nonzero value to the appropriate namelist variable.
- `energyGeV` — Energy of the stored beam in units of GeV.
- `circumference` — Circumference of the storage ring in meters. Used to calculate revolution frequency f_0.
- `energyLossPerTurnMeV` — Energy loss per turn in units of MeV.
- `rfVoltageMV` — Peak RF voltage in unit of MV of cavities in ring.

- `harmonicNumber` — Ratio between external RF frequency f_{RF} and the revolution frequency f_0. Used to determine f_{RF}.
- `momentumCompaction` — Momentum compaction factor. Used to determine unperturbed bunch length.
- `relativeEnergySpread` — Relative energy spread. Used to determine unperturbed bunch length.
- `bunchLengtheningFactor` — Ratio of expected bunch length and unperturbed bunch length, which might be obtained from a separate potential well calculation. This quantity doesn't affect the calculation coherent synchrotron frequency. The bunch lengthening factor appears only in the bunch form factor in the effective HOM impedance expression. This bunch length applies to all bunches in the calculation irrespective of the charge they contain.
- `bunchLengthTableFile` — Input file of bunch length in units of seconds as a function of charge to be used for individual bunches in the calculation. This bunch length is used as form factor to reduce the wake fields for a given resonator. This should be used instead of `bunchLengtheningFactor` when possible.
- `bunchLengthUsedFile` — Output file of bunch lengths for each bunch defined in the calculation. Columns are bucket number, bunch current and bunch length. There will be one row entry for every bucket filled. The output files is created only when the file `bunchLengthTableFile` is specified. Recommended value is "`%s.bl`" where the "`%s`" is replaced with the root name of the command file.
- `longDampingTime, transDampingTime` — Coherent damping time constants in seconds for the motions in the longitudinal and transverse planes. If one doesn't know the coherent damping times, one may use the synchrotron radiation damping times, which are longer. Damping rates are calculated from the damping times, and are simply added to the CBM growth rates to give the final growth rates. When doing Monte Carlo studies, I prefer to ignore the damping rates by making them explicitly zero in the namelist by setting `longDampingTime`, and `transDampingTime` to zero.
- `horizontalTune, verticalTune` — Tunes of the stored beam. Values are required if command `doTransverseMotion` is used.
- `betaxAtRFCavities, betayAtRFCavities` — Beta functions at the RF cavities. Values are required if command `doTransverseMotion` is used. The transverse growth rates will scale with one of the two values, depending on which direction is specified in the `doTransverseMotion` command.
- `parameters` — Output file containing above parameters plus some calculated ones.

3.2 monopoleHOMs

- type: Setup command. Required if the next action command is `doLongitudinalMotion`.
- function: Read in a `SDDS` format file of HOM definitions to form or add to an internal monopole HOM data structure array. Some of the namelist variables can be used to modify the HOM properties once the growth rate calculation starts.

```
&monopoleHOMs
    STRING filename = NULL,
    long clearPreviousMonopoleHOMs = 0,
    double deQFactorMultiplier = 1 ;
    double Q = 0;
    double staggeringStepMultiplier = 1;
    long detuneFundamental = 0;
    long keepFundamentalFixed = 1;
&end.
```

- `filename` — `SDDS` format file of HOM definitions to be interpreted as monopole HOMs. This is the only required variable of the namelist command. The contents of the file is explained in section 4.
- `clearPreviousMonopoleHOMs` — If this logical flag is set to 1, then the current array of monopole HOMs data structure is cleared before the file `filename` is read. If the flag is 0, then the HOM information in the file `filename` is added to the current array of monopole HOMs data structure. Thus there can be more than one `&monopoleHOMs` with different data files.
- `deQFactorMultiplier` — Factor by which all Q's in the file will be divided before running a calculation.
- `Q` — All HOMs of this file will be assigned the value of Q specified here.
- `staggeringStepMultiplier` — If there is a frequency staggering step quantity in the HOM file, then this factor is used to increase the staggering step before a calculation is done.
- `detuneFundamental` — If nonzero, then `clinchor` looks in the HOM list for the mode that most closely matches the fundamental rf frequency (computed from the circumference and the harmonic number). The program sets that HOM frequency to the fundamental rf frequency, then detunes it to compensate beam loading.
- `keepFundamentalFixed` — If nonzero, then the fundamental frequency will not be randomized along with other HOMs. This is to simulate what is done in practise, that the fundamental mode tuning is kept fixed by feedback loops. This fixing of fundamental mode can be done through the data column "Fixed" in the HOM file as well.

3.3 dipoleHOMs

- type: Setup command. Required if the next action command is `doTransverseMotion`.
- function: Read in a `SDDS` format file of HOM definitions to form or add to an internal dipole HOM data structure array. Some of the namelist variables can be used to modify the HOM properties once the growth rate calculation starts.

```
&dipoleHOMs
    STRING filename = NULL
    long clearPreviousDipoleHOMs = 0
    double deQFactorMultiplier = 1;
    double Q = 0;
    double staggeringStepMultiplier = 1;
    long detuneFundamental = 0;
&end.
```

- `filename` — `SDDS` format file of HOM definitions to be interpreted as dipole HOMs. This is the only required variable of the namelist command. The contents of the file is explained in section 4.
- `clearPreviousDipoleHOMs` — If this logical flag is set to 1, then the current array of dipole HOMs data structure is cleared before the file `filename` is read. If the flag is 0, then the HOM information in the file `filename` is added to the current array of dipole HOMs data structure. Thus there can be more than one `&monopoleHOMs` with different data files.
- `deQFactorMultiplier` — Factor by which all Q's in the file will be divided before running a calculation.
- `Q` — All HOMs of this file will be assigned the value of Q specified here.
- `staggeringStepMultiplier` — If there is a frequency staggering step quantity in the HOM file, then this factor is used to increase the staggering step before a calculation is done.
- `detuneFundamental` — If nonzero, then `clinchor` looks in the dipole HOM list for the mode with the lowest frequency. The program sets that HOM frequency to the closest harmonic of the fundamental rf frequency, then detunes it to compensate beam loading. This has applications only for a deflection-mode cavity running at the harmonic of the rf frequency.

3.4 symmetricBunchPattern

- type: Setup command.
- function: This command defines a symmetric bunch pattern.

```
&symmetricBunchPattern
    long startBucket = 0
    long bunches = 1
    double currentPerBucketMA = 0.0
    double totalCurrentMA = 0.0
    long clearPreviousPatterns = 0
&end.
```

- `startBucket` — Bucket number of the first bunch in the bunch train. In the case of a symmetric bunch pattern, `startBucket` is the bucket number where one of the bunches is located. Default is 0, the first bucket.
- `bunches` — Number of bunches in the symmetric distribution. If `bunches` is not a divisor of the RF harmonic number, then a close approximation of the symmetric distribution is generated.
- `currentPerBucketMA` — Current in mA for each bunch defined.
- `totalCurrentMA` — Total current of all bunches defined in this command. Used to determine the current of individual bunches. Either variables `currentPerBucketMA` or `totalCurrentMA` may be defined. If both are present, variable `currentPerBucketMA` takes precedence.
- `clearPreviousPatterns` — If this flag is set to 1, then bunch patterns defined in previous `symmetricBunchPattern` or `generalBunchPattern` are cleared before setting the bunches of this namelist command.

3.5 bunchTrain

- type: Setup command.
- function: This command defines a train of equally spaced bunches of equal charge.

```
&bunchTrain
    long startBucket = 0
    long bucketInterval = 1
    long bunches = 1
    double currentPerBucketMA = 0.0
    double totalCurrentMA = 0.0
    long clearPreviousPatterns = 0
&end.
```

- `startBucket` — Bucket number of the first bunch in the bunch train.
- `bucketInterval` — Bunch spacing interval for a train of equally spaced train of bunches. The number of empty buckets in between the bunches is `bucketInterval`-1.
- `bunches` — Number of bunches in the train.
- `currentPerBucketMA` — Current in mA for each bunch defined.
- `totalCurrentMA` — Total current of all bunches defined in this command. Used to determine the current of individual bunches. Either variables `currentPerBucketMA` or `totalCurrentMA` may be defined. If both are present, variable `currentPerBucketMA` takes precedence.
- `clearPreviousPatterns` — If this flag is set to 1, then bunch patterns defined in all previous bunch defining commands are cleared before creating the bunches of this namelist command.

3.6 generalBunchPattern

- type: Setup command.
- function: The command defines a general distribution of bunches using STRING variables containing a list of bunch positions and current values.

```
&generalBunchPattern
    STRING bucketSelection = NULL
    STRING currentPerBucketMA = NULL
    double totalCurrentMA = 0.0
    long clearPreviousPatterns = 0
&end.
```

- `bucketSelection` — String containing a list of bucket numbers (long integers) for bunches to fill. A string variable is necessary for this input because a string has no pre-set length, while an array variable must be defined with a pre-set length.
- `currentPerBucketMA` — String containing a list of bunch currents (double floating point numbers) corresponding to each bucket number specified in `bucketSelection`. If the variables `bucketSelection` and `currentPerBucketMA` have unequal number of entries, then the longer one is truncated. Here is an example of the use of the string variables for generating some arbitrary general pattern:

  ```
  &generalBunchPattern
      bucketSelection = "1 4 9 16"
      currentPerBucketMA = "1.0 2.0 3.0 4.0"
  &end.
  ```

- `totalCurrentMA` — Total current to be distributed evenly among bunches defined in `bucketSelection`. This variable is used when variable `currentPerBucketMA` is not specified.

3.7 bunchPatternFromFile

- type: Setup command.
- function: The command reads in a file to define a presumably general distribution of bunches. This can be the `bunchLenghtUsedFile` output file produced by a previous run of `clinchor`, since that file contains the bucket number and bunch current.

```
&generalBunchPattern
    STRING filename = NULL
    long clearPreviousPatterns = 0
&end.
```

- `filename` — Input file with column Bucket and Current (mA).

3.8 randomizeBunchCurrent

- type: Setup command.
- function: The command defines the randomization parameters for the bunch pattern. The randomization is performed on the ideal bunch pattern before each growth rate calculation. If a sweep of frequency is requested (`sweepFrequency`), then a new random bunch pattern is generated for each HOM frequency value. If a HOM-frequency Monte Carlo simulation is requested (`randomizeHOMFrequencies`), then a new random bunch pattern is generated for each HOM frequency sample. Thus there are two random components in the resulting growth rate distribution: HOM frequency and bunch population.

```
&randomizeBunchCurrent struct
    long seed = -987654323;
    long uniform = 1;
    double absoluteSpreadMA = 0;
    double relativeSpread = 0;
&end.
```

- `seed` — A large negative number for the random number generator. This will use an independent generator from the HOM frequency.
- `uniform` — Flag for uniform distribution. Presently only uniform distribution is available.
- `absoluteSpreadMA` — Spread in bunch distribution in units of mA. This is useful in top-up mode injection where the total current is constant but individual bunches have spread equal to the charge per injection bunch. For example if a bunch pattern has 4 mA per bunch and the spread is 1 mA, then some bunches may have current as low as 3.5 mA and others as high as 4.5 mA.
- `relativeSpread` — Spread in bunch distribution in fraction.

3.9 sweepFrequency

- type: Setup command
- function: Sets up a sweep of the resonator frequency value of a selected HOM. The setup takes effect only when an action command is executed.

```
&sweepHOMFrequency
    long resonatorIndex = 0
    double frequencyRange = 0.0
    long points = 100
    STRING filename = NULL
&end
```

- `resonatorIndex` — HOM resonator number to be swept in frequency. The numbering starts at 0, and proceed in the order that the resonators were read in the HOM definition file or files.
- `frequencyRange` — Sweep range in Hz. The range starts at the unperturbed HOM resonator frequency minus `frequencyRangeHz` and ends at the unperturbed HOM resonator frequency plus `frequencyRangeHz`. If `frequencyRangeHz` is set to 0 or not specified, then `frequencyRangeHz`=f_0 where f_0 is the revolution frequency.
- `points` — Number of points in the sweep.
- `filename` — `SDDS` file in which the results are written. The columns are the swept HOM frequency, the maximum growth rate, and the frequency shift of the fastest growing CBM. Recommended naming conventions: `<something>.CBMFreq`

3.10 randomizeHOMFrequencies

- type: Setup command
- function: Sets up a Monte Carlo simulation of randomized HOM frequencies for all HOMs. The setup takes effect only when an action command is executed.

```
&randomHOMFrequencies
    double spread = 0.0
    long seed = -987654321
    long uniform = 1
    long samples = 100
    STRING CBMFrequencyFilename = NULL
    STRING HOMFrequencyFilename = NULL
&end
```

- `spread` — The range of the random component of the frequency values is `-spread` to `spread` in Hz. If `spread` is set to zero or left undefined, then `spread` $= f_0$ where f_0 is the revolution frequency.
- `seed` — A large negative number for the random number generator. This will use an independent generator from the bunch population randomizer.
- `uniform` — If flag is set to 1, then a uniform distribution for the random component of the HOM frequencies is used. Since no other random distribution type is available in this present version of `clinchor` using other values of `uniform` makes no difference.
- `samples` — Gives the number of times a following action command (`doLongitudinalMotion` or `doTransverseMotion`)is executed with a different sampling of random HOM frequencies for each calculation.
- `CBMFrequencyFilename` — File containing the growth rate value of the fastest growing mode in each Monte Carlo sample. This file can then be histogram med with the `SDDS` toolkit program `sddshist`. Recommended naming conventions: `<something>.CBMFreq`
- `HOMFrequencyFilename` — `SDDS` file containing the randomized HOM frequencies used by each Monte Carlo sample. Type “`sddsquery <filename>`” for the full description of the file. This file may grow to be very large, so be careful with this option. This variable is optional. Recommended naming conventions: `<something>.HOMFreq`

3.11 doLongitudinalMotion

- type: Action command
- function: Does calculation of longitudinal CBM growth rates using monopole HOMs and bunch definitions currently defined. Monopole HOMs and a bunch pattern must have been defined previously for this command to work. Either one of the optional commands `sweepFrequency` and `randomizeHOMFrequencies` may appear before. If both appear, the last one takes precedence.

  If the optional command `sweepFrequency` is in effect, the `SDDS` file specified by the `filename` variable in that namelist command is opened at this point.

  If the optional command `randomizeHOMFrequencies` is in effect, the `SDDS` files specified by the `filename` and `frequencyListFilename` variables in that namelist command are opened at this point.

```
&doLongitudinalMotion
   long normalModes = 1
   long doLaplace = 0
   STRING eigenvectorFilename = NULL;
   STRING CBMFrequencyFilename = NULL
&end
```

- `normalModes` — If set to 1, then CBM complex frequencies are calculated, which is the only present goal of the program.
- `doLaplace` — Not yet available. This solves the motion as a function of time with initial conditions using an inverse Laplace transform, as explained in [4].
- `eigenvectorFilename` — Print the normal mode matrix used in the CBM frequencies calculation. Recommended naming conventions: `<something>.v`
- `CBMFrequencyFilename` — `SDDS` file listing the complex frequencies of all the CBMs. This file may grow very large with the Monte Carlo simulation. Recommended naming conventions: `<something>.CBMFreq`

3.12 doTransverseMotion

- type: Action command
- function: Does calculation of transverse CBM growth rates using monopole HOMs and bunch definitions currently defined. Of course, monopole HOMs and a bunch pattern must have been defined previously for this command to work. Either one of the optional commands `sweepFrequency` and `randomizeHOMFrequencies` may appear. If both appear, the last one takes precedence.

  If the optional command `sweepFrequency` is in effect, the `SDDS` file specified by the `filename variable` in that namelist command is opened at this point.

  If the optional command `randomizeHOMFrequencies` is in effect, the `SDDS` files specified by the `filename frequencyListFilename` variables in that namelist command are opened at this point.

```
&doTransverseMotion
   long normalModes = 1
   long doLaplace = 0
   long verticalDirection = 0
   long horizontalDirection = 0
   STRING CBMFrequencyFilename = NULL;
   STRING eigenvectorFilename  = NULL;
&end
```

- `normalModes` — If set to 1, then CBM complex frequencies are calculated, which is the present goal of the program.
- `doLaplace` — Not yet available. This solves the motion as a function of time with initial conditions using an inverse Laplace transform, as explained in [4].
- `verticalDirection` — If flag is set to 1, then the vertical tune and the vertical beta function at the RF cavities (see variables of `ringParameters`) are used in the calculation.
- `horizontalDirection` — If flag is set to 1, then the horizontal tune and the horizontal beta function at the RF cavities (see variables of `ringParameters`) are used in the calculation.
- `eigenvectorFilename` — Print the normal mode matrix used in the CBM frequencies calculation.
- `CBMFrequencyFilename` — `SDDS` file listing the complex frequencies of all the CBMs. Recommended naming conventions: `<something>.CBMFreq`

```
stop
```

3.13 semaphores

- type: Action command
- function: Creates small files upon start, failure or normal exit, the presence of which can be used as signals for scripts to monitor a batch of simulations.

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

3.14 stop

- type: Action command
- function: Stops the execution of the program. No variable are defined.

```
&stop
&end
```

4 HOM Definition SDDS File Format

The general SDDS format is defined by two parts, the header and zero or more data tables. The header contains namelist commands defining, among other things, the interpretation of the data in the tables. A table consists of a list of parameters values followed by columns of data. In the HOM definition file, `clinchor` requires the presence of columns `Frequency`, and any two of `ShuntImpedance`, `Q`, `RoQ` (the ratio R/Q). For many columns of physical quanitities the correct units must be specified.

The allowed column names for the HOM definition file are:

- `Frequency` or `f` — Required column. Frequency in units of "Hz" or "MHz". The program internally converts to Hz.
- `ShuntImpedance` or `R` or `Rs` or `Rt` — Required column (see above). Shunt impedance in units of "Ohm" for monopole HOMs, and units of "Ohm/m" for dipole HOMs.
- `Q` or `QualityFactor` — Required column (see above). Quality factor of the resonator.
- `ROverQ` or `R/Q` or `ROQ` — Required column (see above). Shunt impedance divided by Q. In units of "Ohm" for monopole HOMs, and units of "Ohm/m" for dipole HOMs.
- `DeQFactor` or `deQFactor` — Factor by which to reduce the Q of the HOM after reading in this file.
- `StaggeringStep` or `staggeringStep` or `DeltaFrequency` or `deltaFrequency` or `df` — Optional column. Units of Hz only. Default value is 0. If larger than one, then the frequencies of the `NumberOfCavities` HOM resonators are staggered by this interval.
- `ShiftToResonance` — Optional column. Default value is 0. If non-zero, then the frequency of the HOMs is shifted to the closest resonance. If `staggeringStep` is non-zero, then the frequency shifting is done after staggering the frequencies.
- `NumberOfCavities` or `NCavities` or `Cavities` or `cavities` — Optional parameter. Default value is 1. If 0, then this HOM type is not used in the calculation. If greater than one, then the HOM resonator is duplicated internally `numberOfCavities` times. All resonators will be subjected to randomized frequencies separately (if requested by the `randomizeHOMFrequencies` command.)

Here is an example of an SDDS HOM definition file for the APS ring cavities:

```
SDDS1
&description text="APS cavity longitudinal HOMs" contents="HOM definition" &end
&column name=Frequency, symbol=f, units=Hz, type=double,
        description="Resonant frequency of HOM resonator" &end
&column name=ShuntImpedance, symbol=R, units=Ohm, type=double,
        description="Shunt impedance of longitudinal HOM resonator" &end
&column name=Q, symbol=Q, units="", type=double,
        description="Q of longitudinal HOM resonator" &end
&column name=NumberOfCavities, symbol=N$bcav$n, units="", type=long,
        description="Number of cavities for each HOM" &end
&column name=StaggeringStep, units=Hz, type=double,
        description="Staggering frequency steps between cavity HOMs" &end
&column name=ShiftToResonance, type=long,
        description="flag causes automatic shift to resonance after Staggering" &end
&data mode=ascii, noRowCounts=1 &end
558.7e6   13.6e6   68e3   16        -0.08e6   0
761.1e6   25.6e6   53e3   16        -0.7e6    0
962.0e6    6.1e6   54e3   16        -1.2e6    0
1017.4e6   2.6e6   41e3   16        -1.7e6    0
1145.1e6   2.7e6   92e3   16        -1.5e6    0
1219.2e6   3.6e6   41e3   16        -1.9e6    0
```

5 Example Files

Example `clinchor` input files, runs of `clinchor` and post-processing commands are included with the distribution of `clinchor`. The examples can be run with the script files provided.

6 Acknowledgment

The structure of the program is modeled after the program `elegant` written by M. Borland. The program also links to libraries written by M. Borland.

References

[1] M.S. Zisman, S. Chattopadhyay, and J.J. Bisognano, "ZAP User's Guide," LBL 21270, Lawrence Berkeley Laboratory, December 1986.

[2] B. Zotter, "BBI - A Program to Compute Bunch Beam Instabilities in High Energy Particle Accelerators and Storage Rings," CERN LEP/TH 89-74, CERN, 1989.

[3] J. Hagel and B. Zotter, "PC-BBI, a Program to Compute Bunch Beam Instabilities on a PC," CERN SL-AP 90-62, CERN, 1990.

[4] K. Thompson and R. Ruth, "Transverse and Longitudinal Coupled Bunch Instabilities in Trains of Closely Spaced Bunches," in *Proceedings of the 1989 IEEE Particle Accelerator Conference*, p. 792, 1989.

Figure 1: Example of histogram

Figure 2: Examples of postprocessing of the frequency list file

Figure 3: Example of HOM frequency sweep