Genesis 1.3 V4

A friendly introduction

Eugenio Ferrari

A foreword

Genesis 1.3 V4 is in a stable phase, although still actively developed.

Sven is constantly adding new functionality and new bugs, but things generally work.

If something is fishy, please reach out.

https://github.com/svenreiche/Genesis-1.3-Version4

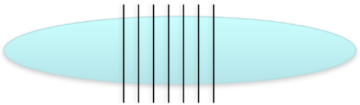
Reference system:

- z is the longitudinal coordinate, along the undulator axis.
  (z = 0 is the undulator entrance)
- s is the (internal) beam coordinate.
  (s = 0 is the tail of the beam)

Units: MKS. The calculated spectrum is in eV.

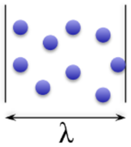
Introduction

Electron Beam (~50 μm)



Co-moving frame:

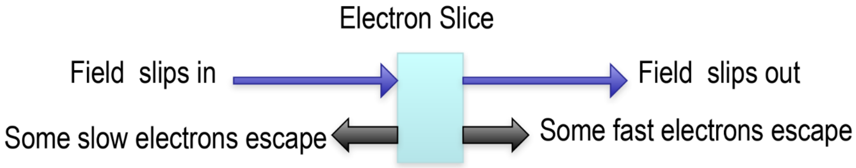
Electron Slice (~1 Å)



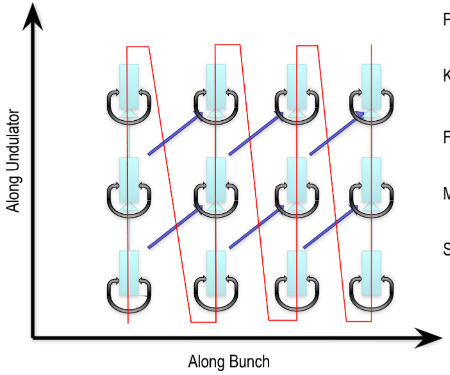
Slice thickness λ defines reference wavelength, which is not necessarily the resonant wavelength. Though both should be close to avoid strong drifts in slice:

Two step algorithm:

- Advance radiation field (diffraction + emission by electrons)
- Advance electrons (interaction with field and change in ponderomotive phase)



Quasi-time dependent simulations



Propagate Field to Next Slice

Keep Slice in position

Feed electrons back into slice

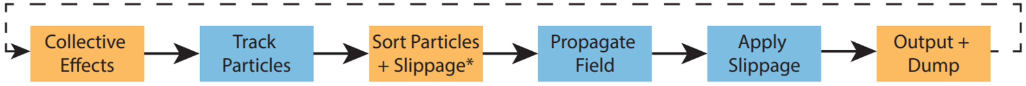
Model of chained amplifiers

Simulation can crawl through bunch:

- Inner loop: undulator
- Outer loop: bunch

Core Algorithm

Basic Leapfrog Step:



🟧 *Optional or occasional applied*

🟦 *Executed at each step*

**If more than 50% would move, field is shifted in opposite direction*

Reference particle tracking defines if slippage is applied

Undulator Break Section:

Particles:

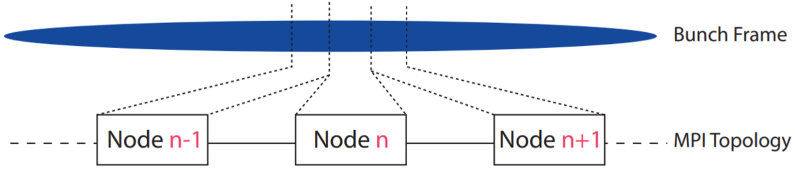
- 6D Matrix Formalism with arbitrary steps
- No Runga-Kutte integration
- Particles can move outside grid (e.g. chicane)

Field (different solvers):

- ADI (Alternating Direction Implicit) Solver
- 2D FFT → Greens Function (non-dispersive) → 2D iFFT
- 3D FFT → Greens Function (dispersive) → 3D iFFT

Green's function approach allows the implementation of self-seeding in a single run

Memory Management



Entire bunch and field is kept in memory.

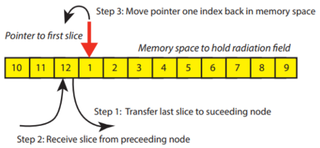
Each node holds multiple adjacent slices.

Resolving each electron (if needed)

200 pC	56 GByte
1 Å (50 fs, 151^2 grid)	51 GBye
Total	**107 GByte**

Field slippage:

Inter-Node (MPI):
one wavefront per node
Intra-Node (C++-Vector):
Increment of Index pointer



Step 3: Move pointer one index back in memory space

Pointer to first slice

Memory space to hold radiation field

Step 1: Transfer last slice to suceeding node

Step 2: Receive slice from preceeding node

Personal suggestions about tools

Having the right tools helps in simplifying life from the start.

A good txt editor.

Associate the .in and .lat files to open directly into it.

Common pitfall: carriage return convention in Windows (“\r\n”) and Unix/Linux (“\n”). Search the option to use the Linux ones depending on the cluster (I lost more time than I want to admit due to this specific problem)

A hdf5 explorer (example: hdfView)

To access directly the output files

Matlab if you want to use xgenesis postprocessor

Python/Jupyter if you like to live dangerously.

Input files

Compulsory:

- Main input file, where you prepare the commands for your simulation
- Lattice file, that describes the components of your beamline

These are “normal” text files following some conventions (see Manual for full description).

Other files that can be needed:

- Description files, e.g. for the e-beam distribution
- Distribution files, coming from another Genesis 1.3 simulation, Elegant or others
- Scripts

Description files are normally hdf5 files, distributions are either hdf5 or sdds files.

Lattice file

Description of the beamline and its components.

General syntax:

```
label: element type = { parameter = value [, · · ·]};
```

Common mistakes:

- The line must end with ;
- Don’t put a comma (,) at the end of the list of parameters, just before the “}”
- Comments should be put on dedicated lines (not inline). A line starting with ! is ignored.

Possible element types:

undulator, quadrupole, drift, corrector, chicane, phaseshifter, marker and line

Font case is not important (upper or lower is ignored).

Lattice file

Undulator:

- **aw**: The dimensionless rms undulator parameter.
- **lambdau**: Undulator period length in meter.
- **nwig**: Number of periods
- **helical**: T/F flag whether the undulator is planar or helical. (Default false). Note that setting it to true, does not change the roll-off parameters, that needs to be explicitly defined to be consistent.
- **kx**, **ky**: Roll-off parameter of the quadratic term of the undulator field in x, y. (Default is x=0, y=1. Undulator focusing)
- **ax**, **ay**: Offset of the undulator module in meters, with respect to the beamline axis. (to study misalignment effects)
- **gradx**, **grady**: Relative transverse gradient of undulator.

Notes I would have liked to know from the begin:

- If you work with circularly polarized undulators and need to take the focusing correctly into account, explicitly write kx=0.5, ky=0.5
- To maintain the resonance between a circular and a linear undulator, aw is the same.

Example: MOD2: UNDULATOR = { lambdau = 0.0826, nwig=30, aw=6.9775 };

Lattice file

Drift:

- **l**: The length of the drift (default 0).

Notes I would have liked to know from the begin:

- Now a line can start with a drift and can also be composed of only a drift.

Quadrupole:

- **l**: Length of the quadrupole in meter. Default 0.
- **k1**: Normalized focusing strength in 1/m^2 . Default 0. No conversions, it’s exactly the K of the quadrupole.
- **dx**, **dy**: Offsets in meters. Default 0, to study misalignments.

Lattice file

Corrector:

- **l**: Length of the corrector in meter. Default value is 0 m.
- **cx**, **cy**: Kick angle in x and y in units of $\gamma\beta_{(x,y)}$.

Phaseshifter:

- **l**: Length of the phase shifter. Default is 0 m.
- **phi**: Change in the ponderomotive phase of the electrons in rad. Default value is 0. Note that Genesis is doing an autophasing, so that the electrons at reference energy are not changing in ponderomotive phase in drifts.

Marker:

- **dumpfield**: to dump of the field distribution at the location of the marker. (0/1, warning: big files)
- **dumpbeam**: to dump of the particle distribution. (0/1, warning: big files)
- **sort**: to start the sorting of particles (only for one-for-one simulations).
- **stop**: To stop the tracking module. The output file still contains the full length with zeros after the stop location.

Lattice file

Chicane:

4-dipole chicane

- **l**: Length of the chicane. The first and last are placed at the beginning and end of the reserved space. The inner ones are defined by the drift length in between. Any remaining distance, (l - 4lb - 2ld) is placed between the second and third dipole. Default value 0 m.
- **lb**: Length of an individual dipole in meter. Default value is 0 m
- **ld**: Drift between the outer and inner dipoles, projected onto the undulator axis. The actual path length is longer by the factor 1/cosθ, where θ is the bending angle of an individual dipole.
- **delay**: Path length difference between the straight path and the actual trajectory in meters. From this value the bending angle is calculated internally by Genesis.

Notes:

- Focusing is present, CSR is not
- R_{56} = 2 delay

Lattice file

Line:

- To order the elements along the propagation of the electrons and build the beamline.
- Nested lines are possible (supported till 10 levels).
- Elements can be repeated several times, if preceded by a multiplication sign.
- Examples:
  - FODO: Line = {F, DRIFT, D, DRIFT};
  - LAT: Line = {6*FODO};
- Placement:
  - Using the symbol “@” after a label places that element at a specified position along the beamline, in m.
  - Example:
    - FODO: Line = {UND, F@0.2, D@1.2};
    - places the F quadrupole at location 0.2 m and the D at location 1.2 m
  - Otherwise, sequential placing according to the length of the different elements and eventual padding drifts.
- Elements can be superimposed to the undulator field.
  - Example:
    - UND: Undulator = {aw=1, lambdau=0.02, nwig=100};
    - FODO: Line = {UND, F@0.2, D@1.2};
    - The F and D quadupoles are superimposed to the undulator.

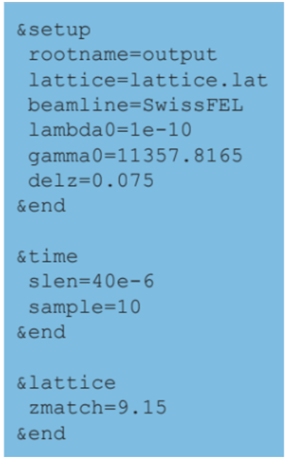
Main input file

Here it’s where the fun stuff happens.

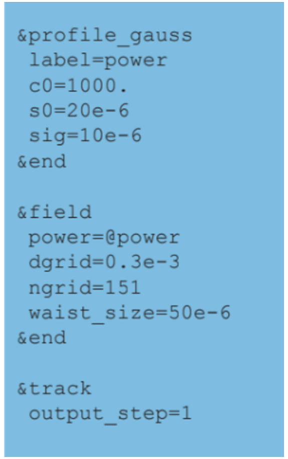
The new version has a syntax similar to elegant, with namelists starting with “&” and finishing with “&end”.

Multiple runs are possible in the same input.

Example:



```
&setup
 rootname=output
 lattice=lattice.lat
 beamline=SwissFEL
 lambda0=1e-10
 gamma0=11357.8165
 delz=0.075
&end

&time
 slen=40e-6
 sample=10
&end

&lattice
 zmatch=9.15
&end
```



```
&profile_gauss
 label=power
 c0=1000.
 s0=20e-6
 sig=10e-6
&end

&field
 power=@power
 dgrid=0.3e-3
 ngrid=151
 waist_size=50e-6
&end

&track
 output_step=1
```

Supported commands

- **setup**: mandatory and it must be at the top of the input. Can be called only once. It is used to define the basic parameters of the simulation.
  - New in latest version:
    - (beam/field)_glob_stat: extra output for the entire bunch, at each step (flags that are false as default)
    - exclude_(spatial/fft/intensity/energy/aux/current)_output: to save space in the output file
    - exclude_field_dump: to have only the total field in the dump file, not the single slices.

- **alter_setup**: to change the parameters of the simulation, e.g. for an harmonic or subharmonic jump.

- **lattice**: to alter the parameters of the lattice.

- **time**: to define the temporal window of the simulation.

- **profiles**: a series of utilities to define any non-constant property (more details later).

- **beam**: electron beam definition.

- **field**: radiation field definition.

Supported commands

- **importdistribution**: to import an Elegant distribution
- **importbeam**: to import a Genesis 1.3 beam distribution
- **importfield**: to import a Genesis 1.3 field distribution
- **efield**: space charge definition
- **sponrad**: takes account of the spontaneous radiation outside the FEL bandwidth
- **wake**: setup for the wakefield calculations
- **sort**: initiates the sorting of the particle distribution
- **write**: dump the field/beam distribution
- **stop**: to stop the execution of the input file at the location of the command.

Supported commands

- **track**: to start the simulation.
    - Useful options:
        - output_step, (field/beam)_dump_step, sort_step (flags): after how many integrations steps the corresponding action is taken
    - New in latest version:
        - field_dump_at_undexit: whether to perform a dump of the field at the exit of each undulator module

Profiles

Utilities to define any non-constant property

Profile definition (first):

```
&profile_gauss
        label=power
        c0=1000.
        s0=20e-6
        sig=10e-6
&end
```

then use it:

```
&field
        power=@power
        dgrid=0.3e-3
        ngrid=151
        waist_size=50e-6
&end
```

Available profiles:

- profile_constant: a constant value.
- profile_gaussian
- profile_step: constant value between a start and an end
- profile_polynom: up to 4th degree
- profile_file: coming from an hdf5 file where arbitrary profiles are defined

Remember:

- the horizontal scale is in meters (except for profile_file that has an option to specify the s-coordinate in seconds)

Time window for the simulation

Single slice (time independent simulations) can be done by putting time = 0 (in the &time command, I know, a bit redundant, but that’s life).

The convention is that the electrons are staying in the time window, and the radiation field is slipping ahead of it.

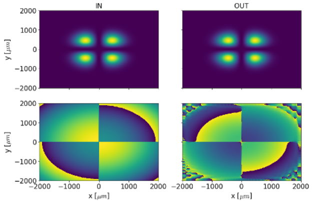
What I would have liked to know from the beginning:

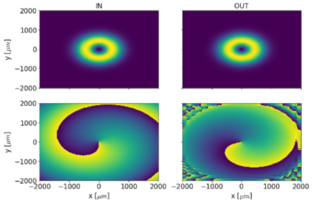
- when using chicanes, it’s possible that the radiation slips ahead by a lot.
- Instead of having a huge time window for the simulation, it’s usually convenient to place the chicane at the end of a beamline, or in a separated tracking.

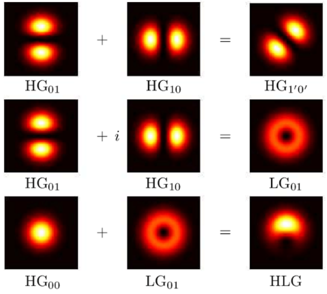
Field

Genesis 1.3 describes the field on a 2D grid.
The grid size needs to be specified, using the dgrid (extension from the center to one edge) and ngrid (number of grid points, in one dimension). Note: ngrid better be odd, to have the (0, 0) point.

Can be defined multiple times along the input file: if the accumulate option is active, the different fields are superimposed one to the other.

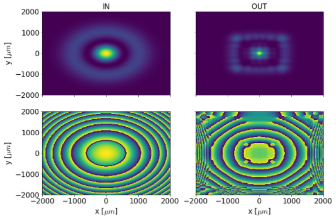






HG$_{01}$ + HG$_{10}$ = HG$_{1'0'}$
HG$_{01}$ + i HG$_{10}$ = LG$_{01}$
HG$_{00}$ + LG$_{01}$ = HLG

Warning: define a large enough grid,
otherwise you will get "numerical" reflections.



It's also possible to define the field on the harmonics.

Output file(s)

Hdf5 file with the following structure:

Main output

/Global

Main input parameters

/Lattice

The magnetic lattice

/Beam

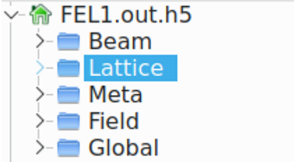
All the beam parameters (initial ones also)

/Field

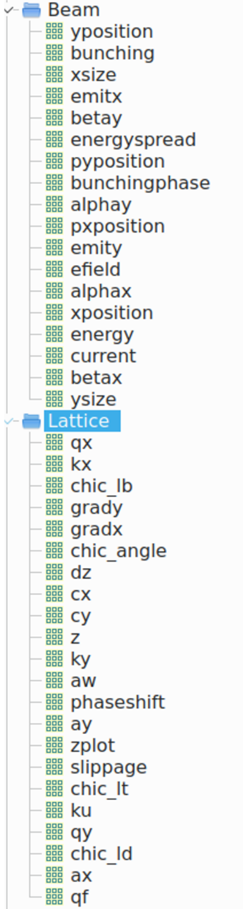
Field parameters

/Meta

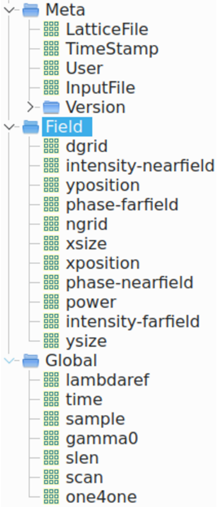
Information (including the input deck)



```
FEL1.out.h5
  Beam
  Lattice
  Meta
  Field
  Global
```



```
Beam
  yposition
  bunching
  xsize
  emitx
  betay
  energyspread
  pyposition
  bunchingphase
  alphay
  pxposition
  emity
  efield
  alphax
  xposition
  energy
  current
  betax
  ysize
Lattice
  qx
  kx
  chic_lb
  grady
  gradx
  chic_angle
  dz
  cx
  cy
  z
  ky
  aw
  phaseshift
  ay
  zplot
  slippage
  chic_lt
  ku
  qy
  chic_ld
  ax
  qf
```



```
Meta
  LatticeFile
  TimeStamp
  User
  InputFile
  Version
Field
  dgrid
  intensity-nearfield
  yposition
  phase-farfield
  ngrid
  xsize
  xposition
  phase-nearfield
  power
  intensity-farfield
  ysize
Global
  lambdaref
  time
  sample
  gamma0
  slen
  scan
  one4one
```

Output file(s)

Hdf5 file with the following structure:

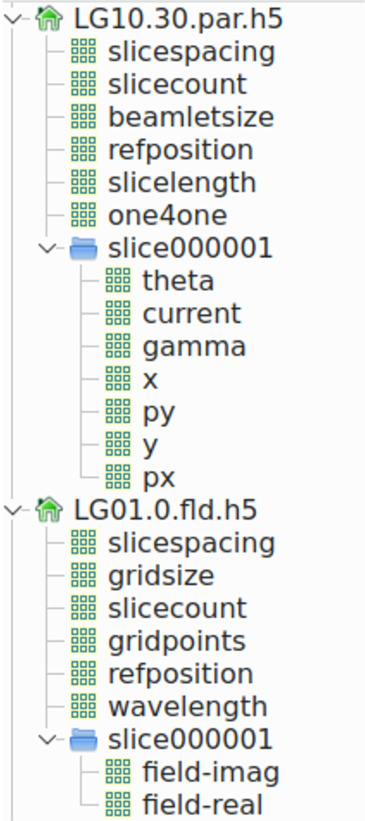
Beam and field:

Global parameters

slice000000i/

Parameters about that particular slice

(.par are beam files, .fld are field files)



- LG10.30.par.h5
  - slicespacing
  - slicecount
  - beamletsize
  - refposition
  - slicelength
  - one4one
  - slice000001
    - theta
    - current
    - gamma
    - x
    - py
    - y
    - px
- LG01.0.fld.h5
  - slicespacing
  - gridsize
  - slicecount
  - gridpoints
  - refposition
  - wavelength
  - slice000001
    - field-imag
    - field-real

Postprocessor: xgenesis

Matlab based (easier than in the past):

- xgeninit: function that takes the output file name as argument and initializes the reading

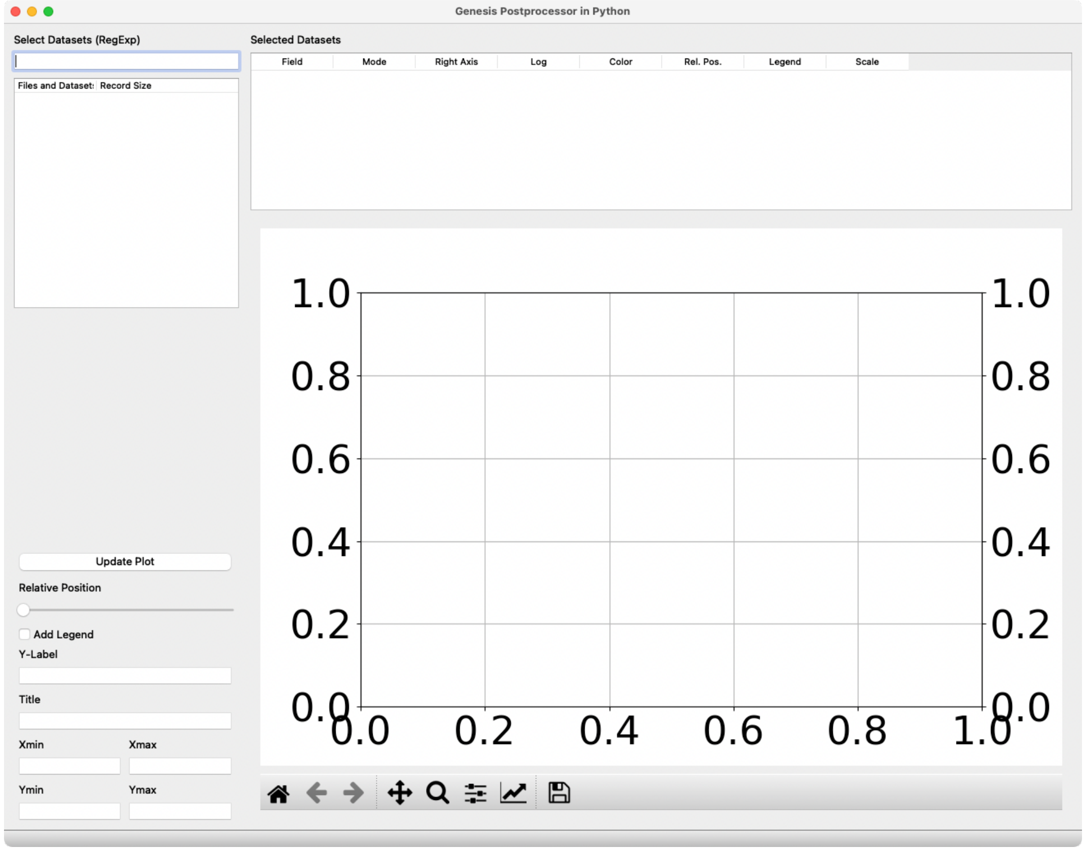
- xgenplot: to plot any of the quantities present in the output file, plus the spectrum. It has different options to plot profiles along the undulator, along the bunch or 2d plots.
  - Example:
    - *xgenplot('power', 'normal', 3e-6)* plots the power at 3 um in the bunch frame, as a function of the undulator coordinate.
    - *xgenplot('power', 'profile', 11.5)* plots the power profile at 11.5 m in the undulator as a function of the bunch internal coordinate.
  - Plot modes:
    - 'normal' - along undulator, requires s-position as 'ref'
    - 'profile' - along beam frame, requires z-position as 'ref'
    - 'mean' - along undulator, using the mean value at each step
    - 'max' - along undulator, using the max value at each step
    - 'rms' - along undulator, RMS value only useful for power and spectrum
    - 'weighted' - along the undulator, using a weighted mean calculation
    - '2d' - Image along beam frame in x and along undulator in y
    - '2dnorm' - same as '2d' but normalized at each step along the undulator

- xgenwigner: to calculate the 2D Wigner distribution.

Postprocessor: PyGenesis

Python based gui for the output.

Available in a separate github repository
https://github.com/svenreiche/PyGenesis



Example: HGHG simulation

Main input (FERMI.in):

```
&setup
rootname=FEL1
lattice=FERMI.lat
beamline=FEL1_MOD
lambda0=260e-09
gamma0=2.93542690E+03
delz=0.1
shotnoise=1
one4one=true
&end

&time
#time=0
slen=120e-6
sample=1
&end

&profile_gauss
label=prof1
c0=2.e7
#c0=150.e6
s0=60e-6
sig=17.7e-6
&end

&field
power=@prof1
dgrid=2.000000e-3
ngrid=301
waist_size=600e-6
waist_pos=1.5
&end
```

```
&beam
betax=4.708539
alphax=-0.7011887
betay=13.490464
alphay=0.6774517
current=700
delgam=0.1957
ex=1.000000e-06
ey=1.000000e-06
&end

&track
&end

&sort
&end

&alter_setup
beamline=FEL1_RAD
harmonic=13
&end

&field
power=0
dgrid=0.500000e-3
ngrid=301
waist_size=100e-6
&end

&track
&end
```

Lattice file (FERMI.lat):

```
FEL1_MOD: LINE = {MOD1, D1, DS1, D2, QM1, DD1, QR1, DD2};

 FEL1_RAD: LINE = {RAD1, DD1, QR2, DD2,
                   RAD2, DD1, QR3, DD2,
                   RAD3, DD1, QR4, DD2,
                   RAD4, DD1, QR5, DD2,
                   RAD5, DD1, QR6, DD2,
                   RAD6};
```

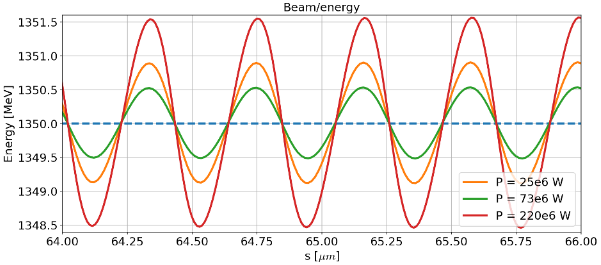
one4one:
Flag to enable or disable simulation to resolve each electron in the simulation. This is mandatory for certain features, such as sorting or slicing of particle distributions

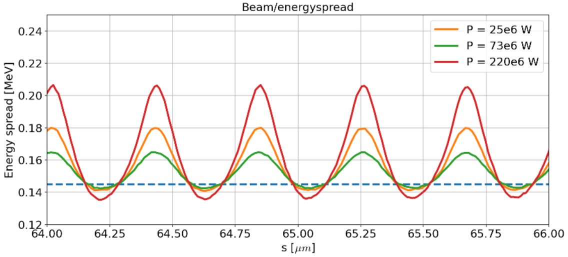
sort:
It initiates the sorting and redistribution of particles only if one-for-one simulation is enabled.
Note that harmonic conversion will automatically invokes sorting.

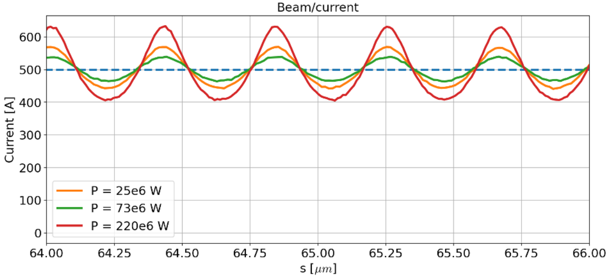
Example: using a profile_file

Seed1: 413 nm, seed2: 294 nm

```
# energy
&profile_file
label=gamma
xdata=beam_after_mod1.h5/s
ydata=beam_after_mod1.h5/gamma
&end

# energy spread
&profile_file
label=delgamma
xdata=beam_after_mod1.h5/s
ydata=beam_after_mod1.h5/delgamma
&end

# current
&profile_file
label=current
xdata=beam_after_mod1.h5/s
ydata=beam_after_mod1.h5/current
&end

&beam
current=@current
gamma=@gamma
delgam=@delgamma
...
&end
```


Beam/energy
Energy [MeV]
s [μm]
P = 25e6 W
P = 73e6 W
P = 220e6 W


Beam/energyspread
Energy spread [MeV]
s [μm]
P = 25e6 W
P = 73e6 W
P = 220e6 W


Beam/current
Current [A]
s [μm]
P = 25e6 W
P = 73e6 W
P = 220e6 W