Adv. IMPACT Documentation

Release 1.0.0

Kei Fukushima

Jun 22, 2018

TABLE OF CONTENTS:

1 Installation **3**
- 1.1 Build from source code . . . 3

2 Tutorial **5**
- 2.1 1. Basic usage . . . 5
- 2.2 2. Lattice parameter control . . . 6
- 2.3 3. Example: Phase scan . . . 8
- 2.4 4. Beam parameter control . . . 9
- 2.5 5. History result usage . . . 10
- 2.6 6. Distribution result usage . . . 11
- 2.7 7. A to B run (partial run) . . . 12
- 2.8 8. Example: Transverse matching . . . 13
- 2.9 9. Lattice construction . . . 13
- 2.10 10. Data export . . . 14

3 Parallel Computing **15**
- 3.1 MPI usage in command line . . . 15
- 3.2 MPI usage in jupyter-notebook . . . 15
- 3.3 Tips . . . 17

4 Lattice Elements **19**
- 4.1 Optical element . . . 19
- 4.2 Special element . . . 30
- 4.3 Synchronous phase input for RF cavities . . . 35
- 4.4 External data format . . . 36

5 Class Library **39**
- 5.1 Sequence class – run control API . . . 39
- 5.2 Input class / base of Sequence class . . . 43
- 5.3 Result class / base of Sequence class . . . 51

6 Indices and tables **79**

Python Module Index **81**

Index **83**

Advanced IMPACT - Integrated Map and Particle Accelerator Tracking Code

Advanced IMPACT is refactored code from IMPACT-Z which developed in LBNL.

Remarkable Features

- Particle tracking with multiple charge states
- Python interface (include MPI and ipython-notebook)
- Lattice construction from Python interface
- Iterative run with lattice parameter controller
- Result management for multiple charge states
- Multiple field data caching on memory
- Unit-test framework for beam dynamics
- Build manage by CMake

CHAPTER

ONE

INSTALLATION

1.1 Build from source code

1.1.1 Pre-requisites

(may need to apt-get with sudo)

For python2.x,

```
$ apt-get install libopenmpi-dev python-scipy python-nose python-mpi4py cmake
```

For python3.x,

```
$ apt-get install libopenmpi-dev python3-scipy python3-nose python3-mpi4py cmake
```

mpi library for fortran source, also scipy and mpi4py for Python interface. The nosetests test runner is used for 'make test' if present.

1.1.2 Building

Git clone from impact repository

```
$ git clone *repository-address*
```

If you are in Master branch, change branch to dev-adv.

```
$ cd impact
$ git checkout dev-adv
```

Make `build` directory and compile with CMake.

```
$ mkdir build
$ cd build
$ cmake ..
$ make
```

Beam dynamics test for IMPACT.

```
$ make test
```

Install with proper permissions.

```
$ make install # may need to install with sudo
```

CHAPTER

TWO

TUTORIAL

2.1 1. Basic usage

In Python interface (includes IPython-notebook), user can import IMPACT *Sequence* class.

```
>>> from impact import Sequence
```

Create IMPACT *Sequence* object with input file.

```
>>> sq = Sequence('lattice_1cavity-1solenoid')
```

If user's working directory has following structure for the external data files,

```
$ ls
data/  lattice_1cavity-1solenoid
$ ls data/
1T421.T7  1T861.T7  fort.885  partcl.data
```

user can define *subdir* for the data directory.

```
>>> sq.subdir = 'data'
```

Load the 3D field data. – *load*

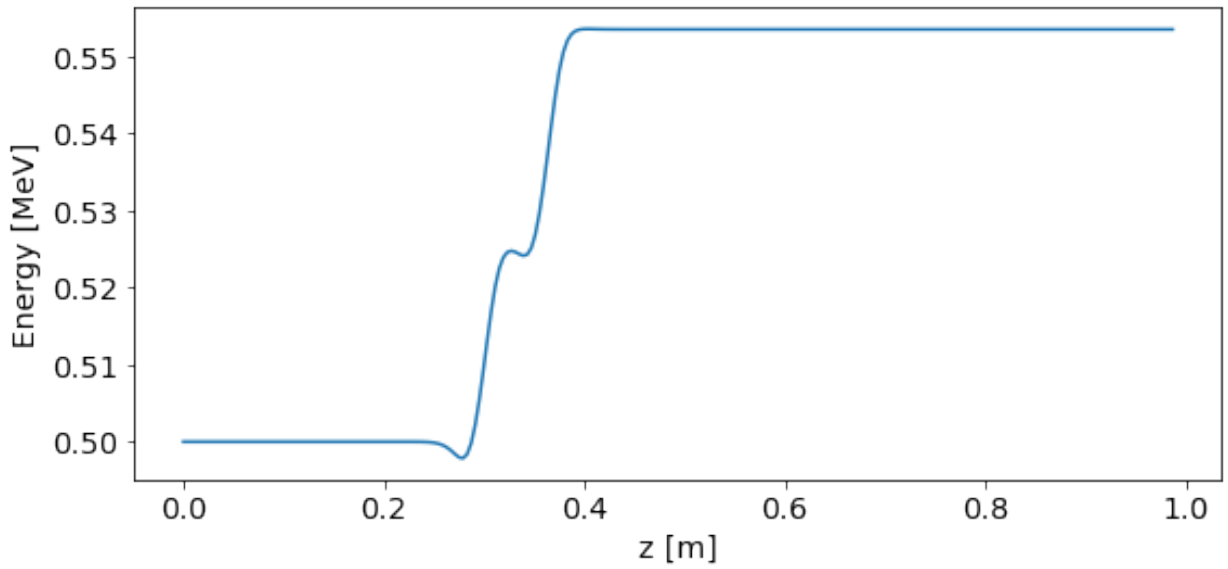
```
>>> sq.load()
```

Run particle tracking simulation. – *distribute*, *run*

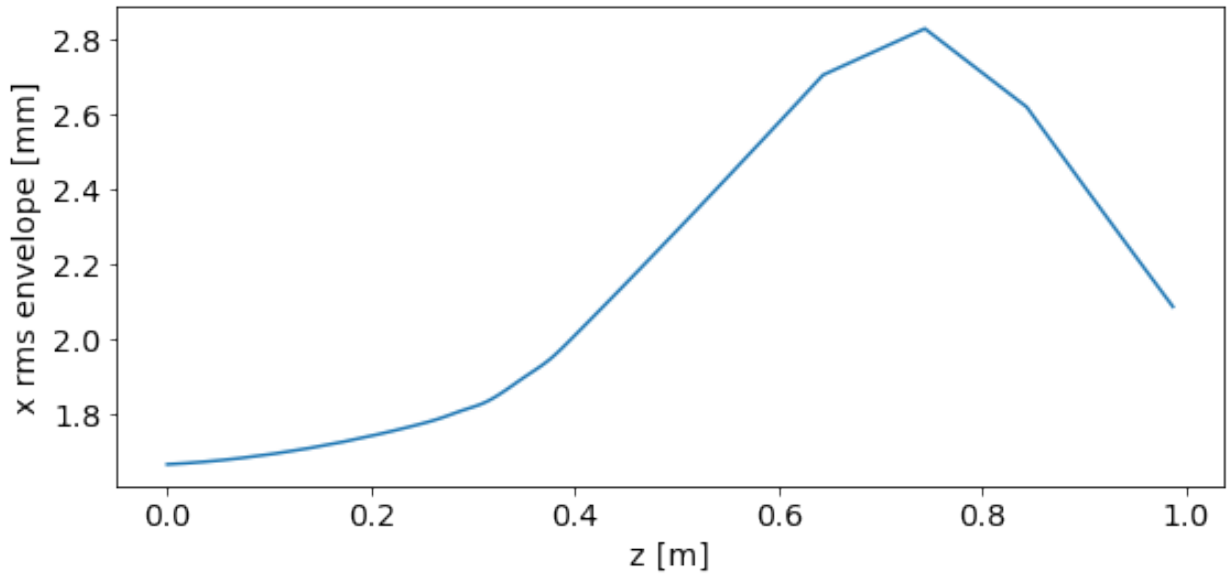
```
>>> sq.distribute() # generate initial distribution by using the input-file's␣
↪parameter
>>> sq.run() # run simulation
```

Sequence class contains all simulation results *(see here)*. User can plot results directly.

```
>>> import matplotlib.pylab as plt
>>> plt.plot(sq.hrefz(), sq.hrefeng('MeV'))
>>> plt.ylabel('Energy [MeV]')
>>> plt.xlabel('z [m]')
>>> plt.show()
```




```
>>> plt.plot(sq.hrefz(),sq.hxrms('mm'))
>>> plt.ylabel('x rms envelope [mm]')
>>> plt.xlabel('z [m]')
>>> plt.show()
```



2.2 2. Lattice parameter control

User can define **tag** (*dtag*) for each element like,

lattice_1cavity-1solenoid:

```
1 1
6 20642 2 0 1
65 65 129 4 0.14 0.14 0.1025446
```

```
19 0 0 2
10111 10531
0.0 0.0
1.4885271891392098E-10 1.5336340736585796E-10
0.0022734189 8.8312578E-5 0.0 1.0 1.0 0.0 0.0
0.0022734189 8.8312578E-5 0.0 1.0 1.0 0.0 0.0
0.076704772 3.4741445E-6 0.0 1.0 1.0 0.0 0.0
0.0 500000.0 9.3149432E8 0.13865546218487396 8.05E7 0.0 99.9
0.072 4 20 0 0.02 / # first lattice element
0.135064 4 20 0 0.02 /
0.24 60 20 110 0.64 8.05E7 349.740866 421 0.017 0.017 0.0 0.0 0.0 0.0 0.0 1. 2. / tag=
↪{'cav1'}
0.064263 4 20 0 0.02 /
0.05588 4 20 0 0.02 /
0.076123 4 20 0 0.02 /
0.1 1 1 3 5.34 0 0.02 / tag={'sol1','split1'} # user can set multiple tags
0 0 0 -21 0.02 0 0 0 0 0 0 /
0 0 0 -21 0.02 0 0 0 0 0 0 /
0.1 1 1 3 5.34 0 0.02 / tag=['sol1','split2'] # user can set the same tag name
0.076123 4 20 0 0.02 /
0.06727 4 20 0 0.02 /
0.0 0 100 -2 0.0 1 / tag=('moni1','pm1') # (),{},[] brackets are available
```

Check the lattice parameter with the tag name. – *conf*

```
>>> tmp = sq.conf('cav1') # user can also input index like, tmp = sq.conf(3)
index :  3 (start :  0.207064 , end :  0.447064  [m])
length :  0.24 [m]
segment :  60
step :  20
type :  110  (emfield)
scaling ('scale', 'scl_fac') :  0.64 [1]
frequency ('f', 'freq') :  349.74086645 [Hz]
input phase ('phi0',) :  -90.0 [deg]
fileid ('file', 'id') :  421.0
xaperture ('xaper', 'xpipe', 'xradius') :  0.017 [m]
yaperture ('yaper', 'ypipe', 'yradius') :  0.017 [m]
dx ('dx0',) :  0.0 [m]
dy ('dy0',) :  0.0 [m]
pitch ('dx1',) :  0.0 [rad]
yaw ('dy1',) :  0.0 [rad]
roll  :  0.0 [rad]
data type ('data',) :  1.0
coordinate ('coor',) :  2.0
synchronous phase flag ('sync_flag', 'syncflag',) :  0.0
scale error ('dscl', 'er_scl') :  0.0 [1]
phase error ('dphi', 'er_phi') :  0.0 [deg]
scaling2 ('scale2',) :  0.0 [1]
frequency2 ('f2', 'freq2') :  0.0 [Hz]
driven phase offset 2 ('phi0_2',) :  0.0 [deg]
scl_fac2 ('scaling3', 'scale3') :  0.0
frequency3 ('f3', 'freq3') :  0.0 [Hz]
driven phase offset 3 ('phi0_3',) :  0.0 [deg]
scale error 2 ('dscl2', 'er_scl2') :  0.0 [1]
phase error 2 ('dphi2', 'er_phi2') :  0.0 [deg]
scale error 3 ('dscl3', 'er_scl3') :  0.0 [1]
phase error 3 ('dphi3', 'er_phi3') :  0.0 [deg]
synchronous phase offset ('sync_offset',) :  0.0 [deg]
```

```
tag :  ['cav1']
```

Change the parameter.

```
>>> sq.conf('cav1', {'phi0':360.0}) # change driven phase to 360.0
```

User can *search* element by element-type or type-id. *search* command returns element indexes.

```
>>> sq.search(type='solenoid')
array([7, 10], dtype=int32)
```

```
>>> sq.search(type=3)
array([7, 10], dtype=int32)
```

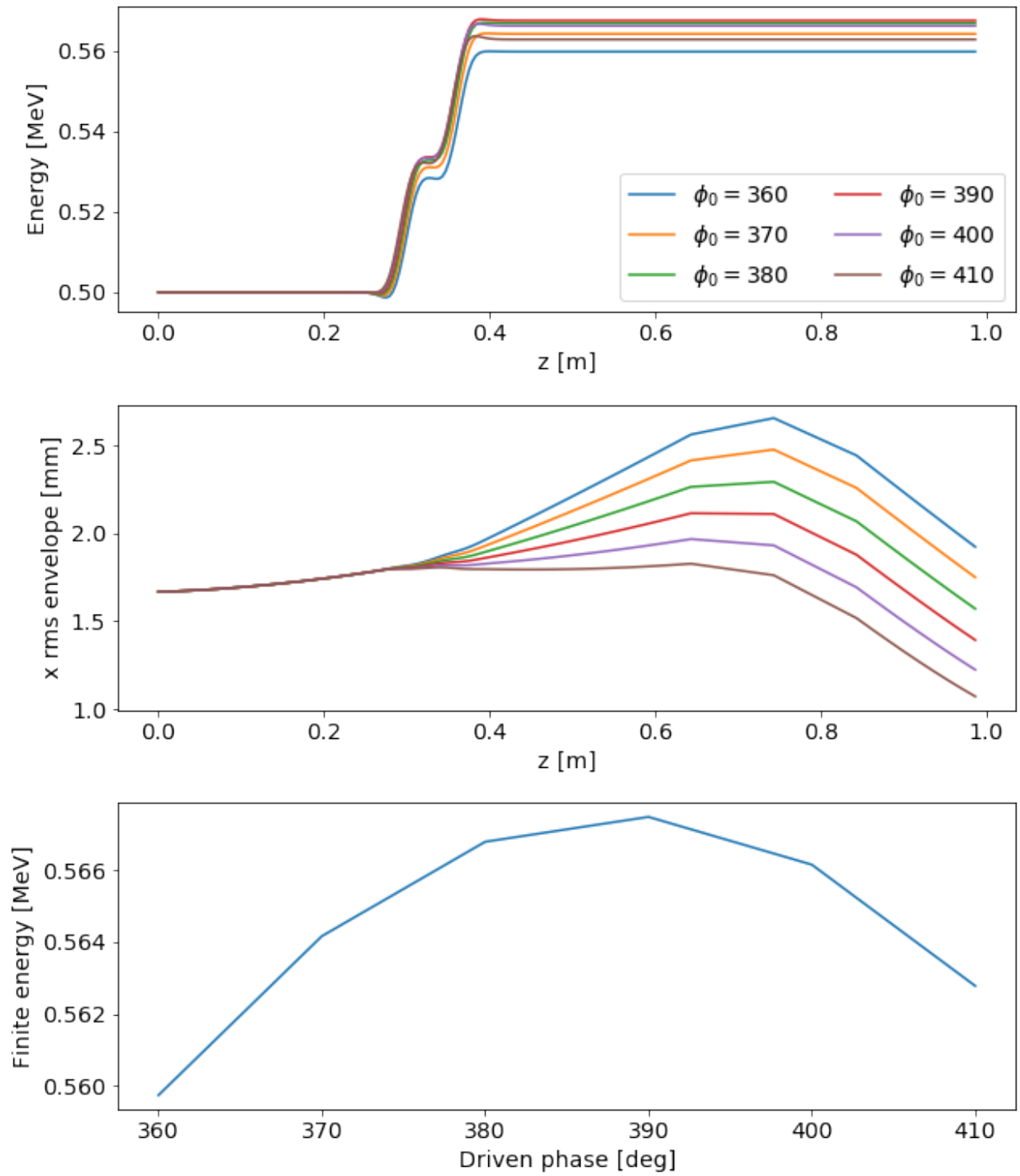
2.3 3. Example: Phase scan

User can scan parameters by using simple loop.

```
>>> phase = [360,370,380,390,400,410] # scan parameter list
>>> pos = []; eng = []; rms = []
>>> for p0 in phase:
>>>     sq.conf('cav1',{'phi0':p0}) # set new parameter
>>>     sq.distribute(); sq.run()
>>>     pos.append(sq.hrefz()) # store results
>>>     eng.append(sq.hrefeng('MeV'))
>>>     rms.append(sq.hxrms('mm'))
```

Plot the scan result.

```
>>> plt.rcParams['figure.figsize'] = [9,10]
>>> for (z,e,x,p0) in zip(pos, eng, xcn, phase):
>>>      plt.subplot(3,1,1)
>>>      plt.plot(z,e,label='$\phi_0 =$'+str(p0))
>>>      plt.subplot(3,1,2)
>>>      plt.plot(z,x)
>>>
>>> plt.subplot(3,1,1)
>>> plt.ylabel('Energy [MeV]')
>>> plt.xlabel('z [m]')
>>> plt.legend(loc='best', ncol=2)
>>> plt.subplot(3,1,2)
>>> plt.ylabel('x rms envelope [mm]')
>>> plt.xlabel('z [m]')
>>>
>>> e1 = [ee[-1] for ee in eng]
>>> plt.subplot(3,1,3)
>>> plt.plot(phase, e1)
>>> plt.xlabel('Driven phase [deg]')
>>> plt.ylabel('Finite energy [MeV]')
>>> plt.tight_layout()
>>> plt.show()
```



2.4 4. Beam parameter control

`Sequence` class contains all input parameters *(see here)*.

For example,

```
>>> sq.Energy # initial kinetic energy [eV]
array(500000.0)
>>> sq.Energy = 0.6e6 # set new energy
```

```
>>> sq.Distxquadratic # initial distribution parameter for x
array([  2.27341890e-03,   8.83125780e-05,   0.00000000e+00])
    # [           sigma,           lambda,               mu] - IMPACT internal unit
>>> sq.Distxtwiss # automatically converted to the twiss (Courant-Snyder) parameter
array([  0.00000000e+00,   4.99999992e-01,   1.18999999e-07])
    # [        alpha(1),          beta(m),   normalized emittance(m-rad)]
>>> sq.Distxtwiss = [0.1, 0.6, 1.3e-7] # set new twiss parameter
```

2.5 5. History result usage

Method for history results returns all segment results by default.

```
>>> sq.hxcen() # returns x centroid history
array([  3.34446507e-06,   3.57517607e-06, ..., -3.91164303e-05,  -3.97671083e-05,  -
↪4.04177862e-05])
```

User can input a unit to the method.

```
>>> sq.hxcen('mm')
array([ 0.00334447,  0.00357518, ..., -0.03911643, -0.03976711, -0.04041779])
```

In case of input float or list of float, the method returns the linear interpolated result.

```
>>> sq.hxcen(0.5, 'mm')
array([ 0.01071064])
>>> sq.hxcen([0.5, 0.6, 0.7], 'mm')
array([ 0.01071064,  0.01260737,  0.00144759])
```

In case of input the tag name, the method returns the result at exit point of the element by default.

```
>>> sq.hxcen('cav1', 'mm')
array([ 0.00758989])
```

In addition, user can specify the data point of the element.

```
>>> sq.hxcen('cav1', 'mm', at='entry') # returns result at the element entry
array([ 0.00599846])
>>> sq.hxcen('cav1', 'mm', at='exit') # returns result at the element exit
array([ 0.00970658])
```

For the multi-charge-states simulation, user can specify the charge state for the result.

```
>>> sq.qmlabel # returns charge-to-mass ratios which appeared in the simulation.
array([  1.48852719e-10,   1.53363407e-10])
```

```
>>> sq.hxcen('mm', qid=1) # returns x centroid history with charge-to-mass ratio = 1.
↪53363407e-10
array([ 0.00731273,  0.00757781, ..., -0.04237878, -0.04297157, -0.04356436])
```

All common parameters can be found in *hrefz*.

2.6 6. Distribution result usage

Method for history results returns initial distribution by default.

```
>>> sq.getx() # returns current x distribution
array([ 0.00190219, -0.00270052, ..., -0.0002277 , -0.00259636,  0.00128789])
```

User can input a unit to the method.

```
>>> sq.getx('mm')
array([ 1.90219117, -2.70052015, ..., -0.22769931, -2.59636435,  1.28789445])
```

User can input ID number of "-2" flagged lattice element to get the distribution there.

```
>>> sq.getx(100, 'mm') # returns x distribution at the flag element
array([ 3.00496309, -2.23733725, ..., -4.09860108, -2.0038936 ,  1.30413004])
```

Able to input the tag name also.

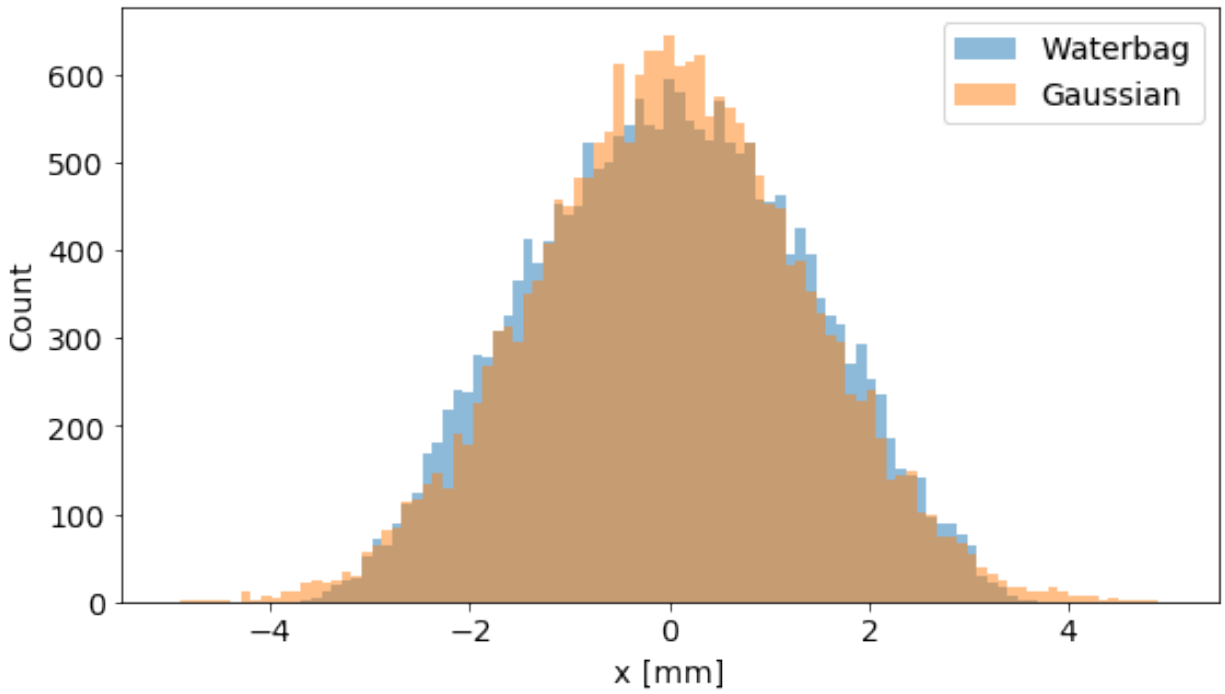
```
>>> sq.getx('pm1', 'mm') # returns x distribution at the flag element
array([ 3.00496309, -2.23733725, ..., -4.09860108, -2.0038936 ,  1.30413004])
```

All common parameters can be found in *getx*.

Note: The initial distribution is generated in *distribute()* function.

For example, user can check initial distribution before running particle tracking.

```
>>> sq.Distxtwiss = [0.0, 0.4, 1.5e-7] # set new x twiss parameter
>>> bins = np.linspace(-5.0,5.0,100)
>>>
>>> sq.Disttype = 'mcWB' # set as multi-charge Waterbag distribution
>>> sq.distribute()
>>> a,b,c=plt.hist(sq.getx(0, 'mm'),bins, alpha=0.5, label='Waterbag')
>>>
>>> sq.Disttype = 'mcGS' # set as multi-charge Gaussian distribution
>>> sq.distribute()
>>> a,b,c=plt.hist(sq.getx(0, 'mm'),bins, alpha=0.5, label='Gaussian')
>>>
>>> plt.xlabel('x [mm]')
>>> plt.ylabel('Count')
>>> plt.show()
```



2.7 7. A to B run (partial run)

run method supports to input *start* and *end* point of the simulation.

```
>>> sq.distribute()              # distribute the initial beam
>>> sq.run(start=1, end='cav1') # simulate from the 1st element to the 1st cavity␣
↪(index=3)
```

Before continuing the simulation, user can store the current beam distribution and the beam energy.

```
>>> dst1 = sq.getall(unit='impact') # store the whole beam distribution with *impact*␣
↪unit
>>> eng1 = sq.hrefeng()[-1]          # store the finite beam energy
>>> zend = sq.hrefz() [-1]           # store the finite beam position
```

Run simulation to the end of the lattice.

```
>>> sq.run(start=4, end=-1, zstart=zend)
```

In addition, user can restore the beam distribution and re-run the simulation. - *distribute()*

```
>>> sq.distribute(particles=dst1, unit='impact', Energy=eng1) # distribute by using␣
↪stored data
>>> sq.run(start=4, end=-1, zstart=zend)
```

2.8 8. Example: Transverse matching

If you want to match the beam at the monitoring point, you can use optimizer like in *SciPy*.

```
>>> from scipy.optimize import minimize
```

Define *cost function* by using the simulation results.

```
>>> desired_xrms = 2.5 # desired x rms size [mm]
>>>
>>> def cost(new_param):
>>>     sq.conf('sol1', {'bz':new_param}) # set new solenoid strength
>>>     sq.distribute()
>>>     sq.run()
>>>     diffx = sq.hxrms('pm1') - desired_xrms
>>>     return np.absolute(diffx)
```

Optimize the solenoid parameter

```
>>> minimize(cost, 5.0, method='Nelder-Mead') # where 5.0 is the initial parameter␣
↪for the solenoid strength
final_simplex: (array([[ 4.58441162],
       [ 4.58447266]]), array([  5.81423675e-06,   2.54995700e-05]))
           fun: 5.8142367538316364e-06
       message: 'Optimization terminated successfully.'
          nfev: 30
           nit: 15
        status: 0
       success: True
             x: array([ 4.58441162])
```

In this example, the parameter is one-dimension, but most of the optimizers support multi-dimensional parameter for more complex systems.

2.9 9. Lattice construction

User can *construct* new beam transport line by using Python interface.

```
>>> drift = [0.5, 10, 10, 0, 0.2]
# define element by using list of numbers (format is the same as the input file)
>>> quad1 = {'length':0.2, 'seg':4, 'step':4, 'type':'quadrupole', 'B2': 10.0, 'aper
↪':0.2}
# define element by using python dictionary
>>> quad2 = quad1.copy()
>>> quad2['B2'] = -10.0
>>> FODO = [drift, quad1, drift, quad2, drift]
>>> sq.construct(FODO)
```

In addtion, user can *insert* new lattice element.

```
>>> quad1 = {'length':0.2, 'seg':4, 'step':4, 'type':'quadrupole', 'B2': 10.0, 'aper
↪':0.2, 'tag':'new_quad'}
# define element by using python dictionary
>>> sq.construct(3, quad1) # insert new element
>>> sq.conf(3)
```

```
index :  4 (start :  1.2 , end :  1.4  [m])
length :  0.2 [m]
segment :  4
step :  4
type :  1  (quadrupole)
gradient ('grad', 'b2', 'k', 'voltage') :  10.0 [T/m]
switch ('flag',) :  0.0
aperture ('aper', 'pipe', 'radius') :  0.2 [m]
dx ('dx0',) :  0.0 [m]
dy ('dy0',) :  0.0 [m]
pitch ('dx1',) :  0.0 [rad]
yaw ('dy1',) :  0.0 [rad]
roll  :  0.0 [rad]
tag :  ['new_quad']
```

2.10 10. Data export

User can *save* current settings as IMPACT input format.

```
>>> sq.save('test.in.new') # export settings to 'test.in.new'
```

User can *output* results as original IMAPCT format.

```
>>> sq.output() # export simulation results to 'fort.*'
```

CHAPTER

THREE

PARALLEL COMPUTING

IMPACT supports MPI (Message Passing Interface) for Parallel computing.

3.1 MPI usage in command line

Python API supports MPI by mpi4py, all *Sequence* class objects are parallelized automatically.

For example,

run_parallel.py:

```
from impact import Sequence, mpisize, mpirank
import scipy as np

print 'Launch with process '+str(mpirank)+' of '+str(mpisize)+'.'
sq = Sequence('user-defined-input')
sq.subdir = 'data'
sq.load()
if mpirank == 0: print 'Run Simulation'
sq.distribute()
sq.run() # compute in parallel

if mpirank == 0:
    print 'Output Simulation Result'
    np.savetxt('result.dat', np.transpose([sq.hrefz(), sq.hxrms(), sq.hyrms()]))
```

and user can execute this script by using `mpirun`.

```
$ mpirun -np 4 python run_parallel.py
Launch with process 3 of 4.
Launch with process 1 of 4.
Launch with process 0 of 4.
Launch with process 2 of 4.
Run Simulation
Output Simulation Result
```

3.2 MPI usage in jupyter-notebook

Install ipyparallel with proper permissions.

```
$ pip install ipyparallel # may need to install with sudo
```

Launch MPI cluster.

```
$ ipcluster start -n 4 --engines=MPIEngineSetLauncher # launch with 4 processes
```

Note: Cluster launched directory is defined as the working directory. If you input **relative path** in the notebook, the path is pursued form this working directory.

After the clusters are running with MPI, user can start `jupyter-notebook` in parallel.

```
$ jupyter-notebook
```

In the notebook, user need to create a `Client` to IPython parallel cluster.

```
import ipyparallel as ipp
rc = ipp.Client()
```

Then, user can write **parallelized cell** in notebook by using `%%px` command.

```
%%px # execute cell in parallel
from impact import Sequence, mpisize, mpirank
print 'Launch with process '+str(mpirank)+' of '+str(mpisize)+'.'
---------------------------------------------------------------
[stdout:0] Launch with process 1 of 4.
[stdout:1] Launch with process 0 of 4.
[stdout:2] Launch with process 2 of 4.
[stdout:3] Launch with process 3 of 4.
```

```
%%px # execute cell in parallel
sq = Sequence('user-defined-input')
sq.subdir = 'data'
sq.load()
sq.distribute()
sq.run()
```

Simulation results are broadcasted to all processes by default. – *autobcast*

```
%%px
print sq.hxcen('moni1','mm')
---------------------------------------------------------------
[stdout:0] [ 2.08796354]
[stdout:1] [ 2.08796354]
[stdout:2] [ 2.08796354]
[stdout:3] [ 2.08796354]
```

```
%%px
sq.autobcast = False # no broadcasting
print mpirank,sq.hxrms('moni1','mm')
---------------------------------------------------------------
[stdout:0] 2 None
[stdout:1] 1 None
[stdout:2] 0 [ 2.08796354]
[stdout:3] 3 None
```

3.3 Tips

There are 2 methods to make cost-function for optimization.

Easy method:

```
sq.autobcast = True # default setting
def cost_func(input_value):
    ...
    # set parameter from input_value
    ...
    sq.distribute(); sq.run() # run simulation

    # calculate cost value
    xx = sq.hxrms() # broadcast full list of x rms
    yy = sq.hyrms() # broadcast full list of y rms
    cost = np.std(xx)*np.mean(xx)*np.std(yy)*np.mean(yy)
    return cost
```

Fast method:

```
sq.autobcast = False # no auto broadcasting
def cost_func(input_value):
    ...
    # set parameter from input_value
    ...
    sq.distribute(); sq.run() # run simulation

    # calculate cost value
    if mpirank == 0:
        xx = sq.hxrms()
        yy = sq.hyrms()
        cost = np.std(xx)*np.mean(xx)*np.std(yy)*np.mean(yy)
    else
        cost = None

    cost = mpicomm.bcast(cost, root=0) # broadcast single float only

    return cost
```

CHAPTER

FOUR

LATTICE ELEMENTS

Basic format of the one lattice element is,

```
col1  col2  col3  col4 ... /
```

Length of the lattice parameter-array (col{n}) must be greater than or equals to **4**.

4.1 Optical element

For the optical elements, first 4 columns of the parameter-array have same significance.

```
element-length    seg    step    type    parameter-array[5:] /
```

parameter-array

Col 1 (float) - *element length ('length', 'L')* [m].

Col 2 (int) - *number of element segmentations ('segment', 'seg')* [1].

Col 3 (int) - *number of map steps for PIC calculation ('step', 'pid')* [1].

Col 4 (int) - *lattice element type ('type')* [1].

element-type	value
drift	**0**
quadrupole	**1**
constant focusing	**2**
linear-map solenoid	**3**
hard-edge solenoid	**13**
dipole	**4**
multipole	**5**
drift tube linac	**101**
coupled cavity dtl	**102**
coupled cavity linac	**103**
superconducting cavity	**104**
solenoid with rf field	**105**
rf quadrupole	**106**
em field	**110**
em field dipole	**114**

4.1.1 drift (`type` : 0)

Drift space element.

(**keywords**: 'drift')

`parameter-array`

Col 5 (float) - *pipe radius* [m].

(**keywords**: 'aperture', 'aper', 'pipe', 'radius')

4.1.2 quadrupole (`type` : 1)

Magnetic or electrostatic quadrupole element.

(**keywords**: 'quadrupole', 'quad')

`parameter-array`

Col 5 (float) - *gradient* [T/m] for magnetic or *voltage* [V] for electrostatic quadrupole.

(**keywords**: 'gradient', 'grad', 'b2', 'k', 'voltage')

Col 6 (float) - *pipe radius* [m].

(**keywords**: 'aperture', 'aper', 'pipe', 'radius')

Col 7 (float) - *quadrupole type switch* [1].

(**keywords**: 'switch', 'flag')

quadrupole type	value
magnetic	**0**
electrostatic	**1**

Col 8 (float) - *horizontal (x) alignment error* [m], or *entrance x offset* [m]. (see `Flagerror`)

(**keywords**: 'dx', 'dx0')

Col 9 (float) - *vertical (y) alignment error* [m], or *entrance y offset* [m]. (see `Flagerror`)

(**keywords**: 'dy', 'dy0')

Col 10 (float) - *pitch angle alignment error* [rad], or *exit x offset* [m]. (see `Flagerror`)

(**keywords**: 'pitch', 'dx1')

Col 11 (float) - *yaw angle alignment error* [rad], or *exit y offset* [m]. (see `Flagerror`)

(**keywords**: 'yaw', 'dy1')

Col 12 (float) - *roll angle alignment error* [rad].

(**keywords**: 'roll')

4.1.3 constant focusing (`type` : 2)

3D constant focusing element.

(**keywords**: 'constant focusing', 'constfocus')

`parameter-array`

Col 5 (float) - *horizontal (x) focusing strength*.

(**keywords**: 'kx0^2', 'kx2')

Col 6 (float) - *vertical (y) focusing strength*.

(**keywords**: 'ky0^2', 'ky2')

Col 7 (float) - *longitudinal (z) focusing strength*.

(**keywords**: 'kz0^2', 'kz2')

Col 8 (float) - *pipe radius* [m].

(**keywords**: 'aperture', 'aper', 'pipe', 'radius')

4.1.4 linear-map solenoid (`type` : 3)

Linear map solenoid element.

(**keywords**: 'solenoid', 'sol')

`parameter-array`

Col 5 (float) - *longitudinal magnetic field strength (Bz)* [T].

(**keywords**: 'bz0', 'bz', 'b')

Col 6 (float) - *file id* [1].

(**keywords**: 'fileid', 'file', 'id')

Col 7 (float) - *pipe radius* [m].

(**keywords**: 'aperture', 'aper', 'pipe', 'radius')

Col 8 ~ 12 (float) - *alignment errors*. *Same as quadrupole col 8 ~ 12*.

4.1.5 hard-edge solenoid (`type` : 13)

Hard-edge solenoid element. This can be used with *hard-edge solenoid wrapper* (`type` = **-40**).

(**keywords**: 'solenoid2', 'sol2')

`parameter-array`

Col 5 (float) - *longitudinal (z) magnetic field strength* [T].

(**keywords**: 'bz0', 'bz', 'b')

Col 6 (float) - *hard-edge wrapper flag* [1].

- **0** for no-auto wrapping (**-40** wrapper is mandatory).
- **1** for auto wrapping with the same field strength.

(**keywords**: 'fileid', 'file', 'id')

Col 7 (float) - *pipe radius* [m].

(**keywords**: 'aperture', 'aper', 'pipe', 'radius')

Col 8 (float) - *horizontal (x) magnetic field strength* [T].

(**keywords**: 'bx')

Col 9 (float) - *vertical (y) magnetic field strength* [T].

(**keywords**: 'by')

Col 10 ~ 14 (float) - *alignment errors*. *Same as quadrupole col 8 ~ 12*.

4.1.6 dipole (`type : 4`)

Magnetic or electrostatic dipole (bending) element.

(**keywords**: 'dipole', 'bend')

`parameter-array`

Col 5 (float) - *bending angle* [deg].

(**keywords**: 'angle', 'phi')

Col 6 (float) - *reference Lorentz beta*gamma* [1].

(**keywords**: 'beta*gamma', 'bg')

Col 7 (float) - *dipole type switch* [1]. Greater than **100** for 2nd order matrix.

(**keywords**: 'switch', 'flag')

dipole type	value
magnetic front-side	**400**
magnetic sector	**500**
magnetic back-side	**600**
electrostatic-cylindrical	**1500**
electrostatic-spherical	**1501**

Col 8 (float) - *pipe radius* [m].

(**keywords**: 'aperture', 'aper', 'pipe', 'radius')

Col 9 (float) - *front pole-face angle* [deg].

(**keywords**: 'front angle', 'angle1', 'phi1')

Col 10 (float) - *back pole-face angle* [deg].

(**keywords**: 'back angle', 'angle2', 'phi2')

Col 11 (float) - *front curvature* [1/m].

(**keywords**: 'front curvature', 'curv1')

Col 12 (float) - *back curvature* [1/m].

(**keywords**: 'back curvature', 'curv2')

Col 13 (float) - *fringe quadrupole term.*

(**keywords**: 'fringe q-term', 'kf')

Col 14 (float) - *main quadrupole term.*

(**keywords**: 'main q-term', 'k')

4.1.7 multipole (`type` : 5)

Magnetic multipole element.

(**keywords**: 'multipole', 'mpole')

`parameter-array`

Col 5 (float) - *quadrupole strength* [T/m].

(**keywords**: 'gradient', 'grad', 'b2')

Col 6 (float) - *sextupole strength* [T/m^2].

(**keywords**: 'b3')

Col 7 (float) - *octupole strength* [T/m^3].

(**keywords**: 'b4')

Col 8 (float) - *decapole strength* [T/m^4].

(**keywords**: 'b5')

Col 9 (float) - *12 pole strength* [T/m^5].

(**keywords**: 'b6')

Col 10 (float) - *14 pole strength* [T/m^6].

(**keywords**: 'b7')

Col 11 (float) - *16 pole strength* [T/m^7].

(**keywords**: 'b8')

Col 12 (float) - *pipe radius* [m].

(**keywords**: 'aperture', 'aper', 'pipe', 'radius')

Col 13 ~ 17 (float) - *alignment errors*. *Same as quadrupole col 8 ~ 12*.

4.1.8 drift tube linac (`type` : 101)

Drift tube linac element

(**keywords**: 'drift tube linac', 'dtl')

`parameter-array`

Col 5 (float) - *scaling factor* [1].

(**keywords**: 'scaling', 'scale', 'scl_fac')

Col 6 (float) - *frequency* [Hz].

(**keywords**: 'frequency', 'f', 'freq')

Col 7 (float) - *driven phase* [deg].

(**keywords**: 'driven phase', 'phi0')

Col 8 (float) - *file id* [1]. This element uses 'rfdata${file_id}' file. *(see here)*

(**keywords**: 'fileid', 'file', 'id')

Col 9 (float) - *pipe radius* [m].

(**keywords**: 'aperture', 'aper', 'pipe', 'radius')

Col 10 (float) - *first quadrupole length* [m].

(**keywords**: 'quad 1 length', 'q1len')

Col 11 (float) - *first quadrupole strength* [1/T].

(**keywords**: 'quad 1 gradient', 'q1grad')

Col 12 (float) - *second quadrupole length* [m].

(**keywords**: 'quad 2 length', 'q2len')

Col 13 (float) - *second quadrupole strength* [1/T].

(**keywords**: 'quad 2 gradient', 'q2grad')

Col 14 ~ 18 (float) - *alignment errors*. *Same as quadrupole col 8 ~ 12*.

4.1.9 coupled cavity dtl (`type` : 102)

Coupled cavity drift tube linac element

(**keywords**: 'coupled cavity drift tube linac', 'ccdtl')

`parameter-array`

Col 5 (float) - *scaling factor* [1].

(**keywords**: 'scaling', 'scale', 'scl_fac')

Col 6 (float) - *frequency* [Hz].

(**keywords**: 'frequency', 'f', 'freq')

Col 7 (float) - *driven phase* [deg].

(**keywords**: 'driven phase', 'phi0')

Col 8 (float) - *file id* [1]. This element uses 'rfdata${file_id}' file. *(see here)*

(**keywords**: 'fileid', 'file', 'id')

Col 9 (float) - *pipe radius* [m].

(**keywords**: 'aperture', 'aper', 'pipe', 'radius')

Col 10 ~ 14 (float) - *alignment errors*. *Same as quadrupole col 8 ~ 12*.

4.1.10 coupled cavity linac (`type` : 103)

Coupled cavity linac element

(**keywords**: 'coupled cavity linac', 'ccl')

`parameter-array`

Col 5 (float) - *scaling factor* [1].

(**keywords**: 'scaling', 'scale', 'scl_fac')

Col 6 (float) - *base frequency* [Hz].

(**keywords**: 'frequency', 'f', 'freq')

Col 7 (float) - *input phase* [deg].

(**keywords**: 'input phase', 'phi0')

Col 8 (float) - *file id* [1]. This element uses 'rfdata${file_id}' file. *(see here)*

(**keywords**: 'fileid', 'file', 'id')

Col 9 (float) - *pipe radius* [m].

(**keywords**: 'aperture', 'aper', 'pipe', 'radius')

Col 10 ~ 14 (float) - *alignment errors*. *Same as quadrupole col 8 ~ 12*.

Col 15 (float) - *flag for synchronous phase input* [1]. *(see here)*

- **0** for input phase (col 6) is driven phase.
- **1** for input phase (col 6) is synchronous phase.

(**keywords**: 'synchronous phase flag', 'sync_flag', 'syncflag')

Col 16 (float) - *scaling error for primary harmonic* [1]. (added to the scaling factor)

(**keywords**: 'scale error', 'dscl', 'er_scl')

Col 17 (float) - *phase error for primary harmonic* [deg].

(**keywords**: 'phase error', 'dphi', 'er_phi')

Col 18 (float) - *scaling factor for second harmonic* [1].

(**keywords**: 'scaling2', 'scale2', 'scl_fac2')

Col 19 (float) - *frequency ratio to the base frequency for second harmonic* [1].

(**keywords**: 'frequency ratio 2', 'f2', 'freq2')

Col 20 (float) - *driven phase offset for second harmonic* [deg].

(**keywords**: 'driven phase offset 2', 'phi0_2')

Col 21 (float) - *scaling factor for third harmonic* [1].

(**keywords**: 'scaling3', 'scale3', 'scl_fac3')

Col 22 (float) - *frequency ratio to the base frequency for third harmonic* [1].

(**keywords**: 'frequency ratio 3', 'f3', 'freq3')

Col 23 (float) - *driven phase offset for third harmonic* [deg].

(**keywords**: 'driven phase offset 3', 'phi0_3')

Col 24 (float) - *scaling error for second harmonic* [1]. (added to the scaling factor)

(**keywords**: 'scale error 2', 'dscl2', 'er_scl2')

Col 25 (float) - *phase error for second harmonic* [deg].

(**keywords**: 'phase error 2', 'dphi2', 'er_phi2')

Col 26 (float) - *scaling error for third harmonic* [1]. (added to the scaling factor)

(**keywords**: 'scale error 3', 'dscl3', 'er_scl3')

Col 27 (float) - *phase error for third harmonic* [deg].

(**keywords**: 'phase error 3', 'dphi3', 'er_phi3')

Col 28 (float) - *synchronous phase offset* [deg]. Read only, set after the simulation.

(**keywords**: 'synchronous phase offset', 'sync_offset')

4.1.11 superconducting cavity (`type` : 104)

Superconducting cavity element

(**keywords**: 'superconducting cavity', 'scc')

`parameter-array`

Col 5 (float) - *scaling factor* [1].

(**keywords**: 'scaling', 'scale', 'scl_fac')

Col 6 (float) - *frequency* [Hz].

(**keywords**: 'frequency', 'f', 'freq')

Col 7 (float) - *driven phase* [deg].

(**keywords**: 'driven phase', 'phi0')

Col 8 (float) - *file id* [1]. This element uses 'rfdata${file_id}' file. *(see here)*

(**keywords**: 'fileid', 'file', 'id')

Col 9 (float) - *pipe radius* [m].

(**keywords**: 'aperture', 'aper', 'pipe', 'radius')

Col 10 ~ 14 (float) - *alignment errors*. *Same as quadrupole col 8 ~ 12*.

4.1.12 solenoid-rf (`type` : 105)

Solenoid with imbeded RF field element.

(**keywords**: 'solenoid-rf', 'solrf')

`parameter-array`

Col 5 (float) - *scaling factor* [1].

(**keywords**: 'scaling', 'scale', 'scl_fac')

Col 6 (float) - *frequency* [Hz].

(**keywords**: 'frequency', 'f', 'freq')

Col 7 (float) - *driven phase* [deg].

(**keywords**: 'driven phase', 'phi0')

Col 8 (float) - *file id* [1]. This element uses 'rfdata${file_id}' file. *(see here)*

(**keywords**: 'fileid', 'file', 'id')

Col 9 (float) - *pipe radius* [m].

(**keywords**: 'aperture', 'aper', 'pipe', 'radius')

Col 10 ~ 14 (float) - *alignment errors*. *Same as quadrupole col 8 ~ 12*.

Col 15 (float) - *longitudinal magnetic field strength (Bz)* [T].

(**keywords**: 'bz0', 'bz', 'b')

4.1.13 rf quadrupole (`type` : 106)

RF quadrupole element. In case of *`Flaginteg`* equals to **2**, `element-length` rescales the length defined in the external file (input **0** for no rescaling), in addition, `segment` and `step` are treated as dummy parameter.

(**keywords**: 'rfquadrupole', 'rfquad', 'rfq')

`parameter-array`

Col 5 (float) - *scaling factor* [1].

(**keywords**: 'scaling', 'scale', 'scl_fac')

Col 6 (float) - *frequency* [Hz].

(**keywords**: 'frequency', 'f', 'freq')

Col 7 (float) - *driven phase* [deg].

(**keywords**: 'driven phase', 'phi0')

Col 8 (float) - *file id* [1]. This element uses 'rfqline${file_id}' file in case of *`Flaginteg`* equals to **2**. *(see here)*

(**keywords**: 'fileid', 'file', 'id')

Col 9 (float) - *pipe radius* [m].

(**keywords**: 'aperture', 'aper', 'pipe', 'radius')

Col 10 (float) - *modulation* [1].

(**keywords**: 'modulation', 'mod')

Col 11 ~ 15 (float) - *alignment errors*. *Same as quadrupole col 8 ~ 12*.

4.1.14 em field (`type` : 110)

3D electro-magnetic field element with data file.

This element only works in case of *`Flaginteg`* equals to **2**.

(**keywords**: 'emfield', 'emfld')

`parameter-array`

Col 5 (float) - *scaling factor* [1].

(**keywords**: 'scaling', 'scale', 'scl_fac')

Col 6 (float) - *frequency* [Hz].

(**keywords**: 'frequency', 'f', 'freq')

Col 7 (float) - *driven phase* [deg].

(**keywords**: 'driven phase', 'phi0')

Col 8 (float) - *file id* [1]. This element uses '1T${file_id}.T7' file. *(see here)*

(**keywords**: 'fileid', 'file', 'id')

Col 9 (float) - *horizontal (x) pipe radius* [m].

(**keywords**: 'xaperture', 'xaper', 'xpipe', 'xradius')

Col 10 (float) - *vertical (y) pipe radius* [m].

(**keywords**: 'yaperture', 'yaper', 'ypipe', 'yradius')

Col 11 ~ 15 (float) - *alignment errors*. *Same as quadrupole col 8 ~ 12*.

Col 16 (float) - *data type* [1].

- **1** for discrete data only. (recommended)
- **2** for both discrete and analytical data.

(**keywords**: 'data type', 'data')

Col 17 (float) - *coordinate type* [1].

- **1** for cylindrical coordinate.
- **2** for Cartesian coordinate.

(**keywords**: 'coordinate', 'coor')

Col 18 (float) - *flag for synchronous phase input* [1]. *(see here)*

- **0** for input phase (col 6) is driven phase.
- **1** for input phase (col 6) is synchronous phase.

(**keywords**: 'synchronous phase flag', 'sync_flag', 'syncflag')

Col 19 (float) - *scaling error for primary field* [1]. (added to the scaling factor)

(**keywords**: 'scale error', 'dscl', 'er_scl')

Col 20 (float) - *phase error for primary field* [deg].

(**keywords**: 'phase error', 'dphi', 'er_phi')

Col 21 (float) - *scaling factor for second field* [1].

(**keywords**: 'scaling2', 'scale2', 'scl_fac2')

Col 22 (float) - *frequency for second field* [Hz].

(**keywords**: 'frequency2', 'f2', 'freq2')

Col 23 (float) - *driven phase offset for second field* [deg].

(**keywords**: 'driven phase offset 2', 'phi0_2')

Col 24 (float) - *scaling factor for third field* [1].

(**keywords**: 'scaling3', 'scale3', 'scl_fac3')

Col 25 (float) - *frequency for second field* [Hz].

(**keywords**: 'frequency3', 'f3', 'freq3')

Col 26 (float) - *driven phase offset for third field* [deg].

(**keywords**: 'driven phase offset 3', 'phi0_3')

Col 27 (float) - *scaling error for second field* [1]. (added to the scaling factor)

(**keywords**: 'scale error 2', 'dscl2', 'er_scl2')

Col 28 (float) - *phase error for second field* [deg].

(**keywords**: 'phase error 2', 'dphi2', 'er_phi2')

Col 29 (float) - *scaling error for third field* [1]. (added to the scaling factor)

(**keywords**: 'scale error 3', 'dscl3', 'er_scl3')

Col 30 (float) - *phase error for third field* [deg].

(**keywords**: 'phase error 3', 'dphi3', 'er_phi3')

Col 31 (float) - *synchronous phase offset* [deg]. Read only, set after the simulation.

(**keywords**: 'synchronous phase offset', 'sync_offset')

4.1.15 em field dipole (`type` : 114)

3D electro-magnetic field dipole element with data file. `element-length` is treated as the path length of the pipe center.

This element only works in case of *`Flaginteg`* equals to **2**.

(**keywords**: 'em field dipole', 'emfdipole', 'emfbend')

`parameter-array`

Col 5 (float) - *scaling factor* [1].

(**keywords**: 'scaling', 'scale', 'scl_fac')

Col 6 (float) - *bending angle* [deg].

(**keywords**: 'angle', 'phi')

Col 7 (float) - *fringe length* [m]. This length is treated as straight section in the `element-length`.

(**keywords**: 'fringe length', 'flen')

Col 8 (float) - *file id* [1]. This element uses '1T${file_id}.T7' file. *(see here)*

(**keywords**: 'fileid', 'file', 'id')

Col 9 (float) - *horizontal (x) pipe width* [m].

(**keywords**: 'xaperture', 'xaper', 'xpipe', 'xwidth')

Col 10 (float) - *vertical (y) pipe width* [m].

(**keywords**: 'yaperture', 'yaper', 'ypipe', 'ywidth')

Col 11 ~ 15 (float) - *alignment errors*. *Same as quadrupole col 8 ~ 12*.

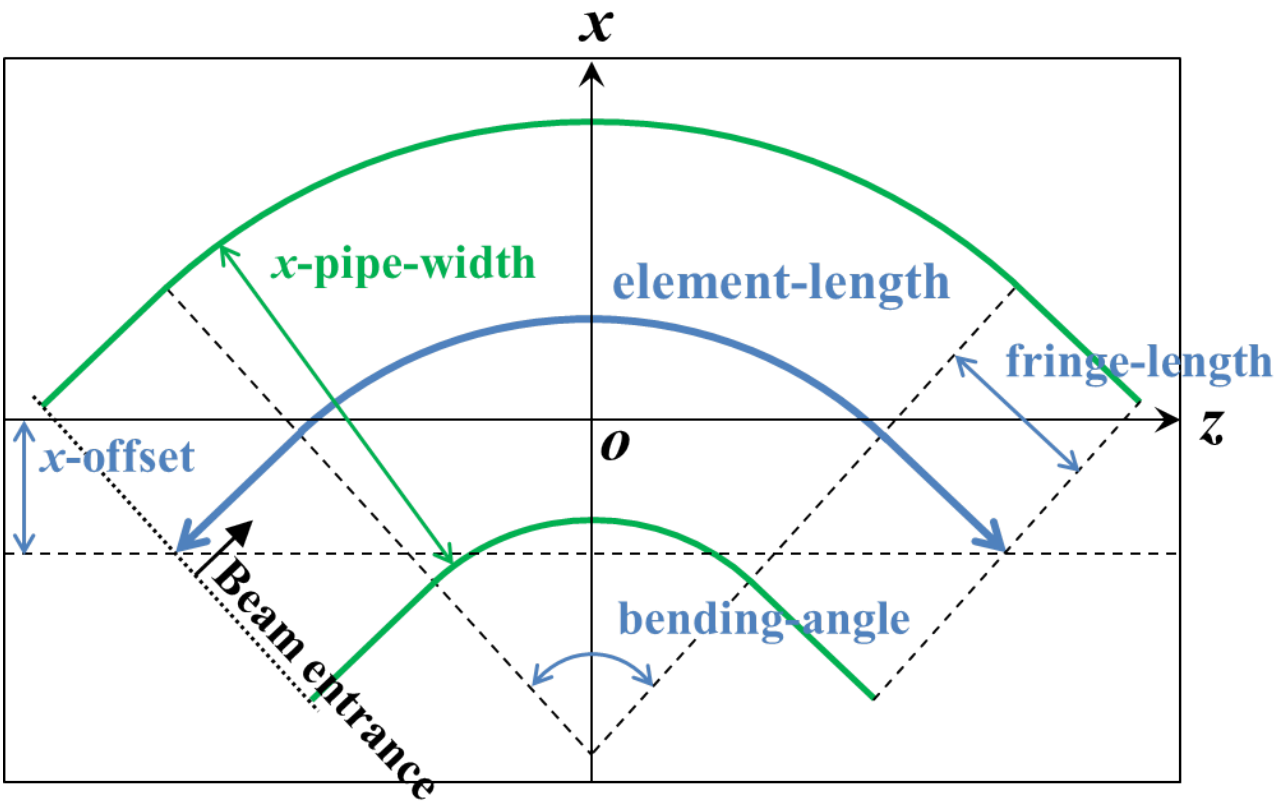
Col 16 (float) - *horizontal (x) offset for field data* [m].

(**keywords**: 'xoffset', 'xoff')

Col 17 (float) - *vertical (y) offset for field data* [m].

(**keywords**: 'yoffset', 'yoff')

Note: z-x plane data image for input parameters. The reference line assumes x-axial symmetry.



4.2 Special element

parameter-array

Col 4 (int) - *special element type* [1].

element-type	value
2d centroid adjust	**-1**
4d centroid adjust	**-26**
store beam distribution	**-2**
make beam mismatch	**-10**
charge stripper	**-11**
collimator slit	**-13**
beam parameter selector	**-14**
multi-charge beam shift	**-21**
pure beam shift	**-25**
transverse rotation	**-22**
diagnostic flag	**-23**
jump reference frequency	**-27**
hard-edge solenoid wrapper	**-40**
lattice end flag	**-99**

4.2.1 2d centroid adjust (`type` : -1)

Adjust the horizontal and vertical centroid to zero.

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *dummy parameter*.

Col 4 (int) - *element-type* = -1

4.2.2 4d centroid adjust (`type` : -26)

Adjust the horizontal and vertical, centroid and momentum centroid to zero.

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *dummy parameter*.

Col 4 (int) - *element-type* = -26

4.2.3 store beam distribution (`type` : -2)

Store the 6D beam distribution.

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *storage id* [1].

Col 4 (int) - *element-type* = -2

Col 5 (int) - *Sampling interval* [1].

4.2.4 make beam mismatch (`type` : -10)

Make mismatch for the 6D beam distribution.

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *dummy parameter*.

Col 4 (int) - *element-type* = -10

Col 5 (float) - *dummy parameter*.

Col 6 (float) - *x mismatch factor* [1].

Col 7 (float) - *x' mismatch factor* [1].

Col 8 (float) - *y mismatch factor* [1].

Col 9 (float) - *y' mismatch factor* [1].

Col 10 (float) - *z mismatch factor* [1].

Col 11 (float) - *z' mismatch factor* [1].

4.2.5 charge stripper (`type` : -11)

MSU model charge stripper.

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *file id* [1]. This element uses 'fort.${file_id}' file.

Col 4 (int) - *element-type* = -11

4.2.6 collimator slit (`type` : -13)

Collimator slit in x-y plane.

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *dummy parameter*.

Col 4 (int) - *element-type* = -13

Col 5 (float) - *dummy parameter*.

Col 6 (float) - *x lower limit* [m].

Col 7 (float) - *x higher limit* [m].

Col 8 (float) - *y lower limit* [m].

Col 9 (float) - *y higher limit* [m].

4.2.7 beam parameter selector (`type` : -14)

Beam parameter selector by usings high and low limitation

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *dummy parameter*.

Col 4 (int) - *element-type* = -14

Col 5 (int) - *index for the beam parameter*.

index	description	unit
0	skip selector	
1	x position	[m]
2	x momentum	[rad]
3	y position	[m]
4	y momentum	[rad]
5	relative phase	[deg]
6	relative or absolute energy	[MeV]
7	charge to mass ratio	[c^2/eV]
8	charge per macro particle	[1]
9	id number	[1]

Col 6 (float) - *lower limit for the parameter*.

Col 7 (float) - *higher limit for the parameters*.

Col 8 (int) - *flag for reference energy offset*.

- **0** for use relative energy (default)
- **1** for use absolute energy

4.2.8 multi-charge beam shift (`type` : -21)

Shift the beam in 6D phase space. The shift amount is depends on each charge state.

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *dummy parameter*.

Col 4 (int) - *element-type* = -21

Col 5 (float) - *dummy parameter*.

Col 6 (float) - *x shift amount* [m].

Col 7 (float) - *x' shift amount* [rad].

Col 8 (float) - *y shift amount* [m].

Col 9 (float) - *y' shift amount* [rad].

Col 10 (float) - *z shift amount* [deg].

Col 11 (float) - *z' shift amount* [MeV].

4.2.9 pure beam shift (`type` : -25)

Shift the beam in 6D phase space. The shift amount is **NOT** depends on each charge state.

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *dummy parameter*.

Col 4 (int) - *element-type* = -25

Col 5 (float) - *dummy parameter*.

Col 6 (float) - *x shift amount* [m].

Col 7 (float) - *x' shift amount* [rad].

Col 8 (float) - *y shift amount* [m].

Col 9 (float) - *y' shift amount* [rad].

Col 10 (float) - *z shift amount* [deg].

Col 11 (float) - *z' shift amount* [MeV].

4.2.10 transverse rotation (`type` : -22)

Transverse rotation of the beam.

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *dummy parameter*.

Col 4 (int) - *element-type* = -22

Col 5 (float) - *dummy parameter*.

Col 6 (float) - *x-y plane rotation angle* [deg].

Col 7 (float) - *x'-y' plane rotation angle* [deg].

4.2.11 diagnostic flag (`type` : -23)

Store the diagnostic data of the beam distribution.

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *dummy parameter*.

Col 4 (int) - *element-type* = -23

4.2.12 jump reference frequency (`type` : -27)

Jump the reference frequency.

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *dummy parameter*.

Col 4 (int) - *element-type* = -27

Col 5 (float) - *dummy parameter*.

Col 6 (float) - *new reference frequency* [Hz].

Col 7 (float) - *phase shift* [deg].

4.2.13 hard-edge solenoid wrapper (`type` : -40)

Wrapping the hard-edge solenoid at the front and back side.

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *dummy parameter*.

Col 4 (int) - *element-type* = -40

Col 5 (float) - *dummy parameter*.

Col 6 (float) - *longitudinal magnetic field strength* [T].

Col 7 (float) - *front/back side flag* [1]. **0** for front, **1** for back side.

4.2.14 lattice end flag (`type` : -99)

End flag of the input lattice.

`parameter-array`

Col 1 (float) - *dummy parameter*.

Col 2 (int) - *dummy parameter*.

Col 3 (int) - *dummy parameter*.

Col 4 (int) - *element-type* = -99

4.3 Synchronous phase input for RF cavities

For the *coupled cavity linac (type:103)* and *em field (type:110)*, input phase is switchable between *drive phase* ϕ_d and *synchronous phase* ϕ_s .

In the case of input phase is *synchronous phase*, -90 [deg] is stable and no energy gain point. Inside the code, the rf field is calculated by using driven phase, and the synchronous phase offset ϕ_{off} is calculated automatically which defined as

$$\phi_d = \phi_s + \phi_{\text{off}}.$$

This offset value is defined by using the reference particle, and the reference particle does not depend on any rf errors and alignment errors. In addition, it is cached in memory and reuse if the reference energy and charge to mass ratio at the element entrance are the same as the previous simulation.

The rf phase error ϕ_{error} is directly added to the driven phase. Basically, the energy gain of the whole beam is not zero even in the synchronous phase is -90 due to the energy deviation.

4.4 External data format

4.4.1 rf cavity data

(Use for *drift tube linac*, *coupled cavity dtl*, *coupled cavity linac*, *superconducting cavity*, *solenoid with rf field*)

In case of `Flaginteg` equals to **1**, the data file (rfdata${file_id}) should have on-axis longitudinal electric field data.

```
z(1)    Ez(1)    Ez'(1)    Ez''(1)
z(2)    Ez(2)    Ez'(2)    Ez''(2)
z(3)    Ez(3)    Ez'(3)    Ez''(3)
.
.
! max data length is 5000
```

In case of `Flaginteg` equals to **2**, the data file (rfdata${file_id}) should have Fourier coefficients of on-axis longitudinal electric field.

```
F(1)
F(2)
F(3)
.
.
! max data length is 101
```

4.4.2 3d field data

(Use for *em field*, *em field dipole*)

This data can be used in case of `Flaginteg` equals to **2**. The data file (1T${file_id}.T7) should have 6D electromagnetic field data and its size information.

```
x_min  x_max  x_ngrid
y_min  y_max  y_ngrid
z_min  z_max  z_ngrid
Ex   Ey   Ez   Bx   By   Bz   ! at (x(1), y(1), z(1))
Ex   Ey   Ez   Bx   By   Bz   ! at (x(2), y(1), z(1))
Ex   Ey   Ez   Bx   By   Bz   ! at (x(3), y(1), z(1))
.
.
Ex   Ey   Ez   Bx   By   Bz   ! at (x(1), y(2), z(1))
Ex   Ey   Ez   Bx   By   Bz   ! at (x(2), y(2), z(1))
Ex   Ey   Ez   Bx   By   Bz   ! at (x(3), y(2), z(1))
.
.
Ex   Ey   Ez   Bx   By   Bz   ! at (x(1), y(3), z(1))
Ex   Ey   Ez   Bx   By   Bz   ! at (x(2), y(3), z(1))
Ex   Ey   Ez   Bx   By   Bz   ! at (x(3), y(3), z(1))
.
.
Ex   Ey   Ez   Bx   By   Bz   ! at (x(1), y(1), z(2))
Ex   Ey   Ez   Bx   By   Bz   ! at (x(2), y(1), z(2))
Ex   Ey   Ez   Bx   By   Bz   ! at (x(3), y(1), z(2))
.
.
! data length is (x_ngrid+1)*(y_ngrid+1)*(z_ngrid+1)
```

For multi-field (in *em field*) data,

```
x_min  x_max  x_ngrid
y_min  y_max  y_ngrid
z_min  z_max  z_ngrid
E1x  E1y  E1z  B1x  B1y  B1z  E2x  E2y  E2z  B2x  B2y  B2z
.
.
! data length is (x_ngrid+1)*(y_ngrid+1)*(z_ngrid+1)
```

or,

```
x_min  x_max  x_ngrid
y_min  y_max  y_ngrid
z_min  z_max  z_ngrid
E1x  E1y  E1z  B1x  B1y  B1z  E2x  E2y  E2z  B2x  B2y  B2z  E3x  E3y  E3z  B3x  B3y
↪B3z
.
.
! data length is (x_ngrid+1)*(y_ngrid+1)*(z_ngrid+1)
```

formats are available.

4.4.3 rfq data

(Use for *rf quadrupole*)

This data can be used in case of `Flaginteg` equals to **2**.

The data file (rfqline${file_id}) should have the list of the RFQ structure constants.

```
Len Nseg Nstp Ctype Scale Phi1 Radius Mod R0 A10 A0 A12I4 A1 A30I0 A21I2 A32I4 A23I6 /
```

`structure constants`

Len (float) - *cell length* [m].

Nseg (int) - *number of element segmentations* [1].

Nstp (int) - *number of map steps for PIC calculation* [1].

Ctype (int) - *cell type* [1].

cell-type	value
2 term XY potential for standard acceleration	**0**
8 term XY potential for standard acceleration	**1**
radial matching	**2**
exit transition (inverse)	**3**
exit transition (normal)	**4**
exit matching	**5**
exit fringe	**6**

Scale (float) - *scaling factor* [1].

Phi1 (float) - *synchronous phase* [deg].

Radius (float) - *pipe radius* [m].

Mod (float) - *modulation* [1].

R0 (float) - *middle point radius* [m].

A10 (float) - *acceleration term* [1].

A0 (float) - *quadrupole term* [1].

A1 (float) - *duodecapole term* [1].

A12I4, A30I0, A21I2, A32I4, A23I6 (float) - *product of multipole coefficient and Bessel function* [1].

Note: In this RFQ model is based on Thomas P. Wangler's "RF Linear Accelerator" book.

The eight-term potential function is,

$$\begin{aligned} U(r,\theta,z) = \frac{V}{2}\Big\{ & A_0 \left(\frac{r}{r_0}\right)^2 \cos 2\theta + A_1 \left(\frac{r}{r_0}\right)^6 \cos 6\theta \\ & + A_{10} I_0(kr) \cos kz + A_{30} I_0(3kr) \cos 3kz \\ & + [A_{12} I_4(kr) \cos kz + A_{32} I_4(3kr) \cos 3kz] \cos 4\theta \\ & + [A_{21} I_2(2kr) \cos 2\theta + A_{23} I_6(2kr) \cos 6\theta] \cos 2kz \Big\} \end{aligned}$$

here, V is applied voltage, $k = \pi/L$ (where L is the length of the cell), and I_n is n-th kind Bessel function.

CHAPTER

FIVE

CLASS LIBRARY

5.1 Sequence class – run control API

class impact.control.**Sequence**(*input=None*, *subdir=None*)

Bases: *impact.input.Input*, *impact.result.Result*

Main class of python API for Advanced IMPACT code.

- **Attribute**

subdir	str: Directory path for data files.
seed	int: Seed value of pseudo random number generator.

- **Methods**

read([fname, debug])	Read input file and set simulation parameters.
load([dname, reload])	Load 3D field data
distribute([particles, unit, dname, fname])	Load 3D field data
run([start, end, zstart, Phaseini])	Run particle tracking simulation.
construct(lattice)	Construct beam transport line.
insert(index, element)	Insert new lattice element.
conf([index, col, value, syntax])	Configure lattice parameters.
search([type, tag])	Search lattice element.
output([fname, qid])	Output original IMPACT results to file.
save([fname])	Save current setting to file.
checker()	Check input parameter correctness.

Parameters **input** : str (optional)

Input file path.

subdir : str (optional)

Directory path for data files.

subdir

str: Directory path for data files.

seed

int: Seed value of pseudo random number generator.

read(*fname=None*, *debug=False*)

Read input file and set simulation parameters.

Parameters **fname** : str

Input file path.

debug : bool

Flag for debug mode. If True, skip over input parameter correctness checker.

Notes

This function is automatically called in case of `input` is defined in *Sequence*.

load(*dname=None*, *reload=False*)
Load 3D field data

Parameters **dname** : str (optional)

Directory path for data files. Default path is defined by *subdir*.

reload : bool

If true, force to reload all 3D field data.

distribute(*particles=None*, *unit=u'physical'*, *dname=None*, *fname=u'partcl.data'*, ***kws*)
Load 3D field data

Parameters **particles** : (9, n) shape ndarray (optional)

All particle information for the simulation. *n* is the number of particles.

- row 1 : x positions [m]
- row 2 : xp momentums [rad]
- row 3 : y positions [m]
- row 4 : yp momentums [rad]
- row 5 : z positions [deg]
- row 6 : zp momentums [MeV]
- row 7 : charge to mass ratios [c^2/eV]
- row 8 : charge per macro-particle weights [1]
- row 9 : ID numbers [1]

unit : str (optional)

Unit information for *particles*.

- 'physical' : Use physical units [m, rad, m, rad, deg, MeV]. (default)
- 'impact' : Use impact internal units.

dname : str (optional)

Directory path for particle data file. Default path is defined by *subdir*.

fname : str (optional)

File name or path for particle data file. Default is 'partcl.data'

Mass : float (optional)

Mass of the reference particle (unit is [eV/c^2]).

Energy : float (optional)

Initial kinetic energy (unit is [eV]).

Frequency : float (optional)

Scaling frequency (unit is [Hz]).

zstart : float (optional)

Starting z point of the simulation.

Phaseini : float (optional)

Initial phase of the reference particle (unit is [rad]).

Notes

`dname` and `fname` are used in case of *`Disttype`* equals to 19.

`run` (*start=0*, *end=-1*, *zstart=None*, *Phaseini=None*)
Run particle tracking simulation.

Parameters **start** : int or str (optional)

Starting element of the particle tracking simulation.

end : int or str (optional)

Ending element of the particle tracking simulation.

zstart : float (optional)

Initial z position of the reference particle. Default is 0.0 .

Phaseini : float (optional)

Initial phase of the reference particle (unit is [rad]).

`construct` (*lattice*)
Construct beam transport line.

Parameters **lattice** : 2D array or list of dict

List of lattice elements.

Examples

```
>>> drift = [0.5, 10, 10, 0, 0.2]
# define element by using list of numbers (format is the same as the input␣
↪file)
>>> quad1 = {'length':0.2, 'seg':4, 'step':4, 'type':'quadrupole', 'B2': 10.0,
↪ 'aper':0.2}
# define element by using python dictionary
>>> quad2 = quad1.copy()
>>> quad2['B2'] = -10.0
>>> FODO = [drift, quad1, drift, quad2, drift]
>>> sq.construct(FODO)
```

`insert` (*index*, *element*)
Insert new lattice element.

Parameters **index** : int

Insert new lattice element before index. If index equals to 0, *append* the new element.

element : list or dict

Lattice element information.

Examples

```
>>> quad1 = {'length':0.2, 'seg':4, 'step':4, 'type':'quadrupole', 'B2': 10.0,
↪ 'aper':0.2}
# define element by using python dictionary
>>> sq.construct(3, quad1) # insert new element
```

conf(*index=None*, *col=None*, *value=None*, *syntax=True*)

Configure lattice parameters.

Parameters **index** : int or str

Index number of the element or tag name of the element.

col : int or str

Column number of the target parameter or name of the target parameter.

col : dict

dict key for the target parameter and dict value for target value.

value : float

New value for the target parameter.

syntax : bool

Print flag for full parameter of the element.

Returns dict

Dictionary style parameter list of the element

Examples

```
>>> sq = Sequence('test.in')
>>> tmp = sq.conf(3) # show parameter of 3rd element
index :  3
length :  0.24 [m]
segment :  60
step :  20
type :  110  (emfld)
scaling ('scale', 'scl_fac') :  0.64
frequency ('f', 'freq') :  80500000.0 [Hz]
driven phase ('phi0',) :  349.74086645 [deg]
fileid ('file', 'id') :  421.0
xaperture ('xaper', 'xpipe', 'xradius') :  0.017 [m]
yaperture ('yaper', 'ypipe', 'yradius') :  0.017 [m]
dx  :  0.0 [m]
dy  :  0.0 [m]
pitch  :  0.0 [rad]
```

```
yaw  :  0.0 [rad]
roll  :  0.0 [rad]
data type ('data',) :  1.0
coordinate ('coor',) :  2.0
tag :  ['cav1']
>>> sq.conf(3, {'phi0':360.0}) # set new value for the driven phase
>>> tmp = sq.conf(3, 'phi0') # confirm new parameter
driven phase ('phi0',) :  360.0 [deg]
```

If you have set the tag, followings return same results.

```
>>> sq.conf('cav1') # show parameter of 3rd element
>>> sq.conf('cav1', {'phi0':360.0}) # set new value for the driven phase
```

search(*type=None*, *tag=None*)
Search lattice element.

Parameters **type** : int or str

Element type ID or type name for _search.

tag : str

Tag name of the element.

Returns ndarray

List of element ID

Notes

In case of input both type and tag, it returns *AND* _search result.

output(*fname=u'fort'*, *qid=-1*)
Output original IMPACT results to file.

Parameters **fname** : str

Header name of output file. Default is 'fort'.

qid : int

Charge state index for return value. Default is -1 (Total of the all charge states). Index is compatible with *qmlabel*.

save(*fname=u'test.in.new'*)
Save current setting to file.

Parameters **fname** : str

File name for saved settings. Default is 'test.in.new'.

checker()
Check input parameter correctness.

5.2 Input class / base of Sequence class

class impact.input.**Input**
Bases: object

Base class for input parameters of Advanced IMPACT.

- **Attributes for simulation parameter**

Mpisize	int: Number of processors in MPI communicator.
Ndim	int: Dimension of the simulation space (dummy parameter).
Nxgrid	int: Number of x mesh grid for space charge (PIC) calculation.
Nygrid	int: Number of y mesh grid for space charge (PIC) calculation.
Nzgrid	int: Number of z mesh grid for space charge (PIC) calculation.
Xbound	float: x (horizontal) pipe width for space charge (PIC) calculation (unit is [m]).
Ybound	float: y (vertical) pipe width for space charge (PIC) calculation (unit is [m]).
Zperiod	float: z periodic length for space charge (PIC) calculation (unit is [m]).
Flaginteg	int: Flag for integration method. 1 for linear map, 2 for nonlinear Lorentz integrator.
Flagerror	int: Flag for error study.
Flagdiag	int: Flag for diagnostic routine. 1 or 3 for standard, 2 or 4 for n % emittance, radius output.
Flagbc	int: Flag of boundary condition for space charge (PIC) calculation.
Flagsubstep	bool: Flag of sub-cycle for space charge calculation.
Flagrestart	bool: Flag for restarting the simulation (dummy parameter).
Flagsc	bool: Flag of space-charge calculation.
Outputemit	float: Percentage for emittance and radius output (unit is [%]).

- **Attributes for beam parameter**

Nptot	int: Total number of macro particles.
Ncharge	int: Number of charge states
Nplist	list of int: List of particle number for each charge state.
Curlist	list of float: List of beam current for each charge state. (unit is [A])
Ctmlist	list of float: List of charge to mass ratio for each charge state. (unit is [c^2/eV])
Energy	float: Initial kinetic energy (unit is [eV]).
Mass	float: Mass of the reference particle (unit is [eV/c^2]).
Frequency	float: Scaling frequency (unit is [Hz]).
Chargestate	float: Charge state of the reference particle (unit is [1]).
Phaseini	float: Initial phase of the reference particle (unit is [rad]).
Disttype	int: Initial distribution type.
Distxsigma	float: x standard deviation of the initial distribution (IMPACT internal unit).

Continued on next page

Table 5.4 – continued from previous page

Distysigma	float: y standard deviation of the initial distribution (IMPACT internal unit).
Distzsigma	float: z standard deviation of the initial distribution (IMPACT internal unit).
Distxlambda	float: x' (xp, px) standard deviation of the initial distribution (IMPACT internal unit).
Distylambda	float: y' (yp, py) standard deviation of the initial distribution (IMPACT internal unit).
Distzlambda	float: z' (zp, pz) standard deviation of the initial distribution (IMPACT internal unit).
Distxmu	float: x-x' coupling term of the initial distribution (IMPACT internal unit).
Distymu	float: y-y' coupling term of the initial distribution (IMPACT internal unit).
Distzmu	float: z-z' coupling term of the initial distribution (IMPACT internal unit).
Distxtwiss	list of float: x twiss parameter for the initial distribution.
Distytwiss	list of float: y twiss parameter for the initial distribution.
Distztwiss	list of float: z twiss parameter for the initial distribution.
Distxquadratic	list of float: x quadratic parameter (IMPACT default) for the initial distribution.
Distyquadratic	list of float: y quadratic parameter (IMPACT default) for the initial distribution.
Distzquadratic	list of float: z quadratic parameter (IMPACT default) for the initial distribution.
Distxoffset	float: x position offset of the initial distribution (IMPACT internal unit).
Distyoffset	float: y position offset of the initial distribution (IMPACT internal unit).
Distzoffset	float: z position offset of the initial distribution (IMPACT internal unit).
Distxpoffset	float: x' (xp, px) momentum offset of the initial distribution (IMPACT internal unit).
Distypoffset	float: y' (yp, py) momentum offset of the initial distribution (IMPACT internal unit).
Distzpoffset	float: z' (zp, pz) offset of the initial distribution (IMPACT internal unit).
Distxoffset_m	float: x position offset of the initial distribution (unit is [m]).
Distyoffset_m	float: y position offset of the initial distribution (unit is [m]).
Distzoffset_deg	float: z position offset of the initial distribution (unit is [deg]).
Distxpoffset_rad	float: x' (xp, px) momentum offset of the initial distribution (unit is [rad]).
Distypoffset_rad	float: y' (yp, py) momentum offset of the initial distribution (unit is [rad]).
Distzpoffset_eV	float: z' (zp, pz) offset of the initial distribution (unit is [eV]).
Distxmismatch	float: x mismatch factor of the initial distribution (unit is [1]).

Continued on next page

Table 5.4 – continued from previous page

Distymismatch	float: y mismatch factor of the initial distribution (unit is [1]).
Distzmismatch	float: z mismatch factor of the initial distribution (unit is [1]).
Distxpmismatch	float: x' (xp, px) mismatch factor of the initial distribution (unit is [1]).
Distypmismatch	float: x' (xp, px) mismatch factor of the initial distribution (unit is [1]).
Distzpmismatch	float: z' (zp, pz) mismatch factor of the initial distribution (unit is [1]).

Mpisize
int: Number of processors in MPI communicator.

Ndim
int: Dimension of the simulation space (dummy parameter).

Nptot
int: Total number of macro particles.

Flaginteg
int: Flag for integration method. 1 for linear map, 2 for nonlinear Lorentz integrator.

Flagerror
int: Flag for error study.

Notes

- 0 : No error simulation
- 1 : Use `dx`, `dy`, `pitch`, `yaw` and `roll` errors to the lattice element.
- 2 : Use entrance offsets (`dx0`, `dy0`), exit offsets (`dx1`, `dy1`), and `roll` errors to the lattice element.

Flagdiag
int: Flag for diagnostic routine. 1 or 3 for standard, 2 or 4 for *n* % emittance, radius output.

Notes

- 1 or 2 : Diagnostic routine is called for each segment.
- 3 or 4 : Diagnostic routine is called in case of element type = -23 (monitor).

Nxgrid
int: Number of x mesh grid for space charge (PIC) calculation.

Notes

The x mesh number must be 2^n (for *Flagbc* equals to 1, 2, 3, or 4) or $2^n + 1$ (for *Flagbc* equals to 5 or 6).

Nygrid
int: Number of y mesh grid for space charge (PIC) calculation.

Notes

The y mesh number must be 2^n (for *Flagbc* equals to 1 or 2) or $2^n + 1$ (for *Flagbc* equals to 3, 4, 5, or 6).

Nzgrid
int: Number of z mesh grid for space charge (PIC) calculation.

Notes

The z mesh number must be 2^n (for *Flagbc* equals to 1, 3, or 5) or $2^n + 1$ (for *Flagbc* equals to 2, 4, or 6).

Flagbc
int: Flag of boundary condition for space charge (PIC) calculation.

Notes

- 1 : 3D open. In this case, all mesh number (*Nxgrid*, *Nygrid*, *Nzgrid*) must be 2^n.
- 2 : Transverse open, longitudinal periodic. In this case, *Nxgrid* and *Nygrid* must be 2^n, *Nzgrid* must be $2^n + 1$.
- 3 : Transverse finite, longitudinal open round pipe. In this case, *Nxgrid* and *Nzgrid* must be 2^n, *Nygrid* must be $2^n + 1$.
- 4 : Transverse finite, longitudinal periodic round pipe. In this case *Nxgrid* must be 2^n, *Nygrid* and *Nzgrid* must be $2^n + 1$.
- 5 : Transverse finite, longitudinal open rectangular pipe. In this case, *Nzgrid* must be 2^n, *Nxgrid* and *Nygrid* must be $2^n + 1$.
- 6 : Transverse finite, longitudinal periodic rectangular pipe. In this case, all mesh number (*Nxgrid*, *Nygrid*, *Nzgrid*) must be $2^n + 1$.

Xbound
float: x (horizontal) pipe width for space charge (PIC) calculation (unit is [m]).

Ybound
float: y (vertical) pipe width for space charge (PIC) calculation (unit is [m]).

Zperiod
float: z periodic length for space charge (PIC) calculation (unit is [m]).

Flagrestart
bool: Flag for restarting the simulation (dummy parameter).

Flagsubstep
bool: Flag of sub-cycle for space charge calculation.

Ncharge
int: Number of charge states

Nplist
list of int: List of particle number for each charge state.

Curlist
list of float: List of beam current for each charge state. (unit is [A])

Ctmlist
list of float: List of charge to mass ratio for each charge state. (unit is [c^2/eV])

Flagsc
bool: Flag of space-charge calculation.

Energy
float: Initial kinetic energy (unit is [eV]).

Mass
float: Mass of the reference particle (unit is [eV/c^2]).

Chargestate
float: Charge state of the reference particle (unit is [1]).

Frequency
float: Scaling frequency (unit is [Hz]).

Phaseini
float: Initial phase of the reference particle (unit is [rad]).

Outputemit
float: Percentage for emittance and radius output (unit is [%]).

Notes

This value is used in case of *Flagdiag* equals to 2 or 4.

Disttype
int: Initial distribution type.

Notes

- 1 : 6D rectangle in phase space for single charge state.
- 2 : 6D Gaussian distribution for single charge state.
- 3 : 6D Waterbag distribution for single charge state.
- 4 : Semi-Gaussian distribution for single charge state.
- 5 : KV distribution for single charge state.
- 16 : 6D Waterbag distribution for multi charge states (recommended).
- 17 : 6D Gaussian distribution for multi charge states (recommended).
- 19 : User input distribution (recommended). File path is defined in *distribute*.

Distxsigma
float: x standard deviation of the initial distribution (IMPACT internal unit).

Distxlambda
float: x' (xp, px) standard deviation of the initial distribution (IMPACT internal unit).

Distxmu
float: x-x' coupling term of the initial distribution (IMPACT internal unit).

Distysigma
float: y standard deviation of the initial distribution (IMPACT internal unit).

Distylambda
float: y' (yp, py) standard deviation of the initial distribution (IMPACT internal unit).

Distymu
float: y-y' coupling term of the initial distribution (IMPACT internal unit).

Distzsigma
float: z standard deviation of the initial distribution (IMPACT internal unit).

Distzlambda
float: z' (zp, pz) standard deviation of the initial distribution (IMPACT internal unit).

Distzmu
float: z-z' coupling term of the initial distribution (IMPACT internal unit).

Distxmismatch
float: x mismatch factor of the initial distribution (unit is [1]).

Distxpmismatch
float: x' (xp, px) mismatch factor of the initial distribution (unit is [1]).

Distxoffset
float: x position offset of the initial distribution (IMPACT internal unit).

Distxpoffset
float: x' (xp, px) momentum offset of the initial distribution (IMPACT internal unit).

Distymismatch
float: y mismatch factor of the initial distribution (unit is [1]).

Distypmismatch
float: x' (xp, px) mismatch factor of the initial distribution (unit is [1]).

Distyoffset
float: y position offset of the initial distribution (IMPACT internal unit).

Distypoffset
float: y' (yp, py) momentum offset of the initial distribution (IMPACT internal unit).

Distzmismatch
float: z mismatch factor of the initial distribution (unit is [1]).

Distzpmismatch
float: z' (zp, pz) mismatch factor of the initial distribution (unit is [1]).

Distzoffset
float: z position offset of the initial distribution (IMPACT internal unit).

Distzpoffset
float: z' (zp, pz) offset of the initial distribution (IMPACT internal unit).

Distxtwiss
list of float: x twiss parameter for the initial distribution.

List of input parameters and units is [alpha(1), beta(m), normalized emittance(m-rad)].

Notes

Input list length must be 3.

keywords: Courant Snyder parameters, horizontal

Distxquadratic
list of float: x quadratic parameter (IMPACT default) for the initial distribution.

List of input parameters is [sigma, lambda, mu], IMPACT internal units.

Notes

Input list length must be 3.

keywords: horizontal

Distytwiss
list of float: y twiss parameter for the initial distribution.

List of input parameters and units is [alpha(1), beta(m), normalized emittance(m-rad)].

Notes

Input list length must be 3.

keywords: Courant Snyder parameters, vertical

Distyquadratic
list of float: y quadratic parameter (IMPACT default) for the initial distribution.

List of input parameters is [sigma, lambda, mu], IMPACT internal units.

Notes

Input list length must be 3.

keywords: vertical

Distztwiss
list of float: z twiss parameter for the initial distribution.

List of input parameters and units is [alpha(1), beta(m), normalized emittance(m-rad)].

Notes

Input list length must be 3.

keywords: Courant Snyder parameters, longitudinal

Distzquadratic
list of float: z quadratic parameter (IMPACT default) for the initial distribution.

List of input parameters is [sigma, lambda, mu], IMPACT internal units.

Notes

Input list length must be 3.

keywords: vertical

Distxoffset_m
float: x position offset of the initial distribution (unit is [m]).

Distxpoffset_rad
float: x' (xp, px) momentum offset of the initial distribution (unit is [rad]).

Distyoffset_m
float: y position offset of the initial distribution (unit is [m]).

Distypoffset_rad
float: y' (yp, py) momentum offset of the initial distribution (unit is [rad]).

Distzoffset_deg
float: z position offset of the initial distribution (unit is [deg]).

Distzpoffset_eV
float: z' (zp, pz) offset of the initial distribution (unit is [eV]).

5.3 Result class / base of Sequence class

class impact.result.**Result**
Bases: object

Base class for simulation results of Advanced IMPACT.

- **Attributes**

dtag	dict: Tag dictionary of lattice elements.
autobcast	bool: Flag of auto MPI bcast for simulation results.
qmlabel	list: List of charge to mass ratios for history results

- **Methods for history results**

hrefz(*args, **kws)	Returns z distance history of the reference particle (default unit is [m]).
hrefphi(*args, **kws)	Returns absolute phase history of the reference particle (default unit is [rad]).
hrefgma(*args, **kws)	Returns Lorentz gamma history of the reference particle (default unit is [1]).
hrefeng(*args, **kws)	Returns kinetic energy history of the reference particle (default unit is [MeV]).
hrefbeta(*args, **kws)	Returns Lorentz beta history of the reference particle (default unit is [1]).
hrefr(*args, **kws)	Returns maximum r distance history of particles from pipe center (default unit is [m]).
hxcen(*args, **kws)	Returns x centroid position history (default unit is [m]).
hycen(*args, **kws)	Returns y centroid position history (default unit is [m]).
hzcen(*args, **kws)	Returns z (phase) centroid position history (default unit is [deg]).
hxrms(*args, **kws)	Returns x rms size history (default unit is [m]).
hyrms(*args, **kws)	Returns y rms size history (default unit is [m]).
hzrms(*args, **kws)	Returns z (phase) rms size history (default unit is [deg]).
hxpcen(*args, **kws)	Returns x'(xp, px) centroid momentum history (default unit is [rad]).

Continued on next page

Table 5.6 – continued from previous page

hypcen(*args, **kws)	Returns y'(yp, py) centroid momentum history (default unit is [rad]).
hzpcen(*args, **kws)	Returns z'(zp, pz) centroid history (default unit is [MeV]).
hxprms(*args, **kws)	Returns x'(xp, px) rms momentum history (default unit is [rad]).
hyprms(*args, **kws)	Returns y'(yp, py) rms momentum history (default unit is [rad]).
hzprms(*args, **kws)	Returns z'(zp, pz) rms spread history (default unit is [MeV]).
hxtwsa(*args, **kws)	Returns x twiss parameter alpha history (default unit is [1]).
hytwsa(*args, **kws)	Returns y twiss parameter alpha history (default unit is [1]).
hztwsa(*args, **kws)	Returns z twiss parameter alpha history (default unit is [1]).
hxtwsb(*args, **kws)	Returns x twiss parameter beta history (default unit is [m]).
hytwsb(*args, **kws)	Returns y twiss parameter beta history (default unit is [m]).
hztwsb(*args, **kws)	Returns z twiss parameter beta history (default unit is [m]).
hxepsn(*args, **kws)	Returns x normalized emittance history (default unit is [m-rad]).
hyepsn(*args, **kws)	Returns y normalized emittance history (default unit is [m-rad]).
hzepsn(*args, **kws)	Returns z normalized emittance history (default unit is [m-rad]).
hxepsnp(*args, **kws)	Returns *n* % x normalized emittance history (default: *n* = 99.9, unit is [m-rad]).
hyepsnp(*args, **kws)	Returns *n* % y normalized emittance history (default: *n* = 99.9, unit is [m-rad]).
hzepsnp(*args, **kws)	Returns *n* % z normalized emittance history (default: *n* = 99.9, unit is [m-rad]).
hxepsnf(*args, **kws)	Returns full x normalized emittance history (default unit is [m-rad]).
hyepsnf(*args, **kws)	Returns full y normalized emittance history (default unit is [m-rad]).
hzepsnf(*args, **kws)	Returns full z normalized emittance history (default unit is [m-rad]).
hxmax(*args, **kws)	Returns maximum x position history (default unit is [m]).
hymax(*args, **kws)	Returns maximum y position history (default unit is [m]).
hzmax(*args, **kws)	Returns maximum z position (phase) history (default unit is [deg]).
hxpmax(*args, **kws)	Returns maximum x' (xp, px) momentum history (default unit is [rad]).
hypmax(*args, **kws)	Returns maximum y' (yp, py) momentum history (default unit is [rad]).

Continued on next page

Table 5.6 – continued from previous page

hzpmax(*args, **kws)	Returns maximum z' (zp, pz) history (default unit is [MeV]).
hrrms(*args, **kws)	Returns r rms history (default unit is [m]).
hrnpr(*args, **kws)	Returns n % r size history (default: n = 99.9) (default unit is [m]).
hrmax(*args, **kws)	Returns maximum r distance history (default unit is [m]).
hx3rd(*args, **kws)	Returns x 3rd root of 3rd moment history (default unit is [m]).
hy3rd(*args, **kws)	Returns y 3rd root of 3rd moment history (default unit is [m]).
hz3rd(*args, **kws)	Returns z (phase) 3rd root of 3rd moment history (default unit is [deg]).
hxp3rd(*args, **kws)	Returns x' (xp, px) 3rd root of 3rd moment history (default unit is [rad]).
hyp3rd(*args, **kws)	Returns y' (yp, py) 3rd root of 3rd moment history (default unit is [rad]).
hzp3rd(*args, **kws)	Returns z' (zp, pz) 3rd root of 3rd moment history (default unit is [MeV]).
hx4th(*args, **kws)	Returns x 4th root of 4th moment history (default unit is [m]).
hy4th(*args, **kws)	Returns y 4th root of 4th moment history (default unit is [m]).
hzp4th(*args, **kws)	Returns z' (zp, pz) 4th root of 4th moment history (default unit is [MeV]).
hxp4th(*args, **kws)	Returns x' (xp, px) 4th root of 4th moment history (default unit is [rad]).
hyp4th(*args, **kws)	Returns y' (yp, py) 4th root of 4th moment history (default unit is [rad]).
hz4th(*args, **kws)	Returns z (phase) 4th root of 4th moment history (default unit is [deg]).
hlpmin(*args, **kws)	Returns minimum local particle number history (default unit is [1]).
hlpmax(*args, **kws)	Returns maximum local particle number history (default unit is [1]).
hptot(*args, **kws)	Returns total particle number history (default unit is [1]).

- **Methods for distribution results**

getx(*args, **kws)	Returns x positions of the particles (default unit is [m]).
getxp(*args, **kws)	Returns x' (xp, px) momentums of the particles (default unit is [rad]).
gety(*args, **kws)	Returns y positions of the particles (default unit is [m]).
getyp(*args, **kws)	Returns y' (yp, py) momentums of the particles (default unit is [rad]).
getz(*args, **kws)	Returns z positions (phase) of the particles (default unit is [deg]).
getzp(*args, **kws)	Returns z' (zp, pz) of the particles (default unit is [MeV]).

Continued on next page

Table 5.7 – continued from previous page

getctm(*args, **kws)	Returns charge to mass ratio of the particles (default unit is [c^2/eV]).
getcpmp(*args, **kws)	Returns charge per macro particle of the particles (default unit is [1]).
getid(*args, **kws)	Returns ID number of the particles (default unit is [1]).
getall(*args, **kws)	Returns full information of the particles).

dtag
dict: Tag dictionary of lattice elements.

autobcast
bool: Flag of auto MPI bcast for simulation results.

qmlabel
list: List of charge to mass ratios for history results

hrefz (**args*, ***kws*)
Returns z distance history of the reference particle (default unit is [m]).

Parameters ***args :**

str – Tag name or physical unit.

float or list of float – Substitution value of interpolation function.

****kws :**

tag : str

Tag name of the target position.

unit : str

Physical unit name of the return value.

value : float or list of float

Substitution value for the interpolation function.

qid : int

Charge state index for return value. Default is -1 (Total of the all charge states). Index is compatible with *qmlabel*.

at : `entry`, `mid`, or `exit` (default)

Data taking location in the target element. Used when the tag name is given.

Returns ndarray

z distance history

Notes

keywords: longitudinal, fort.18

hrefphi (**args*, ***kws*)
Returns absolute phase history of the reference particle (default unit is [rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

absolute phase history

Notes

keywords: fort.18

hrefbeta (**args*, ***kws*)

Returns Lorentz beta history of the reference particle (default unit is [1]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

Lorentz beta history

Notes

keywords: fort.18

hrefgma (**args*, ***kws*)

Returns Lorentz gamma history of the reference particle (default unit is [1]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

Lorentz gamma history

Notes

keywords: fort.18

hrefeng (**args*, ***kws*)

Returns kinetic energy history of the reference particle (default unit is [MeV]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

kinetic energy history

Notes

keywords: fort.18

hrefr (**args*, ***kws*)
Returns maximum r distance history of particles from pipe center (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

maximum r distance history

Notes

keywords: fort.18

hxcen (**args*, ***kws*)
Returns x centroid position history (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

x centroid position history

Notes

keywords: horizontal, fort.24

hxrms (**args*, ***kws*)
Returns x rms size history (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

x rms size history

Notes

keywords: root mean square, envelope, horizontal, fort.24

hxpcen (**args*, ***kws*)
Returns x'(xp, px) centroid momentum history (default unit is [rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

x' centroid momentum history

Notes

keywords: horizontal, fort.24

hxprms (**args*, ***kws*)
Returns x'(xp, px) rms momentum history (default unit is [rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

x' rms momentum history

Notes

keywords: root mean square, envelope, horizontal, fort.24

hxtwsa (**args*, ***kws*)
Returns x twiss parameter alpha history (default unit is [1]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

x twiss parameter alpha history

Notes

keywords: Courant Snyder parameters, horizontal, fort.24

hxtwsb (**args*, ***kws*)
Returns x twiss parameter beta history (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

x twiss parameter beta history

Notes

keywords: Courant Snyder parameters, horizontal, fort.24

hxepsn (**args*, ***kws*)
Returns x normalized emittance history (default unit is [m-rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

x normalized emittance history

Notes

keywords: horizontal, fort.24

hxepsnp (**args*, ***kws*)
Returns *n* % x normalized emittance history (default: *n* = 99.9, unit is [m-rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

n % x normalized emittance history

Notes

This value is calculated in case of *Flagdiag* equals to 2 or 4.

The percentage *n* [%] is defined by *Outputemit*.

keywords: horizontal, fort.24

hxepsnf (**args*, ***kws*)
Returns full x normalized emittance history (default unit is [m-rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

full x normalized emittance history

Notes

This value is calculated in case of *Flagdiag* equals to 2 or 4.

keywords: horizontal, fort.24

hycen (**args*, ***kws*)
Returns y centroid position history (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

y centroid position history

Notes

keywords: vertical, fort.25

hyrms (**args*, ***kws*)
Returns y rms size history (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

y rms size history

Notes

keywords: root mean square, envelope, vertical, fort.25

hypcen (**args*, ***kws*)
Returns y'(yp, py) centroid momentum history (default unit is [rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

y' centroid momentum history

Notes

keywords: vertical, fort.25

hyprms (**args*, ***kws*)
Returns y'(yp, py) rms momentum history (default unit is [rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

y' rms momentum history

Notes

keywords: root mean square, envelope, vertical, fort.25

hytwsa (**args*, ***kws*)
Returns y twiss parameter alpha history (default unit is [1]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

y twiss parameter alpha history

Notes

keywords: Courant Snyder parameters, vertical, fort.25

hytwsb (**args*, ***kws*)
Returns y twiss parameter beta history (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

y twiss parameter beta history

Notes

keywords: Courant Snyder parameters, vertical, fort.25

hyepsn (**args*, ***kws*)
Returns y normalized emittance history (default unit is [m-rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

y normalized emittance history

Notes

keywords: vertical, fort.25

hyepsnp (**args*, ***kws*)
Returns *n* % y normalized emittance history (default: *n* = 99.9, unit is [m-rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

n % y normalized emittance history

Notes

This value is calculated in case of *Flagdiag* equals to 2 or 4.

The percentage *n* [%] is defined by *Outputemit*.

keywords: vertical, fort.25

hyepsnf (**args*, ***kws*)
Returns full y normalized emittance history (default unit is [m-rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

full y normalized emittance history

Notes

This value is calculated in case of *Flagdiag* equals to 2 or 4.

keywords: vertical, fort.25

hzcen (**args*, ***kws*)
Returns z (phase) centroid position history (default unit is [deg]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

z centroid position history

Notes

keywords: longitudinal, fort.26

hzrms (**args*, ***kws*)
Returns z (phase) rms size history (default unit is [deg]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

z rms size history

Notes

keywords: root mean square, envelope, longitudinal, fort.26

hzpcen (**args*, ***kws*)
Returns z'(zp, pz) centroid history (default unit is [MeV]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

z' centroid history

Notes

keywords: longitudinal, energy deviation, fort.26

hzprms (**args*, ***kws*)
Returns z'(zp, pz) rms spread history (default unit is [MeV]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

z' rms spread history

Notes

keywords: root mean square, envelope, longitudinal, energy deviation, fort.26

hztwsa (**args*, ***kws*)
Returns z twiss parameter alpha history (default unit is [1]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

z twiss parameter alpha history

Notes

keywords: Courant Snyder parameters, longitudinal, fort.26

hztwsb (**args*, ***kws*)
Returns z twiss parameter beta history (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

z twiss parameter beta history

Notes

keywords: Courant Snyder parameters, longitudinal, fort.26

hzepsn (**args*, ***kws*)
Returns z normalized emittance history (default unit is [m-rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

z normalized emittance history

Notes

keywords: longitudinal, fort.26

hzepsnp (**args*, ***kws*)
Returns *n* % z normalized emittance history (default: *n* = 99.9, unit is [m-rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

n % z normalized emittance history

Notes

This value is calculated in case of *Flagdiag* equals to 2 or 4.

The percentage *n* [%] is defined by *Outputemit*.

keywords: longitudinal, fort.26

hzepsnf (**args*, ***kws*)
Returns full z normalized emittance history (default unit is [m-rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

full z normalized emittance history

Notes

This value is calculated in case of *Flagdiag* equals to 2 or 4.

keywords: longitudinal, fort.26

hxmax (**args*, ***kws*)
Returns maximum x position history (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

maximum x position history

Notes

keywords: horizontal, fort.27

hxpmax (**args*, ***kws*)
Returns maximum x' (xp, px) momentum history (default unit is [rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

maximum x' momentum history

Notes

keywords: horizontal, fort.27

hymax (**args*, ***kws*)
Returns maximum y position history (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

maximum y position history

Notes

keywords: vertical, fort.27

hypmax (**args*, ***kws*)
Returns maximum y' (yp, py) momentum history (default unit is [rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

maximum y' momentum history

Notes

keywords: vertical, fort.27

hzmax (**args*, ***kws*)
Returns maximum z position (phase) history (default unit is [deg]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

maximum z position history

Notes

`hphimax` is alias of `hzmax` (`hphimax` is named in fortran code).

keywords: longitudinal, fort.27

hphimax (**args*, ***kws*)
Returns maximum z position (phase) history (default unit is [deg]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

maximum z position history

Notes

`hphimax` is alias of `hzmax` (`hphimax` is named in fortran code).

keywords: longitudinal, fort.27

hzpmax (**args*, ***kws*)
Returns maximum z' (zp, pz) history (default unit is [MeV]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

maximum z' history

Notes

`hdemax` is alias of `hzpmax` (`hdemax` is named in fortran code).

keywords: longitudinal, energy deviation, fort.27

hdemax (**args*, ***kws*)
Returns maximum z' (zp, pz) history (default unit is [MeV]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

maximum z' history

Notes

`hdemax` is alias of `hzpmax` (`hdemax` is named in fortran code).

keywords: longitudinal, energy deviation, fort.27

hlpmin (**args*, ***kws*)
Returns minimum local particle number history (default unit is [1]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

minimum local particle number history

Notes

keywords: fort.28

hlpmax (**args*, ***kws*)
Returns maximum local particle number history (default unit is [1]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

maximum local particle number history

Notes

keywords: fort.28

hptot (**args*, ***kws*)
Returns total particle number history (default unit is [1]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

total particle number history

Notes

keywords: fort.28

hx3rd(**args*, ***kws*)
Returns x 3rd root of 3rd moment history (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

x 3rd moment history

Notes

This value is calculated in case of *Flagdiag* equals to 1 or 3.

keywords: horizontal, fort.29

hxp3rd(**args*, ***kws*)
Returns x' (xp, px) 3rd root of 3rd moment history (default unit is [rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

x' 3rd moment history

Notes

This value is calculated in case of *Flagdiag* equals to 1 or 3.

keywords: horizontal, fort.29

hy3rd(**args*, ***kws*)
Returns y 3rd root of 3rd moment history (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

y 3rd moment history

Notes

This value is calculated in case of *Flagdiag* equals to 1 or 3.

keywords: vertical, fort.29

hyp3rd(**args*, ***kws*)
Returns y' (yp, py) 3rd root of 3rd moment history (default unit is [rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

y' 3rd moment history

Notes

This value is calculated in case of *Flagdiag* equals to 1 or 3.

keywords: vertical, fort.29

hz3rd(**args*, ***kws*)
Returns z (phase) 3rd root of 3rd moment history (default unit is [deg]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

z 3rd moment history

Notes

This value is calculated in case of *Flagdiag* equals to 1 or 3.

keywords: longitudinal, fort.29

hzp3rd(**args*, ***kws*)
Returns z' (zp, pz) 3rd root of 3rd moment history (default unit is [MeV]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

z' 3rd moment history

Notes

This value is calculated in case of *Flagdiag* equals to 1 or 3.

keywords: longitudinal, energy deviation, fort.29

hx4th (**args*, ***kws*)
Returns x 4th root of 4th moment history (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

x 4th moment history

Notes

This value is calculated in case of *Flagdiag* equals to 1 or 3.

keywords: horizontal, fort.30

hxp4th (**args*, ***kws*)
Returns x' (xp, px) 4th root of 4th moment history (default unit is [rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

x' 4th moment history

Notes

This value is calculated in case of *Flagdiag* equals to 1 or 3.

keywords: horizontal, fort.30

hy4th (**args*, ***kws*)
Returns y 4th root of 4th moment history (default unit is [m]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

y 4th moment history

Notes

This value is calculated in case of *Flagdiag* equals to 1 or 3.

keywords: vertical, fort.30

hyp4th (**args*, ***kws*)
Returns y' (yp, py) 4th root of 4th moment history (default unit is [rad]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

y' 4th moment history

Notes

This value is calculated in case of *Flagdiag* equals to 1 or 3.

keywords: vertical, fort.30

hz4th (**args*, ***kws*)
Returns z (phase) 4th root of 4th moment history (default unit is [deg]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

z 4th moment history

Notes

This value is calculated in case of *Flagdiag* equals to 1 or 3.

keywords: longitudinal, fort.30

hzp4th (**args*, ***kws*)
Returns z' (zp, pz) 4th root of 4th moment history (default unit is [MeV]).

Parameters ***args :**

The same arguments are available with *hrefz*.

****kws :**

The same keywords are available with *hrefz*.

Returns ndarray

z' 4th moment history

Notes

This value is calculated in case of `Flagdiag` equals to 1 or 3.

keywords: longitudinal, energy deviation, fort.30

hrrms (**args*, ***kws*)
Returns r rms history (default unit is [m]).

Parameters ***args :**

The same arguments are available with `hrefz`.

****kws :**

The same keywords are available with `hrefz`.

Returns ndarray

r rms size history

Notes

This value is calculated in case of `Flagdiag` equals to 2 or 4.

keywords: root mean square, envelope, radius, fort.29

hrnpr (**args*, ***kws*)
Returns *n* % r size history (default: *n* = 99.9) (default unit is [m]).

Parameters ***args :**

The same arguments are available with `hrefz`.

****kws :**

The same keywords are available with `hrefz`.

Returns ndarray

n % r size history

Notes

This value is calculated in case of `Flagdiag` equals to 2 or 4.

The percentage *n* [%] is defined by `Outputemit`.

keywords: radius, fort.29

hrmax (**args*, ***kws*)
Returns maximum r distance history (default unit is [m]).

Parameters ***args :**

The same arguments are available with `hrefz`.

****kws :**

The same keywords are available with `hrefz`.

Returns ndarray

maximum r distance history

Notes

This value is calculated in case of `Flagdiag` equals to 2 or 4.

keywords: radius, fort.29

getx (**args*, ***kws*)
Returns x positions of the particles (default unit is [m]).

Parameters ***args :**

str – Tag name or physical unit.

int – ID number in the flag element. Default is 0 (= current distribution).

****kws :**

tag : str

Tag name of the target position.

unit : str

Physical unit name of the return value.

pid : int

ID number in the flag element.

Returns ndarray

x positions of the particles

Notes

keywords: horizontal, distribution

getxp (**args*, ***kws*)
Returns x' (xp, px) momentums of the particles (default unit is [rad]).

Parameters ***args :**

The same arguments are available with `getx`.

****kws :**

The same keywords are available with `getx`.

Returns ndarray

x' momentums of the particles

Notes

keywords: horizontal, distribution

gety (**args*, ***kws*)
Returns y positions of the particles (default unit is [m]).

Parameters ***args :**

The same arguments are available with `getx`.

****kws :**

The same keywords are available with *getx*.

Returns ndarray

y positions of the particles

Notes

keywords: vertical, distribution

getyp (**args*, ***kws*)
Returns y' (yp, py) momentums of the particles (default unit is [rad]).

Parameters ***args :**

The same arguments are available with *getx*.

****kws :**

The same keywords are available with *getx*.

Returns ndarray

y' momentums of the particles

Notes

keywords: vertical, distribution

getz (**args*, ***kws*)
Returns z positions (phase) of the particles (default unit is [deg]).

Parameters ***args :**

The same arguments are available with *getx*.

****kws :**

The same keywords are available with *getx*.

Returns ndarray

z positions of the particles

Notes

keywords: longitudinal, distribution

getzp (**args*, ***kws*)
Returns z' (zp, pz) of the particles (default unit is [MeV]).

Parameters ***args :**

The same arguments are available with *getx*.

****kws :**

The same keywords are available with *getx*.

Returns ndarray

z' momentums of the particles

Notes

keywords: energy deviation, longitudinal, distribution

getctm(**args*, ***kws*)
Returns charge to mass ratio of the particles (default unit is [c^2/eV]).

Parameters ***args :**

The same arguments are available with *getx*.

****kws :**

The same keywords are available with *getx*.

Returns ndarray

charge to mass ratio of the particles

getcpmp(**args*, ***kws*)
Returns charge per macro particle of the particles (default unit is [1]).

Parameters ***args :**

The same arguments are available with *getx*.

****kws :**

The same keywords are available with *getx*.

Returns ndarray

charge per macro-particle of the particles

getid(**args*, ***kws*)
Returns ID number of the particles (default unit is [1]).

Parameters ***args :**

The same arguments are available with *getx*.

****kws :**

The same keywords are available with *getx*.

Returns ndarray

ID number of the particles

getall(**args*, ***kws*)
Returns full information of the particles).

Parameters ***args :**

The same arguments are available with *getx*.

****kws :**

The same keywords are available with *getx*.

Returns (9, n) shape ndarray

All particle information. *n* is the number of particles.

- row 1 : x positions [m]
- row 2 : xp momentums [rad]
- row 3 : y positions [m]

- row 4 : yp momentums [rad]
- row 5 : z positions [deg]
- row 6 : zp momentums [MeV]
- row 7 : charge to mass ratios [c^2/eV]
- row 8 : charge per macro-particle weights [1]
- row 9 : ID numbers [1]

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

i

impact.control, 39
impact.input, 43
impact.result, 51

INDEX

A

autobcast (impact.result.Result attribute), 54

C

Chargestate (impact.input.Input attribute), 48
checker() (impact.control.Sequence method), 43
conf() (impact.control.Sequence method), 42
construct() (impact.control.Sequence method), 41
Ctmlist (impact.input.Input attribute), 47
Curlist (impact.input.Input attribute), 47

D

distribute() (impact.control.Sequence method), 40
Disttype (impact.input.Input attribute), 48
Distxlambda (impact.input.Input attribute), 48
Distxmismatch (impact.input.Input attribute), 49
Distxmu (impact.input.Input attribute), 48
Distxoffset (impact.input.Input attribute), 49
Distxoffset_m (impact.input.Input attribute), 50
Distxpmismatch (impact.input.Input attribute), 49
Distxpoffset (impact.input.Input attribute), 49
Distxpoffset_rad (impact.input.Input attribute), 51
Distxquadratic (impact.input.Input attribute), 49
Distxsigma (impact.input.Input attribute), 48
Distxtwiss (impact.input.Input attribute), 49
Distylambda (impact.input.Input attribute), 48
Distymismatch (impact.input.Input attribute), 49
Distymu (impact.input.Input attribute), 49
Distyoffset (impact.input.Input attribute), 49
Distyoffset_m (impact.input.Input attribute), 51
Distypmismatch (impact.input.Input attribute), 49
Distypoffset (impact.input.Input attribute), 49
Distypoffset_rad (impact.input.Input attribute), 51
Distyquadratic (impact.input.Input attribute), 50
Distysigma (impact.input.Input attribute), 48
Distytwiss (impact.input.Input attribute), 50
Distzlambda (impact.input.Input attribute), 49
Distzmismatch (impact.input.Input attribute), 49
Distzmu (impact.input.Input attribute), 49
Distzoffset (impact.input.Input attribute), 49
Distzoffset_deg (impact.input.Input attribute), 51
Distzpmismatch (impact.input.Input attribute), 49
Distzpoffset (impact.input.Input attribute), 49
Distzpoffset_eV (impact.input.Input attribute), 51
Distzquadratic (impact.input.Input attribute), 50
Distzsigma (impact.input.Input attribute), 49
Distztwiss (impact.input.Input attribute), 50
dtag (impact.result.Result attribute), 54

E

Energy (impact.input.Input attribute), 48

F

Flagbc (impact.input.Input attribute), 47
Flagdiag (impact.input.Input attribute), 46
Flagerror (impact.input.Input attribute), 46
Flaginteg (impact.input.Input attribute), 46
Flagrestart (impact.input.Input attribute), 47
Flagsc (impact.input.Input attribute), 48
Flagsubstep (impact.input.Input attribute), 47
Frequency (impact.input.Input attribute), 48

G

getall() (impact.result.Result method), 76
getcpmp() (impact.result.Result method), 76
getctm() (impact.result.Result method), 76
getid() (impact.result.Result method), 76
getx() (impact.result.Result method), 74
getxp() (impact.result.Result method), 74
gety() (impact.result.Result method), 74
getyp() (impact.result.Result method), 75
getz() (impact.result.Result method), 75
getzp() (impact.result.Result method), 75

H

hdemax() (impact.result.Result method), 67
hlpmax() (impact.result.Result method), 68
hlpmin() (impact.result.Result method), 68
hphimax() (impact.result.Result method), 67
hptot() (impact.result.Result method), 68
hrefbeta() (impact.result.Result method), 55
hrefeng() (impact.result.Result method), 55
hrefgma() (impact.result.Result method), 55
hrefphi() (impact.result.Result method), 54

hrefr() (impact.result.Result method), 56
hrefz() (impact.result.Result method), 54
hrmax() (impact.result.Result method), 73
hrnpr() (impact.result.Result method), 73
hrrms() (impact.result.Result method), 73
hx3rd() (impact.result.Result method), 69
hx4th() (impact.result.Result method), 71
hxcen() (impact.result.Result method), 56
hxepsn() (impact.result.Result method), 58
hxepsnf() (impact.result.Result method), 59
hxepsnp() (impact.result.Result method), 58
hxmax() (impact.result.Result method), 65
hxp3rd() (impact.result.Result method), 69
hxp4th() (impact.result.Result method), 71
hxpcen() (impact.result.Result method), 57
hxpmax() (impact.result.Result method), 65
hxprms() (impact.result.Result method), 57
hxrms() (impact.result.Result method), 56
hxtwsa() (impact.result.Result method), 57
hxtwsb() (impact.result.Result method), 58
hy3rd() (impact.result.Result method), 69
hy4th() (impact.result.Result method), 71
hycen() (impact.result.Result method), 59
hyepsn() (impact.result.Result method), 61
hyepsnf() (impact.result.Result method), 62
hyepsnp() (impact.result.Result method), 61
hymax() (impact.result.Result method), 66
hyp3rd() (impact.result.Result method), 70
hyp4th() (impact.result.Result method), 72
hypcen() (impact.result.Result method), 60
hypmax() (impact.result.Result method), 66
hyprms() (impact.result.Result method), 60
hyrms() (impact.result.Result method), 59
hytwsa() (impact.result.Result method), 60
hytwsb() (impact.result.Result method), 61
hz3rd() (impact.result.Result method), 70
hz4th() (impact.result.Result method), 72
hzcen() (impact.result.Result method), 62
hzepsn() (impact.result.Result method), 64
hzepsnf() (impact.result.Result method), 65
hzepsnp() (impact.result.Result method), 64
hzmax() (impact.result.Result method), 66
hzp3rd() (impact.result.Result method), 70
hzp4th() (impact.result.Result method), 72
hzpcen() (impact.result.Result method), 63
hzpmax() (impact.result.Result method), 67
hzprms() (impact.result.Result method), 63
hzrms() (impact.result.Result method), 62
hztwsa() (impact.result.Result method), 63
hztwsb() (impact.result.Result method), 64

I

impact.control (module), 39
impact.input (module), 43
impact.result (module), 51
Input (class in impact.input), 43
insert() (impact.control.Sequence method), 41

L

load() (impact.control.Sequence method), 40

M

Mass (impact.input.Input attribute), 48
Mpisize (impact.input.Input attribute), 46

N

Ncharge (impact.input.Input attribute), 47
Ndim (impact.input.Input attribute), 46
Nplist (impact.input.Input attribute), 47
Nptot (impact.input.Input attribute), 46
Nxgrid (impact.input.Input attribute), 46
Nygrid (impact.input.Input attribute), 46
Nzgrid (impact.input.Input attribute), 47

O

output() (impact.control.Sequence method), 43
Outputemit (impact.input.Input attribute), 48

P

Phaseini (impact.input.Input attribute), 48

Q

qmlabel (impact.result.Result attribute), 54

R

read() (impact.control.Sequence method), 39
Result (class in impact.result), 51
run() (impact.control.Sequence method), 41

S

save() (impact.control.Sequence method), 43
search() (impact.control.Sequence method), 43
seed (impact.control.Sequence attribute), 39
Sequence (class in impact.control), 39
subdir (impact.control.Sequence attribute), 39

X

Xbound (impact.input.Input attribute), 47

Y

Ybound (impact.input.Input attribute), 47

Z

Zperiod (impact.input.Input attribute), 47