Introduction to Accelerator Simulation with elegant and SDDS

Michael Borland
Operations Analysis Group
APS Operations Division
Argonne National Laboratory

A talk delivered to the NSLS Lattice Design Group
March 29, 2004

Outline of this Talk

- A brief history of elegant
- Program philosophy
  - Computer science
  - Physics
- Details
- Related programs
- Examples for NSLS II TBA lattice

A Brief History of elegant

- Name stands for
  ELEctron Generation AN Tracking
- Started in 1988 out of frustration with existing tracking codes
- Original purpose was tracking with 2nd order matrices and time-dependent elements
- This talk applies to version 15.1
  - Will be released next week

Some Applications of elegant

- First application: beam transport lines for the SSRL preinjector
- Design of the APS Positron Accumulator Ring and transport lines
- APS top-up safety tracking
- LCLS start-to-end and S2E jitter simulations
- Low-emittance optics development for the APS and APS booster
- Design of bunch compressor and linac optics for LEUTL
- All APS accelerators now use elegant-designed optics

Computer Science Philosophy

- Internal to program
  - Highly-structured C
    - Adding new elements is only as hard as the physics
  - Implement many features as beamline elements to simplify program structure, e.g.,
    - Energy ramping done with a beamline element
    - Beam acquires charge from a beamline element
  - Command file and lattice files are distinct
    - Lattice file is MAD-like
    - Command file is namelist-like

Computer Science Philosophy

- External to program
  - Adopt a tool approach
    - Graphics and display functions external to program
    - Vastly simplifies the simulation code
    - Allows common pre- and post-processing tools for many codes
  - External scripting required and supported
    - Data provided in a form that emphasizes scripting, not manual examination
    - Simplifies the program while empowering the user
    - Allows use of open-source scripting languages
    - Well-suited to cluster computing

UNIX-inspired Toolkit Approach

- In UNIX “Everything is a file”
  - Programs are “filters” that transform files
  - Using pipes allows composing new filters from a sequence of existing filters
  - Anyone who can write code can add a filter
- We do the same thing using SDDS data files
  - Programs are “operators” that transform data
  - Using pipes allows concatenating operators to make a complex transformation
  - Anyone who can write code can add an operator

SDDS

- SDDS = Self-Describing Data Sets
  - A file protocol for data storage
  - A toolkit of programs that transform such files
  - A set of libraries for working with such files
- Knowing SDDS is key to using elegant effectively
  - Pre- and post-processing
  - Graphical and text output
  - Linking of multiple simulations and codes
  - Cluster computing

Why Use SDDS Files?

- Programs that use SDDS are robust and flexible
  - Check existence, data-type, units of data instead of crashing or doing an incorrect calculation
  - Respond appropriately to the data provided
    - Exit and warn user if required data is missing, has unknown units, etc.
    - Supply defaults for missing data (e.g., old data set)
- Existing data doesn't become obsolete when the program is upgraded
- Self describing files make generic toolkits possible, which saves effort on writing pre- and post-processors

Toolkit Approach

- Elegant is a very complex SDDS “operator.” E.g., it
  - Transforms phase space from beginning to end of a system
  - Transforms magnet parameters into, e.g., Twiss parameters, tunes
- All input to and output from elegant is in SDDS files except
  - Lattice structure
  - Command stream

Computer Science Philosophy

- Quality control
  - Elegant is important to APS operations, so we are careful about this
  - Source code is in CVS for version control and tracking
  - Use extensive regression testing to “guarantee” that program updates don't break anything
    - When a feature is added, it is thoroughly tested
    - The test results are saved and used for checking of later versions
    - SDDS largely automates this process

Physics Philosophy

- Choose appropriate, expedient methods rather than some assumed “best” approach
- For tracking, variously use
  - Canonical integration (standard elements)
  - Non-symplectic integration (most time-dependent elements, elements with complex fields)
  - Simple drift-kick-drift (collective effects, rf)
  - Matrices
- Use matrix methods for “everything” except tracking

Capabilities of elegant: Elements

- Second-order matrices for drifts, solenoids, bends, and correctors
- Third-order matrices for quads, sextupoles, and alpha magnets
- User-suppplied second-order matrix
- Fourth-order Ruth integrator for bends, quads, sextupoles, higher multipoles.
  - Hamiltonian has exact energy dependence
  - Can add synchrotron radiation for tracking
- General Taylor map from DA program

Capabilities of elegant: Elements

- Time-dependent elements
  - Kicker with user-specified waveform
  - Rf cavities with exact phase dependence
    - Simple cavity with perfect source
    - Phase-, frequency-, and amplitude-modulated
    - Phase-, frequency-, and amplitude-ramped
    - User-specified on-axis field profile
    - Deflecting cavity
  - Momentum ramp
  - Traveling-wave accelerator

Capabilities of elegant: Elements

- Numerically-integrated elements
  - Planar undulator with co-propagating laser beam
  - Solenoid from user-supplied field map
  - Dipole with extended fringe fields
- Apertures/material
  - One- and two-sided rectangular, elliptical, and super-elliptical collimators
  - Scrapers with optional elastic scattering
  - Foil elastic scattering

Capabilities of elegant: Elements

- Collective effects
  - Intra-beam scattering
  - Short-range longitudinal and transverse wakes
  - Longitudinal- and transverse rf modes
  - Coherent synchrotron radiation in dipoles and drifts
  - Longitudinal space charge in drifts and cavities
- Diagnostics
  - Beam position monitors
  - Particle coordinate/property analysis points
  - Histogram analysis points

Capabilities of elegant: Elements

- Miscellaneous
  - SCRIPT element will incorporate an external program as an element in an elegant lattice
  - Pick-up/driver elements for simulating transverse single-bunch digital feedback

Capabilities of elegant: Calculations

- Twiss parameter computation
  - Optionally-computed on-orbit and with errors
  - Includes radiation integrals
  - Includes chromaticity to third order and first-order tune shift with amplitude
  - Doesn't include coupling terms
  - SDDS output

Capabilities of elegant: Calculations

- Matrix
  - Optionally computed on-orbit and with errors
  - SDDS and text output vs s (first- and second-order)
- Floor coordinates
  - Fully three-dimensional computation
  - 3d misalignment and roll on most elements
  - SDDS output

Capabilities of elegant: Calculations

- Errors
  - Errors for “any” parameter of any element
  - User-selectable distributions, amplitudes, cutoff, ...
  - Linking of errors between elements
  - Multiple error sets in one run
  - Loading of error sets from external files

Capabilities of elegant: Calculations

- Closed orbit
  - Variable or fixed path length
  - On- and off-momentum
  - SDDS output
- Correction
  - Will correct tunes, chromaticities, and orbit
  - Does these sequentially with optional iteration
  - Output of the orbit correction matrix

Capabilities of elegant: Calculations

- Optimization
  - Optimizes user-supplied penalty function depending on almost any calculated quantity
    - Final or interior beam parameters from tracking
    - Final or interior Twiss parameters
    - Global values like equilibrium emittance, tunes, chromaticities
    - Final or interior matrix elements
    - Final or interior floor coordinates
  - Uses Simplex by default, but has other methods

Capabilities of elegant: Calculations

- Saving and loading
  - Optimized lattice can be saved
    - To a new (text) lattice file
    - To an SDDS lattice parameter file
  - SDDS file can be manipulated with Toolkit, reloaded
  - SDDS file can be generated by another program to provide loadable custom error sets
- Variation
  - Nested sweeps “any” parameters of any elements
  - Sweep using values supplied in an external file

Capabilities of elegant: Calculations

- Beam generation
  - Gaussian, hard-edge, and other distributions
  - Optional quiet-start sequences
  - Bunch train generation
  - Initial distribution can be saved to a file
- Beam importation
  - Load beam from SDDS file
    - Previously-generated and saved
    - Previously tracked
  - Sequences of beams in one file

Capabilities of elegant: Calculations

- Aperture finding
  - Searches for aperture boundary with optional subdivision of search interval
- Particle losses
  - Optional output of locations and coordinates of lost particles
  - Optional output of initial coordinates of transmitted particles

Capabilities of elegant: Calculations

- Miscellaneous
  - Frequency map analysis
  - Orbit amplification factors, with correction
  - Slice analysis along a beamline
  - Approximate SASE FEL evaluation
  - Subdivide elements
  - Subprocess execution for pre-, intermediate-, or post-processing
  - Set up semaphore files for flagging run status

Related Programs

- SDDS files increase productivity by letting programs interface smoothly
- Simulations that work with elegant include
  - shower, an interface to the EGS4 electron-gamma shower code
  - spiffe, a PIC code for gun simulations
  - clinchor computes single- and coupled-bunch growth rates due to HOMs
  - URMEL/APS provides mode data in a form that clinchor and elegant accept

Related Programs

- ABCI/APS provides wake data that elegant accepts for tracking
- MAFIA/APS provides data that can be used in tracking with elegant (after a transformation)
- sddsrandmult simulates random construction errors for multipoles and provides data for elegant tracking
- sddsbrightness uses elegant output to compute undulator brightness curves
- sddsemitmeas uses elegant output and experimental data to analyze emittance measurements
- etc. (See Borland et al, PAC 2001)

Basic Structure of an elegant Run

- *Setup commands*, e.g., define beam energy, lattice file, error distributions, correction methods, etc.
- *Minor action commands*, e.g., output Twiss parameters etc. for ideal lattice
- *Major action commands* execute defined calculations in one of several contexts, including
  - Tracking
  - Optimization
  - Aperture determination
- Multiple setup/action sequences allowed

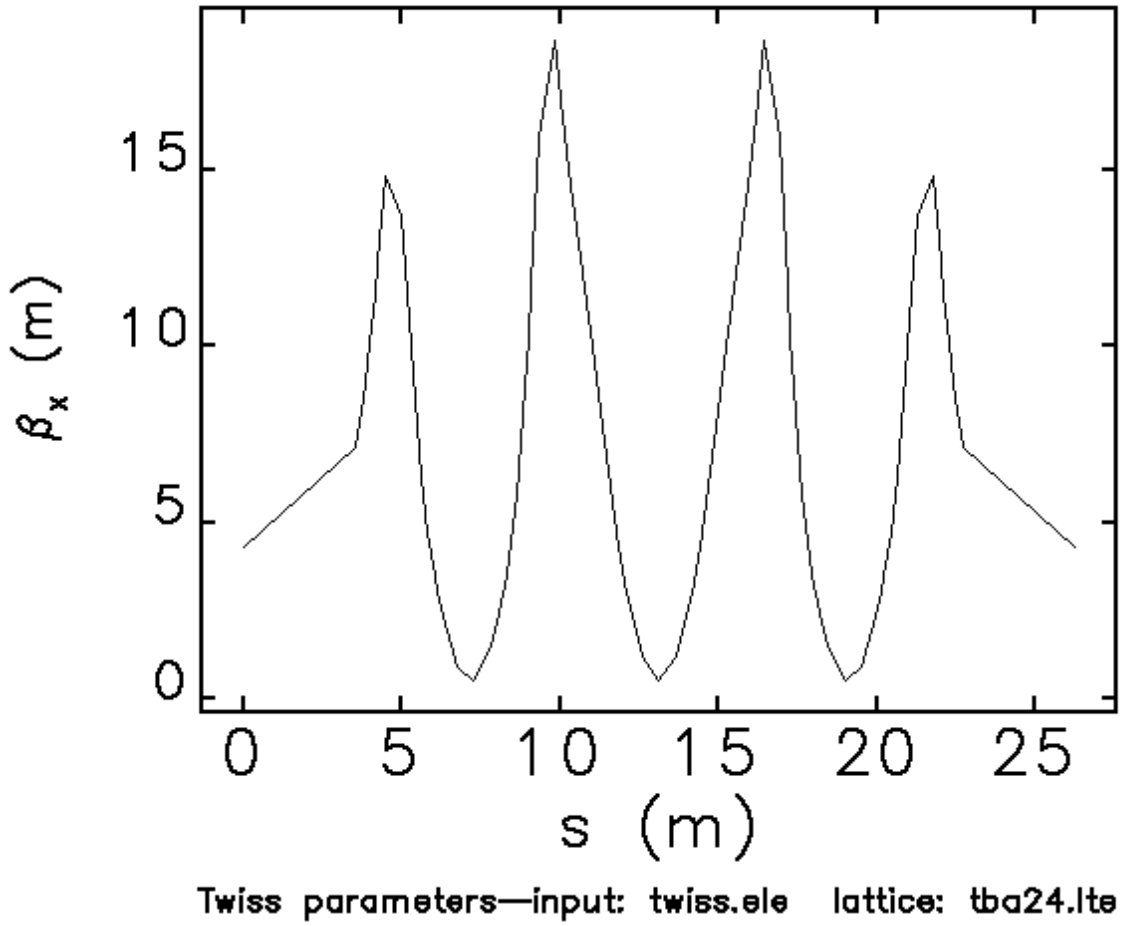
A Simple elegant run: Twiss parameters

```
! File twiss.ele
! Use beamline PER from file twiss.lte at 3 GeV
&run_setup
    lattice = tba24.lte,
    use_beamline = PER,
    magnets = %s.mag,       ! magnet profile data --> twiss.mag
    p_central_mev = 3000.0,
&end

! twiss_output is a minor action command in this instance
&twiss_output
    filename = %s.twi ,       ! twiss parameter data --> twiss.twi
    radiation_integrals = 1   ! include radiation integrals, emittance, etc
&end
```

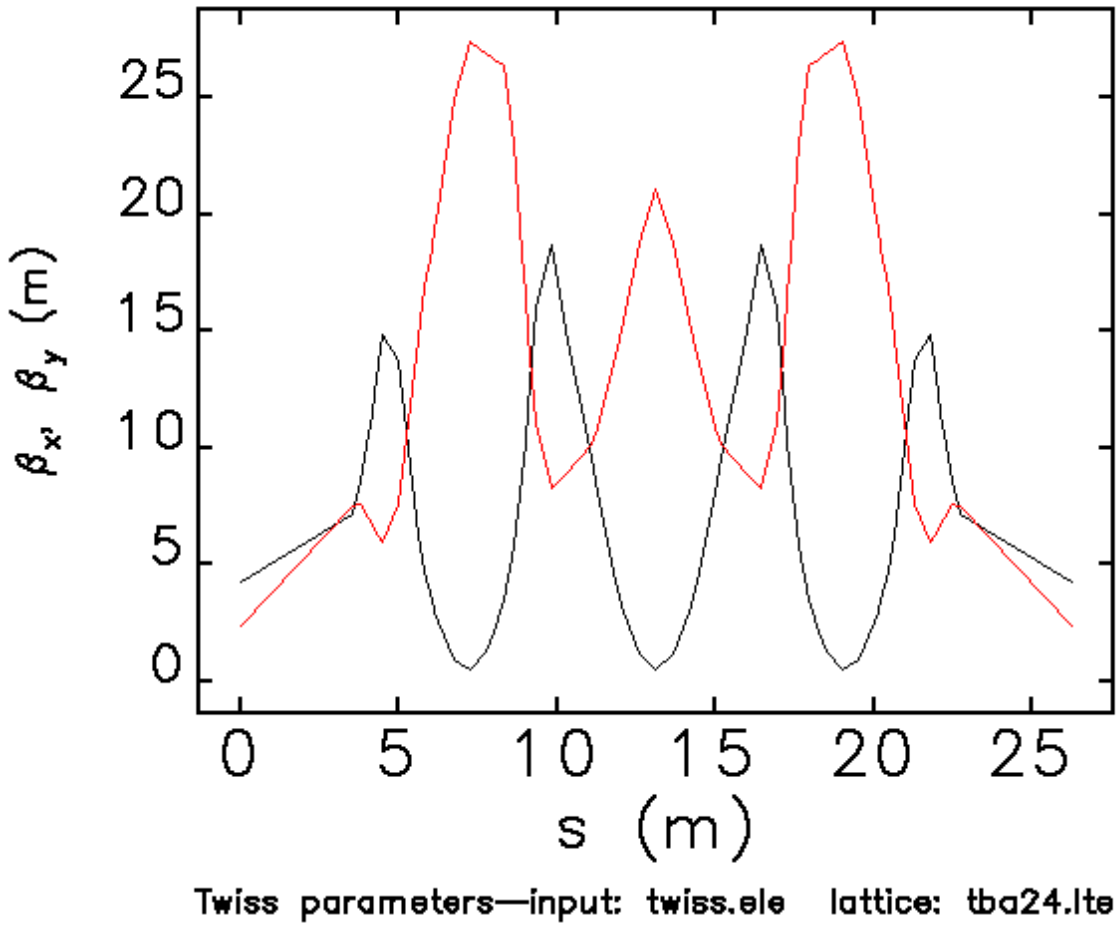
Graphing Data with sddsplot

% sddsplot -column=s,betax twiss.twi



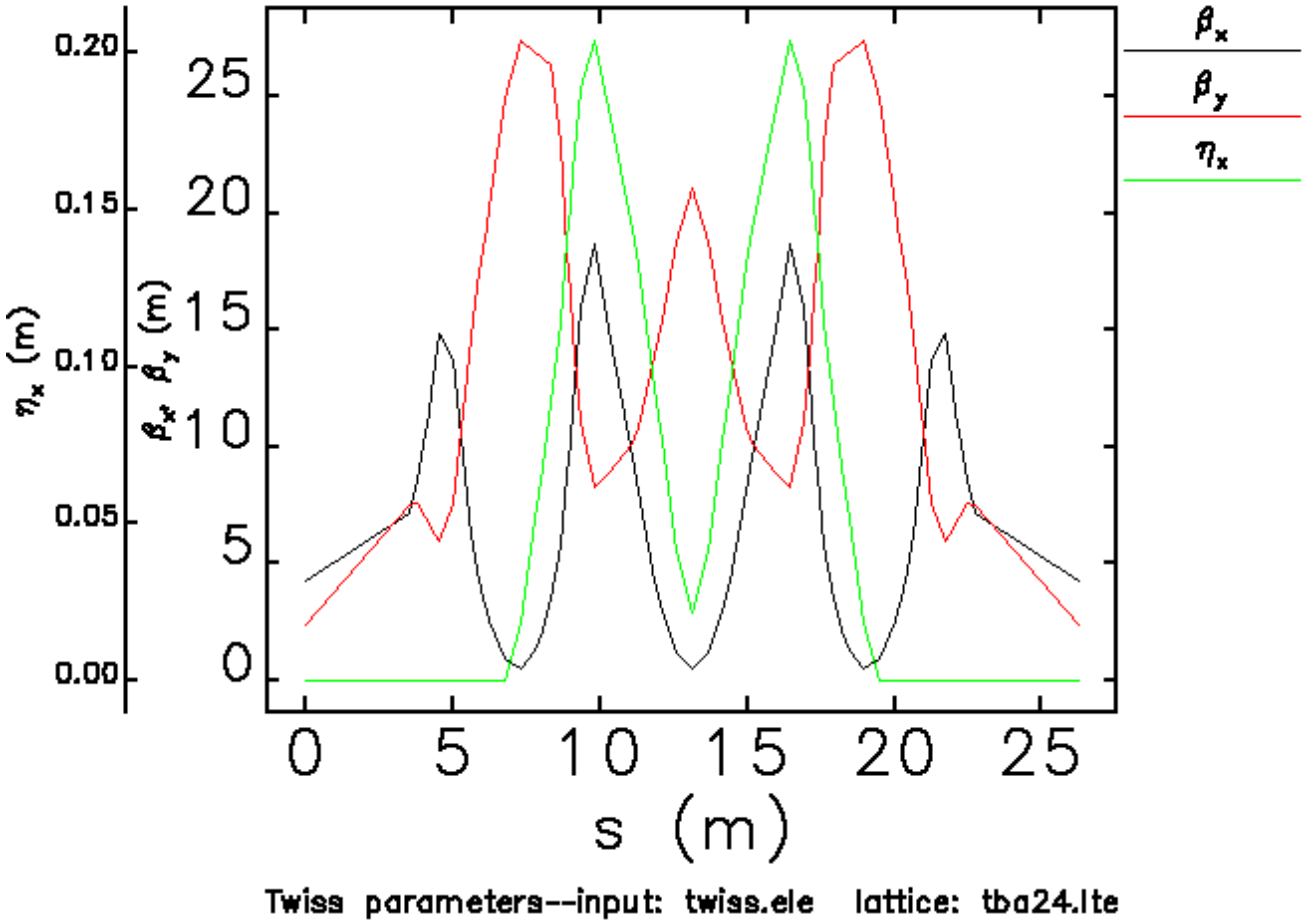
Graphing Data with sddsplot

% sddsplot -graph=line,vary -unsuppress=y -legend -column=s,beta? twiss.twi



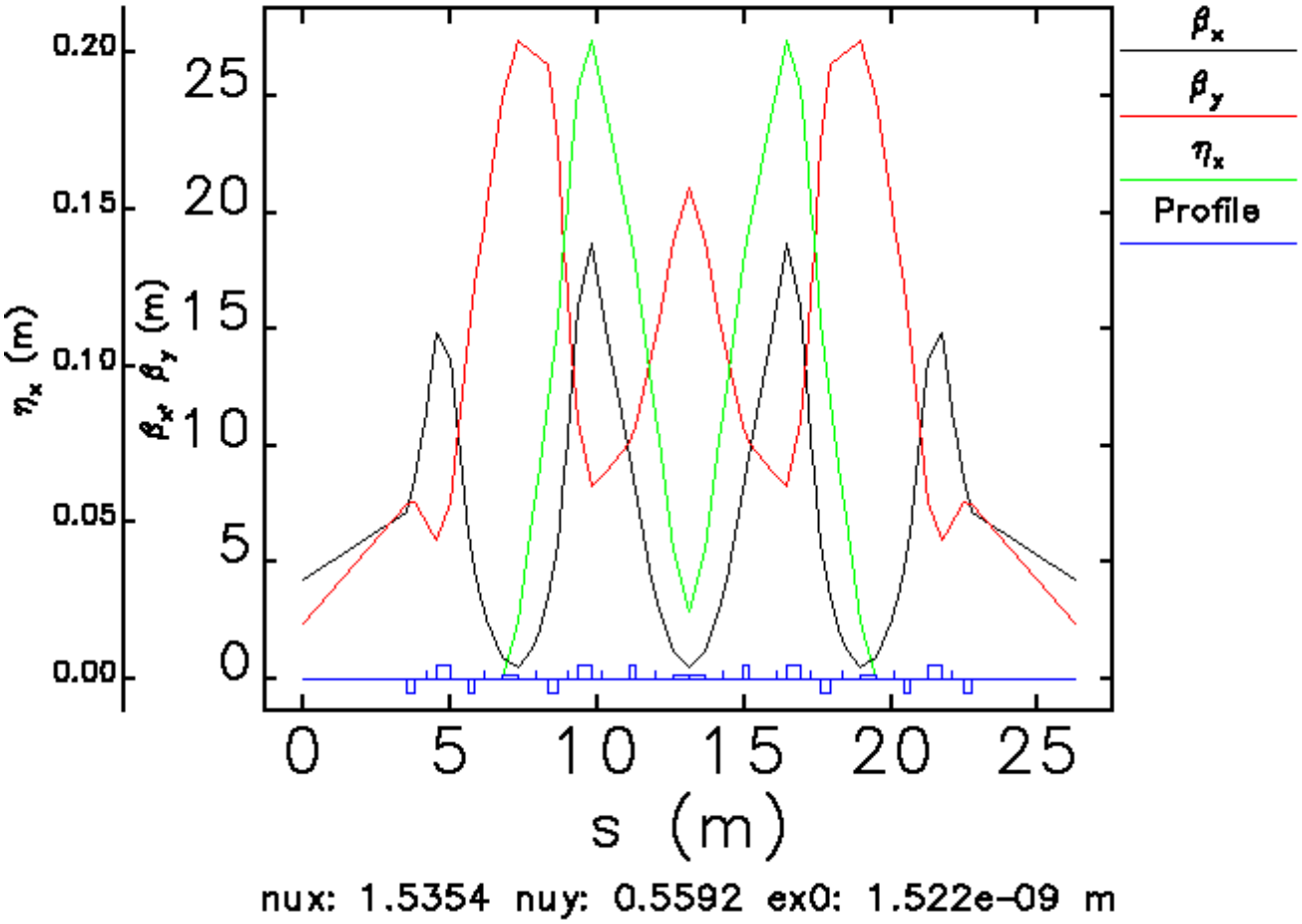
Graphing Data with sddsplot

```
% sddsplot  -graph=line,vary -unsuppress=y -legend \
      -column=s,beta? -yscales=id=beta twiss.twi \
      -column=s,etax -yscales=id=etax twiss.twi
```



Graphing Data with sddsplot

```
% sddsprocess twiss.twi \
      -print=param,aLabel,"nux: %.4f   nuy:  %.4f   ex0: %.4g m",nux,nuy,ex0
%  sddsplot  -graph=line,vary -unsup=y -legend \
      -col=s,beta? -yscales=id=beta twiss.twi -title=@aLabel \
      -col=s,etax -yscales=id=etax twiss.twi  \
      -col=s,Profiles twiss.mag -overlay=xmode=normal,yfactor=0.04
```



Parameter Table

- Generated by sddsprintout and imported. (No manual entry.)

	1 cell	24 cells
nux	1.5354	36.8500
nuy	.5592	13.4200
dnux/dp	.0002	.0044
dnuy/dp	-.0427	-1.0251
betaxMax	18.6949	18.6949
betayMax	27.3760	27.3760
etaxMax	.2040	.2040
alphac2	.0013	.0013
alphac	1.70E-04	1.70E-04
ex0	1.52E-09	1.52E-09
Sdelta0	9.09E-04	9.09E-04
Jx	1.0975	1.0975
Jy	1.0000	1.0000
Jdelta	1.9025	1.9025
taux	.0135	.0135
tauy	.0148	.0148
taudelta	.0078	.0078
U0	.0355	.8527

← Would be zero if 2nd order edge effects were turned off in dipoles.

Chromaticity Calculations

- elegant uses matrices to compute first- and higher-order chromacities
  - This won't get the second-order chromaticity exactly right in all cases since elegant doesn't have third-order dipole matrix
- Can also use tracking and SDDS to determine the chromaticity
  - Use kick elements (energy dependence to all orders)
- Any program worth using should get the same answers both ways

Chromaticity Calculations

Results of matrix analysis using 2nd order dipoles and 3rd order quads and sextupoles

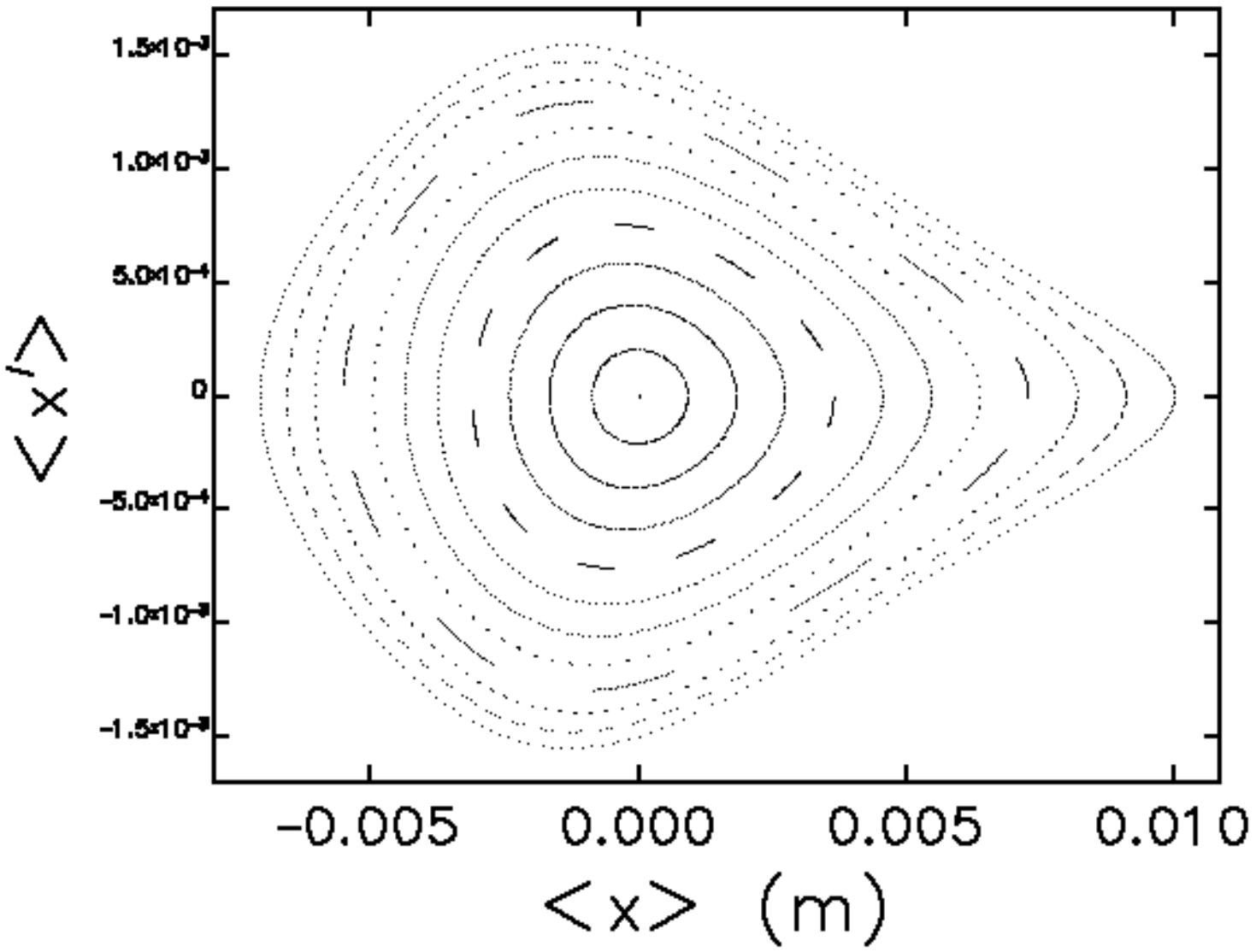
Order	dnux	dnuy
0	3.685E+01	1.342E+01
1	2.554E-02	7.007E-02
2	-3.454E+02	1.703E+02
3	-4.311E+04	-3.682E+03

Results of tracking with kick elements, good to all orders in energy deviation

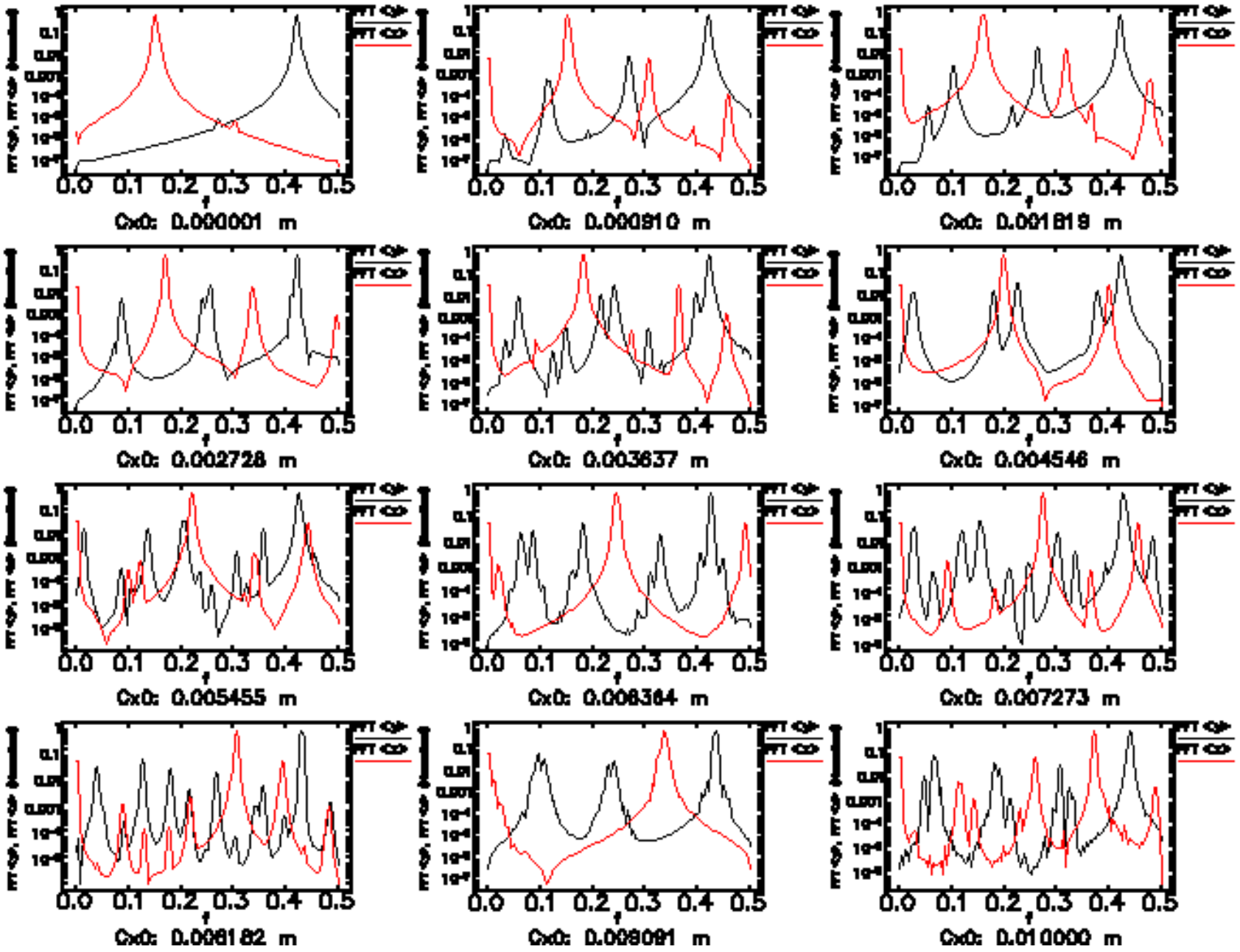
Order	CxFrequencyDeriv	CyFrequencyDeriv
0	1.51E-001	4.20E-001
1	-4.40E-002	6.97E-002
2	3.59E+002	1.60E+002
3	4.17E+004	-4.59E+003

- Set dipole edges to order=1 for non-chromatic terms
- Differences in x due to having tune in upper half plane

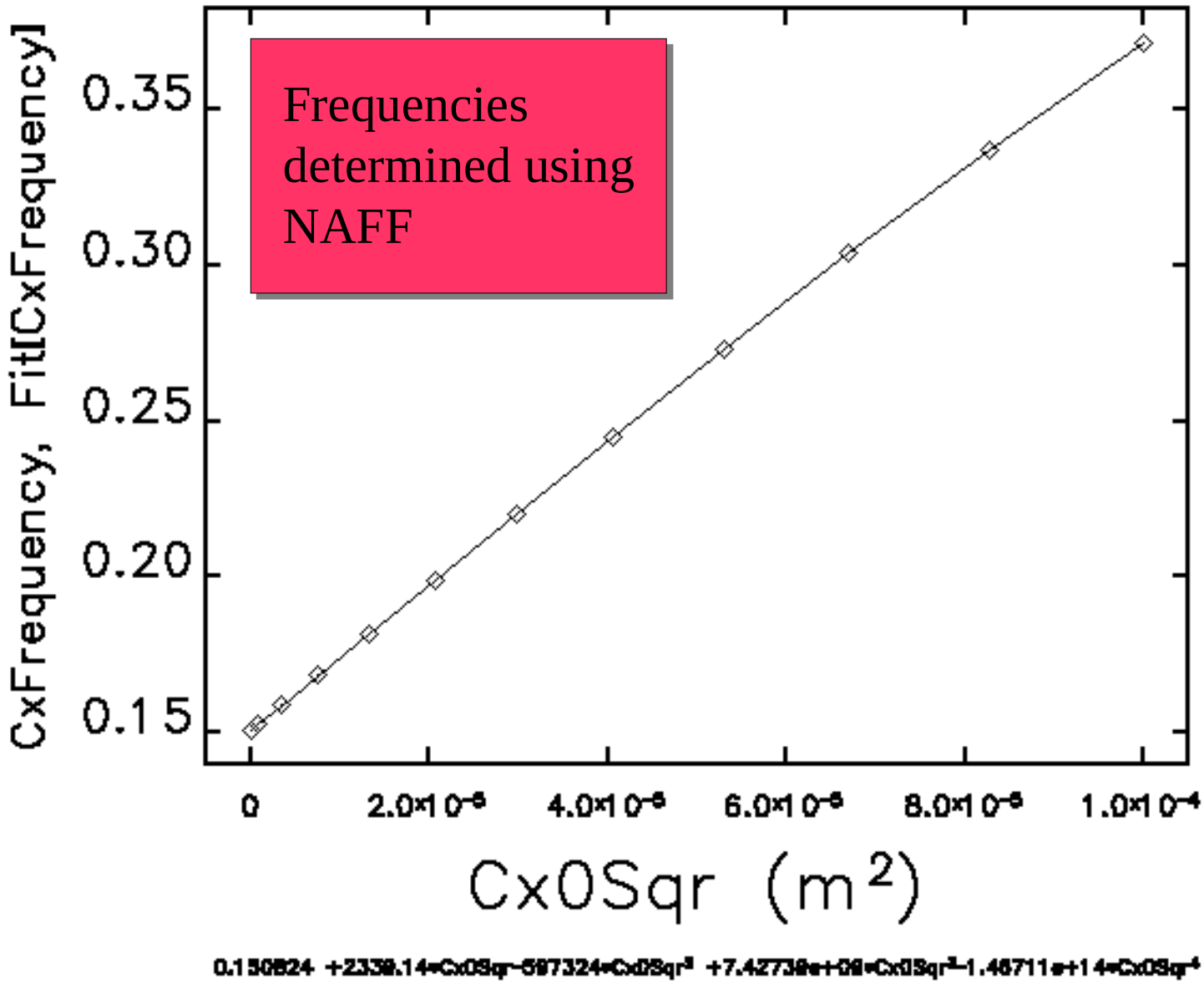
On-Momentum Phase Space Tracking



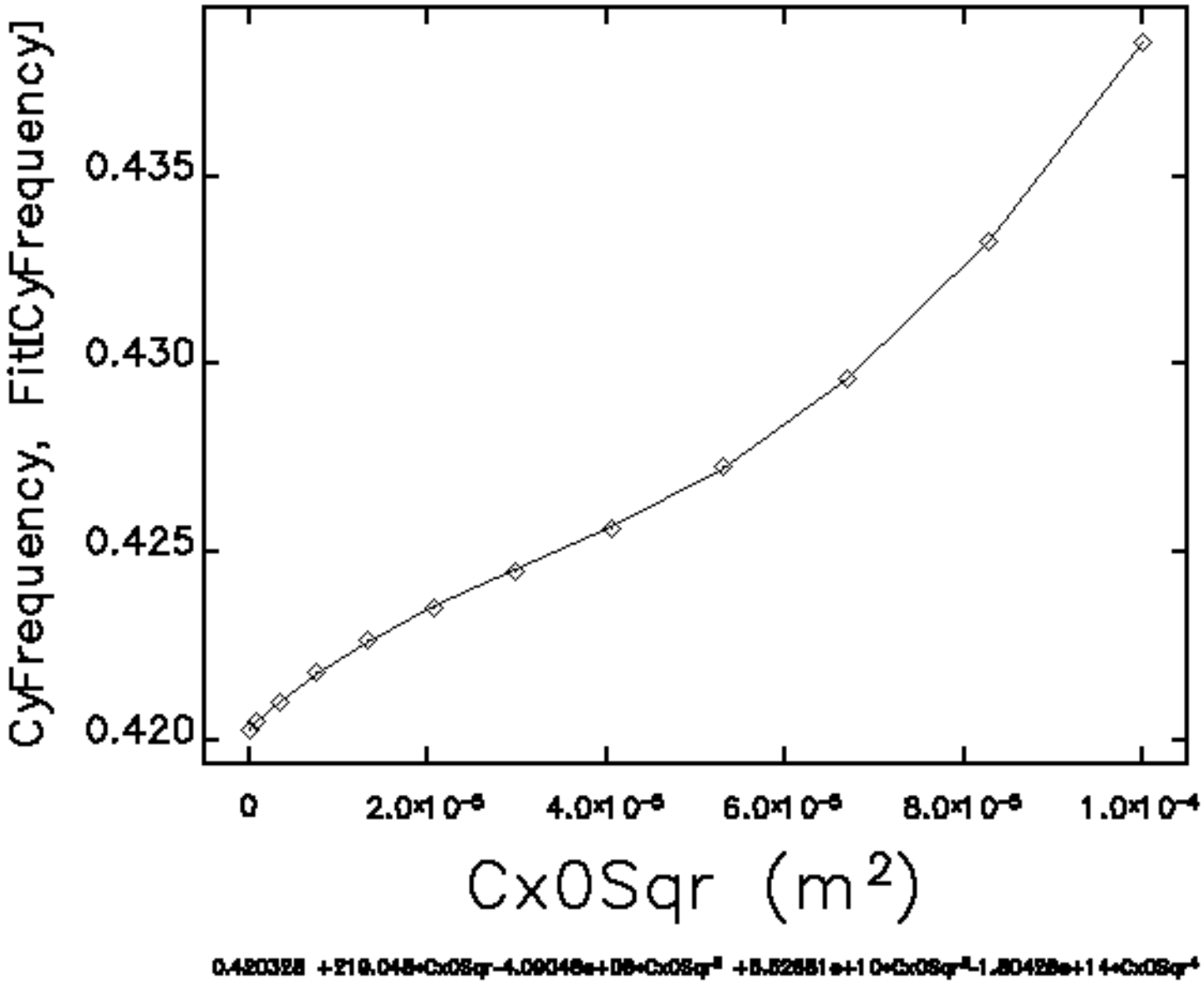
On-Momentum Phase Space Tracking



On-Momentum Phase Space Tracking



On-Momentum Phase Space Tracking



On-Momentum Phase Space Tracking

Fit terms for tune shift with horizontal offset squared

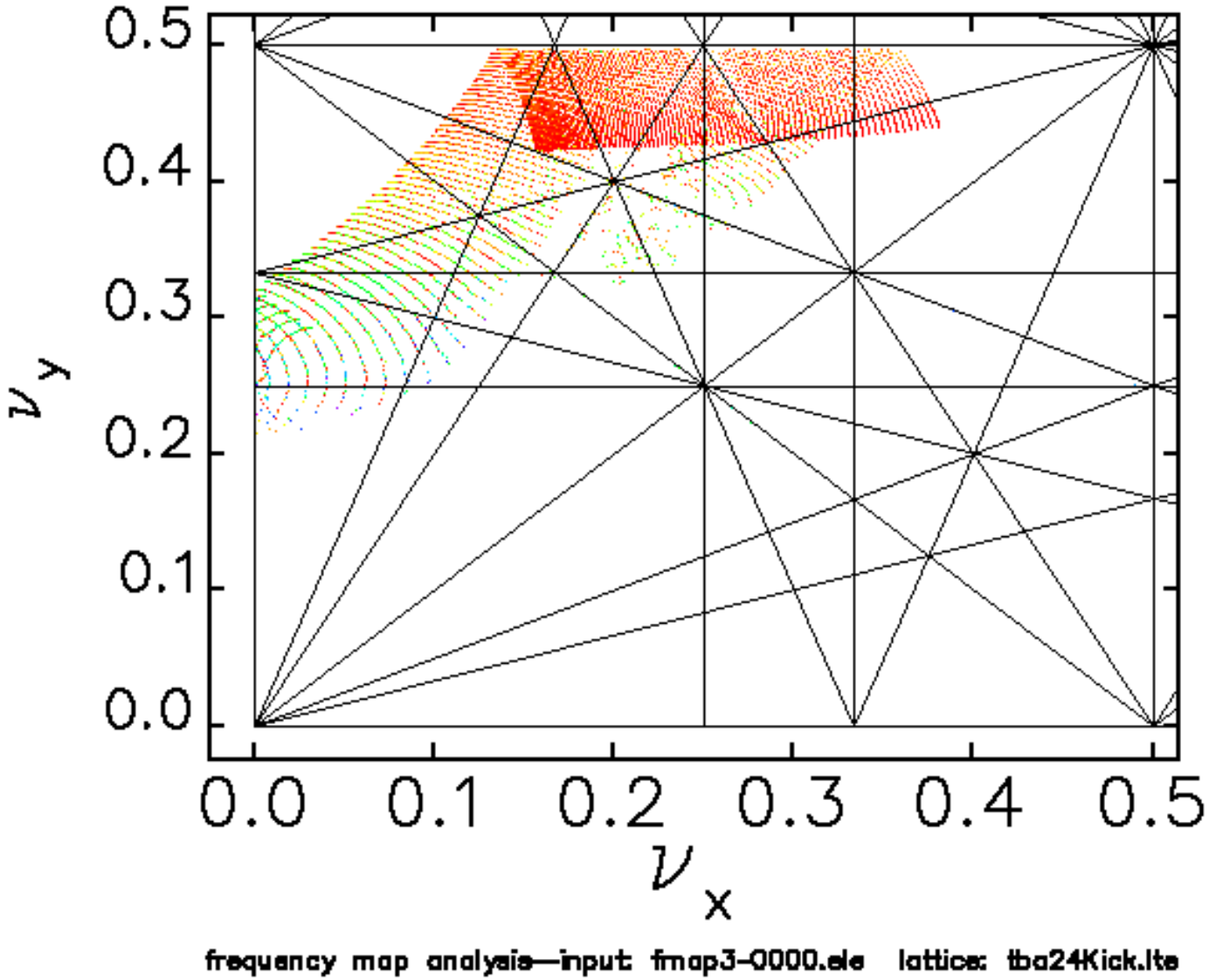
Order	CyFrequencyCoefficient	CxFrequencyCoefficient
0	4.20E-001	1.51E-001
1	2.19E+002	2.34E+003
2	-4.09E+006	-5.97E+005
3	5.53E+010	7.43E+009
4	-1.80E+014	-1.47E+014

- elegant also uses matrix methods to get the leading terms

dnuy/dx0Sqr	dnux/dx0Sqr
1.82E+002	-2.54E+003

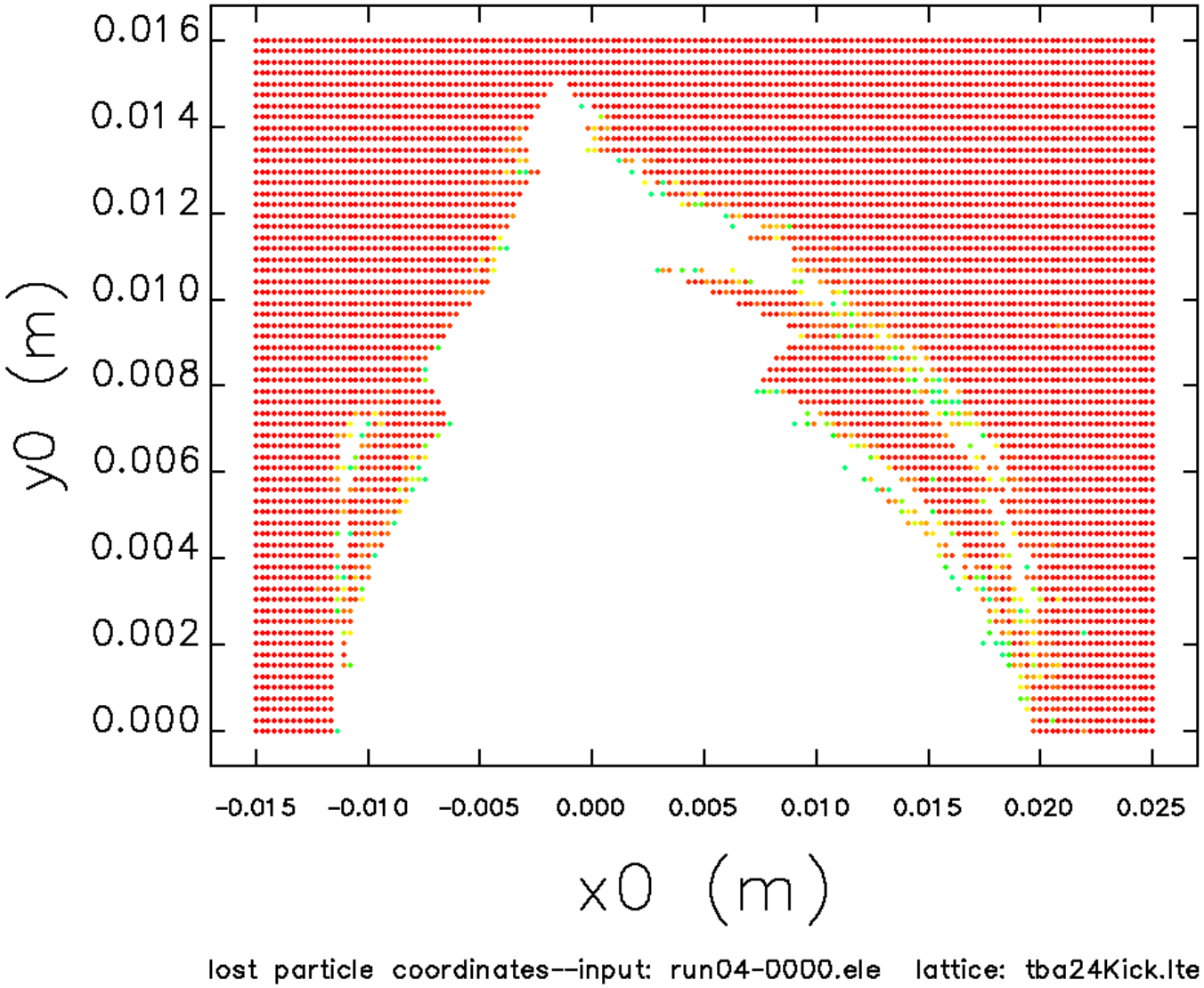
- Sign difference for x tune is due to the tune being in the upper half plane

Frequency Map Analysis



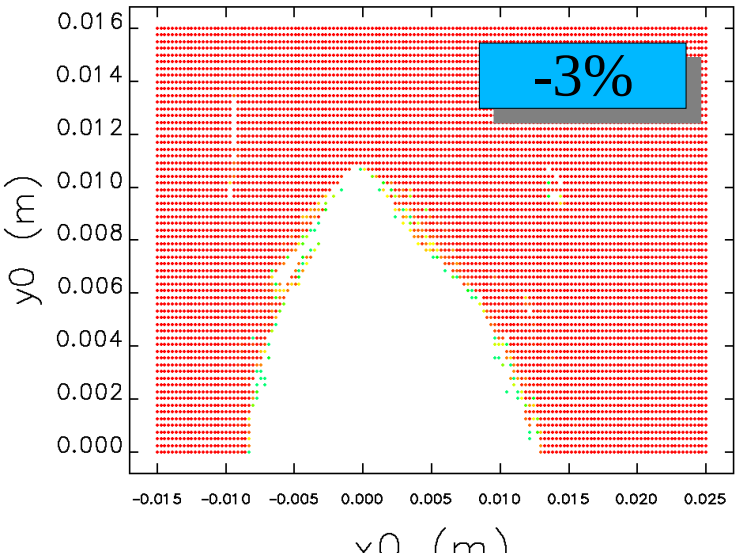
- 1024-turn tracking, 15 kicks per element
- Color code shows change in tune
- 100 independent jobs, results collated with SDDS script
- Used ~50, 1.8 GHz CPUs simultaneously
- Took ~27 minutes

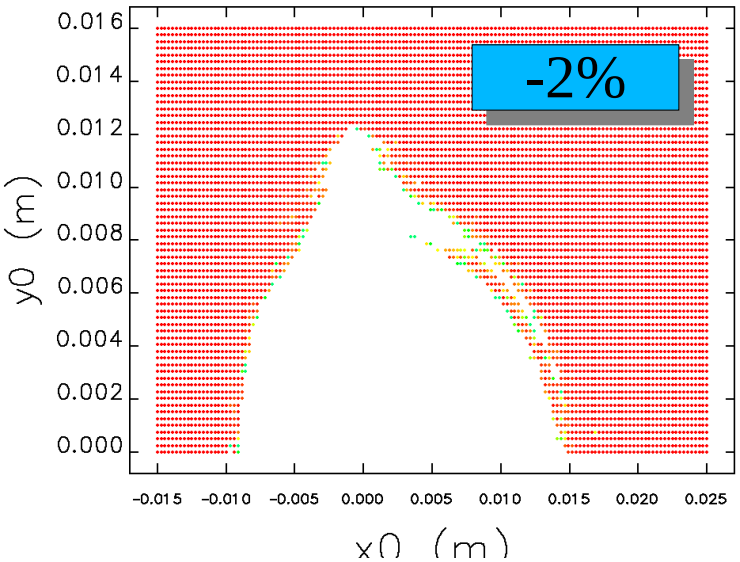
Dynamic Aperture

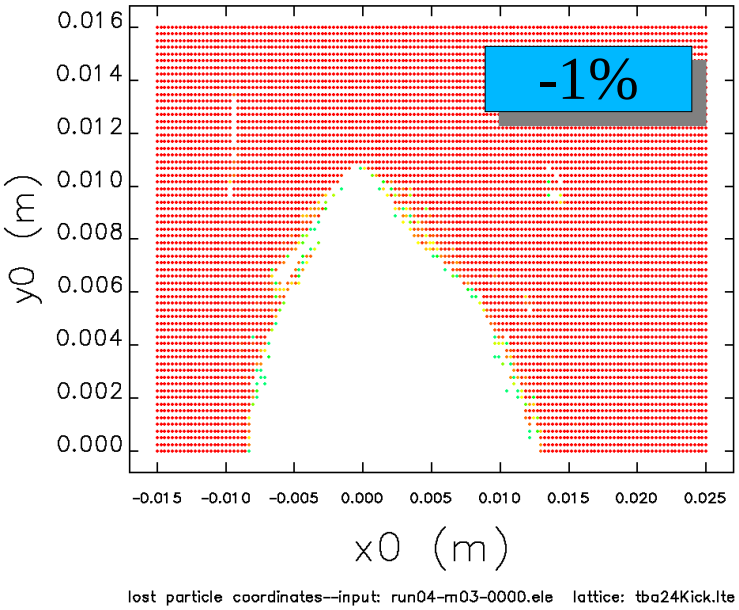


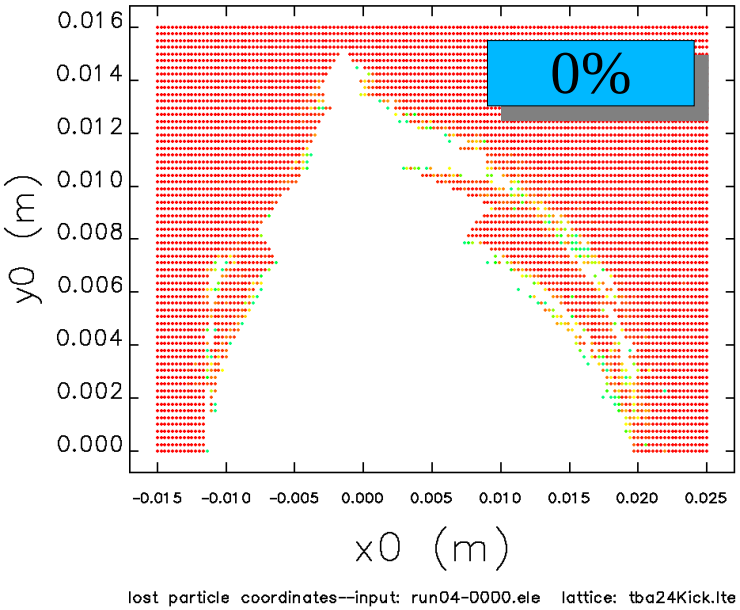
- 500-turn tracking, 15 kicks per element
- Color code shows turns survived
- 144 independent jobs, results collated with SDDS script
- Used ~50, 1.8 GHz CPUs simultaneously
- Took ~7 minutes

Off-Momentum Dynamic Aperture

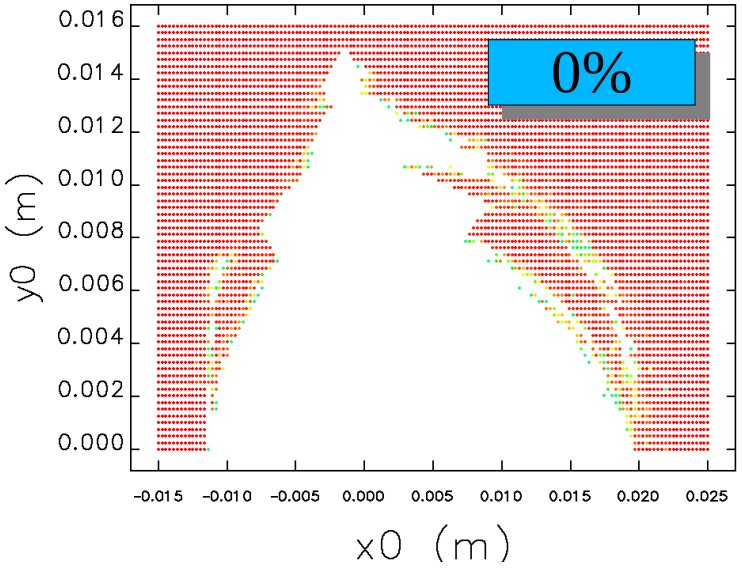


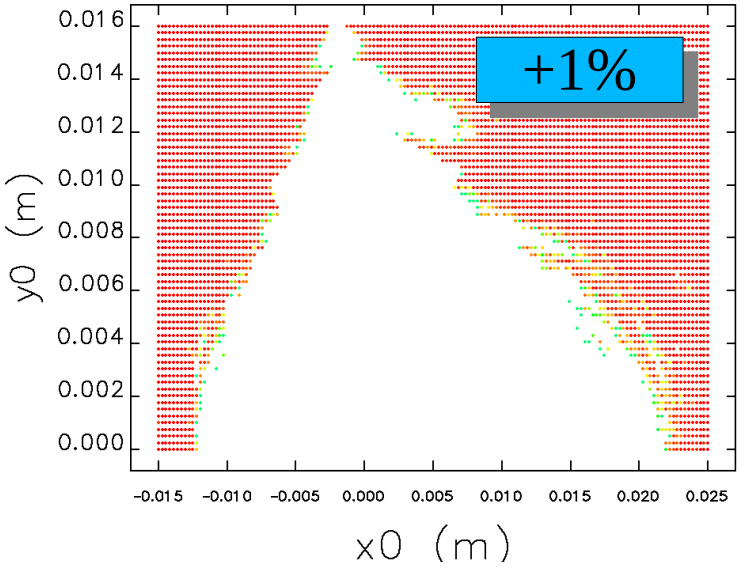


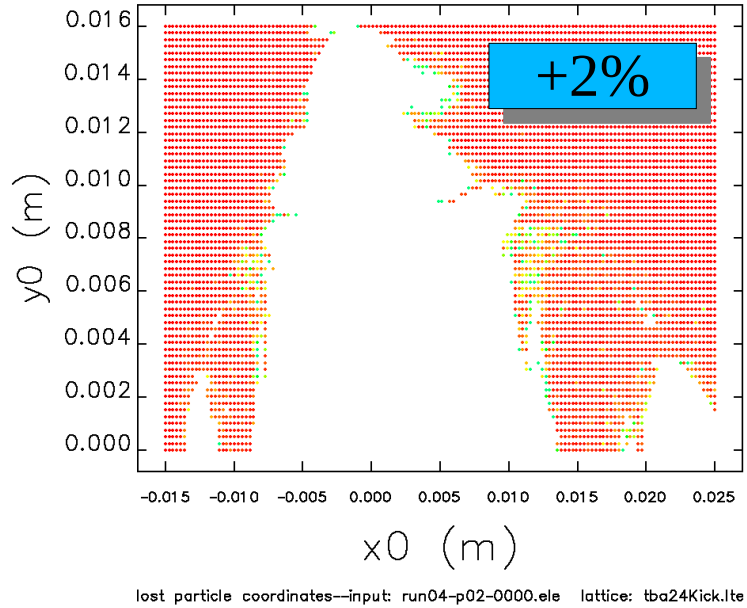


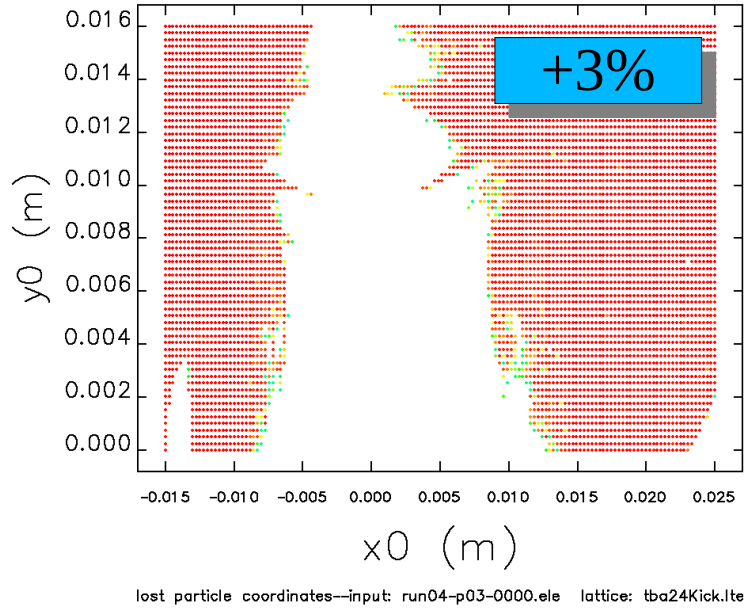


Off-Momentum Dynamic Aperture









Recommended Simulations

- Minimize the tune-shifts with amplitude while keeping the chromaticity fixed
  - Add higher-order TSWA to elegant if needed
- Try direct optimization of DA
  - Use “hollow” beam and maximize number of survivors
  - Try geneticOptimizer (cluster-based)
- Dynamic aperture with errors and vertical apertures
- Injection simulations with errors and vertical apertures
- Tracking with impedances and HOMs