**April**

2011

**User manual for Tracy3, SOLEIL Version**

Jianfeng ZHANG, Laurent NADOLSKI

Draft of Tracy3 manual

**Synchrotron SOLEIL, France**

08

**Automne**

**Table of contents**

1 Introduction 4

1.1 File path 4

1.2 File names 5

1.2.1 Lattice file name 5

1.2.2 Multipole file name 5

1.2.3 Correctors and skew quadrupole file names with multipole errors in Soleil lattice 5

1.2.4 Cutoff value 6

1.3 Commands 6

1.3.1 Active quadrupole fringe field 6

1.3.2 Set voltage of RF cavity 7

1.3.3 Print twiss parameters to a user defined file 7

1.3.4 Print close orbit to a user defined file 7

1.3.5 Read vacuum chamber setting from a file 8

1.3.6 Read lattice element field error from a file 8

1.3.7 Read lattice element misalignment error from a file 8

1.3.8 Set parameters for orbit correction 9

1.3.9 Get tunes by tracking 9

1.3.10 Get chromaticities by tracking 9

1.3.11 Tune shift with amplitude 9

1.3.12 Tune shift with energy 10

1.3.13 Frequency map analysis for on momentum particle 10

1.3.14 Frequency map analysis for off momentum particle 11

1.3.15 Add random rotation coupling error to the full quadrupoles 12

1.3.16 Add random rotation coupling error to the half quadrupoles 12

1.3.17 Calculate natural emittance in the lattice 12

1.3.18 Momentum acceptance 12

1.3.19 Read multipole error from a file 13

1.3.20 Fit tune for the lattice with full quadrupole 13

1.3.21 Fit tune for the lattice with half quadrupoles 13

1.3.22 Fit chromaticity 14

1.3.23 Touschek lifetime determined by RF acceptance 14

1.3.24 Intra Beam Scattering(IBS) 14

1.3.25 Touschek lifetime determined by RF acceptance and momentum accpetance 14

1.3.26 Phase space 14

1.4 Lattice file 15

1.4.1 Syntax 15

1.4.2 variables 15

1.4.3 Start line 15

1.4.4 Global variables 16

1.4.5 Drift space 16

1.4.6 dipole 16

1.4.7 quadrupole 17

1.4.8 Skew quadrupole 18

1.4.9 sextupole 18

1.4.10 multipole 18

1.4.11 wiggler 19

1.4.12 field map 20

1.4.13 Insertion device 20

1.4.14 RF cavity 20

1.4.15 corrector 21

1.4.16 marker 21

1.4.17 BPM 22

1.4.18 Girder 22

1.4.19 Element block 22

1.4.20 Cell 23

1.4.21 ring 23

1.4.22 End line 23

1.5 Multipole error file 23

1.5.1 Define systematic multipole errors 23

1.5.2 Define ramdom multipole error 25

1.6 Misalignment error file 25

1.6.1 Define systematic misalignment errors 26

1.6.2 Define random misalignment errors 26

1.7 Vacuum chamber file 27

# Introduction

Soleil version of Tracy 3 is an executable code with more flexibility and easier to use. User does not need to know the structure and how to compile the code, what they need to do is to write an input script, and then run the code. Depending on the need, user can also write the files to define multipole errors and vacuum chamber, and provide the file name in the user input script, then set the multipole error of the lattice elements and the vacuum limit for the different region of the lattice.

Execute the code

To execute the tracy code, the user need to provide an input script with the file extension ‘.prm’. For example, ‘Input\_test.prm’ is a user defined script, then user can type the command

**tracy3**  Input\_test.prm

in the command line, and then press ‘return’ key to execute the code. The input script can be defined with any valid file name but must be ended with the file extension “.prm”.

For soleil lattice, the command is: **soltracy3** Input\_test.prm

User input script

There are two parts in the user input script. The first part is to set the file path and the filenames at the beginning of the input script; the second part is to define Boolean commands with or without parameters to do different calculations. The blank lines and lines starting with "#" (comment line) are ignored by the code.

* Files and other keywords are executed not according to the defined sequence in user input script.
* The command with the last 4 characters as “Flags” are executed according to the sequence in the code.

## File path

In the user input script, user can specify the name of the files which are used for the further calculation, such as the lattice file, multipole error file, vacuum chamber file; when specifying the filename, user also need to provide the absolute path for the file which is not located at the current path, such as: “/home/physmach/Tracy3/tracy/soleil.lat”. For convenience, user can put the files specified in the input script at a certain directory, and then specific this directory using the command:

**in\_dir** user\_defined\_path

For example:

**in\_dir** /home/zhang/codes/TracyIII/lattice/

This command tells the code all the files specified in the script are located at the directory “/home/zhang/codes/TracyIII/lattice/”. If user does not set the directory through command in\_dir and files specified in the script are not found at the current working directory, the code will give an error message and stops running.

## File names

### Lattice file name

In the input script, user must define the lattice name, this is mandatory. The command is:

**lat\_file** lattice\_file\_name

Here ‘lattice\_file\_name’ is the lattice file name without “.lat” extension. For example,

the following command sets the lattice file of Soleil ring:

**lat\_file** solamor2\_reglage\_focalisation\_chcvqt\_thicksextu\_LQPintermediaire\_QFF

If in the lattice, there are skew quadrupoles, or correctors which are used for orbit correction, or beam position monitors, or girders, user MUST specify the element names of these elements using the following command in the “.prm” file:

**h\_corr** HCM

**v\_corr** VCM

**gs** GS

**ge** GE

**bpm\_name** BPM

**qt** QT

Here ‘HCM’ is the name of horizontal correctors which are used for horizontal orbit correction; ‘VCM’ is the name of horizontal correctors which are used for vertical orbit correction; ‘GS’ is the name of the start of the girder; ‘GE’ is the name of the end of the girder; ‘BPM’ is the name of beam position monitors defined in the lattice; ‘QT’ is the name of the skew quadrupoles defined in the lattice.

### Multipole file name

User can also read the multipole errors from an external file, the file is specified in the user script using the following command:

**multipole\_file** multipole\_file

Here ‘multipole\_file’ is the user defined multipole file, the format of this file is given at section 1.5.

### Correctors and skew quadrupole file names with multipole errors in Soleil lattice

User can read the multipole errors of horizontal, vertical correctors and skew quadrupoles from the files. In the script, user can specify the file names, for example:

**fic\_hcorr** corh.txt

**fic\_vcorr** corv.txt

**fic\_skew** corqt.txt

Here ‘corh.txt’, ‘corv.txt’, ‘corqt.txt’ are the multipole error files of horizontal, vertical correctors and skew quadrupoles, respectively. Currently these files are machine based, they only work for Soleil lattice.

### Cutoff value

**normalcut** n

Set the cutoff value of all random (Gaussian distribution) to n sigma

## Commands

The following commands turn on the boolean flags in the code to set the machine parameters and to do different calculations. All these commands are optional, user can choose whichever they need. If users want to use the flag, they can write the flag in the script, if they do not want to use it, they can delete or comment out the flag. The Boolean flags in the user input script have the following features:

* If the flags are not active, then the default values for all the boolean commands are false.
* The code will run the command according to the defined sequence in the “.prm” script.
* For the commands without parameters, such as:  “TuneTracFlag” etc, user can use them as often as they want.
* The code will execute the command according to the sequence in the file. For example,

FitTune4Flag     qp7a qp7b  qp9a qp9b  18.18 10.28

ReadMultipoleFlag

FitTune4Flag     qp7a qp7b  qp9a qp9b  18.202 10.317

Then the code will fit tunes to the taraget tunes (18.18 10.28), and then read multipole errors into the lattice, then fit the tunes of the lattice with new settings to (18.202 10.317).

* For the commands with parameters, such as “FitTune4Flag     qp7a qp7b  qp9a qp9b  18.202 10.317”, the code will always use the last set of parameters, although the user defines the different setting of parameters. So the best way to use such command is to use many times in the script but always with the same parameters. This feature will be improved in the future.
* User can define maximum 500 commands in one input script.

### Active quadrupole fringe field

To activate quadrupole fringefield, use the command:

**QuadFringeOnFlag**

With this command, user can define the fringe field at the entrance or exit of the quadrupole with the command FF1=1 or FF2=2 in the lattice file when defining the quadrupole. This flag is a global flag, if user set this flag in the input script, it will always have effect until this flag is reset.

### Set voltage of RF cavity

User can reset the RF voltage by setting 'RFvoltageFlag' in the user input script, in order to replace the value of RF voltage which is defined in the lattice. For example:

**RFvoltageFlag** 3000000

Here ‘RFvoltageFlag’ is the name of the key word, ‘3000000’ is the value of RF voltage with the unit [volt].

### Print the tracking coordinates at each element to a file

To print the coordinates around COD at each element which are obtained by tracking, use the command as the following example:

**PrintTrackFlag**  track\_file x px y py delta ctau nturn

For example:

**PrintTrackFlag**  track.out 0.001 0.0 0.0 0.0 0.0 0.0 50

The parameters and the default value is shown in ……………………….

If the user ………………..

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Default value | Unit |
| track\_file | File to save the tracked coordinates around each lattice element | track.out |  |
| x | start horizontal coordinate | 0.001 | [m] |
| px | start horizontal derivative coordinate | 0.0 | [rad] |
| y | start vertical coordinate | 0.0 | [m] |
| py | start vertical derivative coordinate | 0.0 | [rad] |
| delta | start relative energy offset | 0.0 |  |
| ctau | start longitudinal coordinate | 0.0 | [m] |
| nturn | number of turn to track | 50 |  |

### Print twiss parameters to a user defined file

With the command “PrintTwissFlag”, the code will print the twiss parameters to a user defined file. The format is:

**PrintTwissFlag**  user\_defined\_file

If user use the command **PrintTwissFlag** but without define the file name, then the code will print the twiss to the default file “twiss.out”.

In Tracy 3, twiss parameters are automatically printed to the following files:

* “linlat.out” after reading the lattice into the code.

            “linlat\_coupling.out”               after calculating the coupling factor of the lattice

            “linlat\_errcoupling.out”           after add the coupling error by rotating the full quadrupoles in the lattice

               “linlat\_errcoupling2.out”           after add the coupling error by rotating half quadrupoles in the lattice

### Print close orbit to a user defined file

With the command “PrintTwissFlag”, the code will print the close orbit to a user defined file. The format is:

**PrintCODFlag**  user\_defined\_file

If user use the command **PrintCODFlag** but without defining the file name, then the closed orbit will be printed to the default file “printcod.out”.

In Tracy 3, close orbit file “cod.out” is automatically generated after reading the lattice.

In the close orbit file, the data is save in the following format:

# i name s code betax nux betay nuy xcod ycod dSx dSy dipx dipy.

“#” denotes this is command line, and the meaning of other parameters are shown in

Table 1

|  |  |  |
| --- | --- | --- |
| Name | Description | Unit |
| I | Number of the element |  |
| Name | Element name defined in lattice |  |
| S | Longitudinal length | Meter |
| Code | Symbol for the element;  0.5 dipole  -1.0 defocusing quadrupole  1.0 focusing quadrupole  -1.5 defocusing sextupole  1.5 focusing sextupole  0 other element |  |
| Betax | Horizontal Beta function | meter |
| Nux | Horizontal tune |  |
| Betay | Vertical Beta function | Meter |
| Nuy | Vertical tune |  |
| Xcod | Horizontal close orbit | Millimeter |
| Ycod | vertical close orbit | Millimeter |
| dSx | Horizontal displacement of the element | Millimeter |
| dSy | vertical displacement of the element | Millimeter |
| Dipx | Horizontal dipole strength of the element | Milli radian |
| dipy | vertical dipole strength of the element | Milli radian |

# [m] [m] [m] [mm] [mm] [mm] [mm] [mrad] [mrad]

#

### Read vacuum chamber setting from a file

To read the vacuum chamber from the user defined chamber file, use the Command :

**ReadChamberFlag** Chamber\_example.dat

In the file ‘Chamber\_example.dat’, user can specify the vacuum limit at the different region of the lattice, the code will take into account the vacuum limit if the Boolean flag ReadChamberFlag is defined in the input script. The format of the chamber file is given at section 1.7.

This command is a generic command, and works for all the machines.

### Read lattice element field error from a file

To read the field error of the lattice elements from the user defined file, use the Command :

**ReadfefileFlag dip.fe**

The code will read the systematic and random field errors of the lattice elements from the file ‘dip.fe’, then the field error with replace the corresponding field components of the elements. In the file ‘dip.fe’, user can specify the systematic and random field error of the lattice elements. The format of the field error file is given at section 1.5.

This command is a generic command, and works for all the machines.

### Read lattice element misalignment error from a file

To read the misalignment error of the lattice elements from the user defined file, use the Command :

**ReadaefileFlag dip.ae**

The code will read the systematic and random misalignment errors of the lattice elements from the file ‘dip.fe’, then the misalignment error with replace the corresponding components of the elements; finally the code will do orbit correction, so user need to define the relative parameters when doing orbit correction.

In the file ‘dip.ae’, user can specify the systematic and random misalignment error of the lattice elements. The format of the misalignment error file is given at section 1.6.

This command is a generic command, and works for all the machines.

### Set parameters for orbit correction

In order to do orbit correction, use need to define the following parameters:

**hcorr\_file** hcorr\_56nom.state

This is to specify the file which correctors are used for H orbit correction.

**vcorr\_file** vcorr\_56nom.state

this is to specify the file which correctors are used for V orbit correction

**n\_stat** 2

This defines number of statistics (seeds) --> N flat-files

**n\_scale** 1

this defines number of times to scale the errors

**n\_orbit** 0

This defines number of iterations for orbit correction, if 0, no orbit correction

**nwh** 60

This defines number of singular values in H-plane, must be not larger than the number of correctors used for orbit correction

**nwv** 60

This defines number of singular values in V-plane, must be not larger than the number of correctors used for orbit correction.

if ReadaefieldFlag is on, then must set these parameters to do orbit correction. But If the user set ‘n\_orbit’ larger than 0, then the code will just read the misalignment error, and no orbit correction.

### Get tunes by tracking

To get tunes obtained by tracking, use command:

**TuneTracFlag**

### Get chromaticities by tracking

To get chromaticity obtained by tracking, use command:

**ChromTracFlag**

### Tune shift with amplitude

To calculate tune shift with amplitude, one need to use the following command:

**AmplitudeTuneShiftFlag** nudx\_filenudz\_filenxpoint nypoint nturn xmax ymax delta

For example:

**AmplitudeTuneShiftFlag** nudx.out nudz.out 50 30 516 0.035 0.02 0.0

The meaning of parameters and default values of command “AmplitudeTuneShiftFlag” are shown in Table 2. If user only use command **AmplitudeTuneShiftFlag** without parameters, then the code will use all the default values.

Table 2 Parameters of the command to calculate tune shift with amplitude

|  |  |  |
| --- | --- | --- |
| **parameter** | **meaning** | **default value** |
| nudx\_file | file to save the calculated tune shift with horizontal amplitude | nudx.out |
| nudz\_file | file to save the calculated tune shift with vertical amplitude | nudz.out |
| nxpoint | number of points in horizontal direction | 31 |
| nypoint | number of points in vertical direction | 21 |
| nturn | number of turns to track tune | 516 |
| xmax | maximum amplitude of x with the unit [m] | 0.025 |
| ymax | maximum amplitude of y with the unit [m] | 0.005 |
| delta | energy offset of the particle | 0.0 |

### Tune shift with energy

To calculate tune shift with energy, one need to use the command:

**EnergyTuneShiftFlag**  file npoint nturn deltamax

For example:

**EnergyTuneShiftFlag** nudptest.out 31 1026 0.06

The meaning of parameters and default values of this command are shown in Table 3.

If user only use command **EnergyTuneShiftFlag** without parameters, then the code will use all the default values.

Table 3 Parameters of the command to calculate tune shift with energy

|  |  |  |
| --- | --- | --- |
| **parameter** | **meaning** | **default value** |
| file | file to save the calculated tune shift with energy | nudp.out |
| npoint | number of points | 31 |
| nturn | number of turns for tracking | 516 |
| deltamax | maximum energy offset of the particle | 0.06 |

### Frequency map analysis for on momentum particle

To do frequency map for the on momentum particle, use the command:

**FmapFlag** fmap\_file nxpoint nypoint nturn xmax ymax delta diffusion

For example:

**FmapFlag** fmap\_file 31 21 516 0.025 0.005 0.0 true

The meaning of parameters and default values are shown in Table 4. If user only use the command **FmapFlag** but without all the parameters, then the code use the default values.

Table 4 Parameters of the command to do frequency map analysis for on momentum particle.

|  |  |  |  |
| --- | --- | --- | --- |
| **parameter** | **meaning** | | **default value** |
| fmap\_file | File to save the calculated frequency map analysis | | fmap.out |
| nxpoint | number of points in the horizontal direction | | 31 |
| nypoint | number of points in the vertical direction | | 21 |
| nturn | number of turns for tracking | | 516 |
| xmax | maximum amplitude in the horizontal direction with the unit [m] | | 0.025 |
| ymax | | maximum amplitude in the vertical direction with the unit [m] | 0.005 |
| delta | | energy offset of the particle | 0.0 |
| diffusion | | boolean flag to compute tune diffusion | true |

### Frequency map analysis for off momentum particle

To do frequency map for off momentum particle, use the command:

**FmapdpFlag** fmapdp\_file nxpoint nepoint nturn xmax emax y diffusion

The meaning of parameters and default values are shown in Table 5. If user only use the command **FmapdpFlag** but without defining all the parameters, then the code use the default values.

Table 5 Parameters of the command to do frequency map analysis for off momentum particle

|  |  |  |
| --- | --- | --- |
| **parameters** | **meaning** | **default value** |
| fmapdp\_file | File to save the calculated frequency map analysis | fmapdp.out |
| nxpoint | number of points in the horizontal direction | 31 |
| nepoint | number of points for the energy | 21 |
| nturn | number of turns for tracking | 516 |
| xmax | maximum amplitude in the horizontal direction with the unit [m] | 0.025 |
| emax | maximum energy offset of the particle | 0.005 |
| y | amplitude in the vertical direction with the unit [m] | 0.0 |
| diffusion | boolean flag to compute tune diffusion | true |

### Add random rotation coupling error to the full quadrupoles

To add random rotation error to the full quadrupole in the lattice, user can use the command as the following example:

**ErrorCouplingFlag** 0 0.0007

In this example, '0' is the random seed number; '0.0007' is the RMS value of the rotation angle of the quadrupole with the unit [rad].

After setting the rotation error in the lattice, the code will generate a file at the current working directory, and the file name is “flat\_file\_errcoupling.dat”, user can check the error setting in this file.

### Add random rotation coupling error to the half quadrupoles

To add random rotation coupling error to the two halfs quadrupoles in the lattice, user can use the command as the following example:

**ErrorCoupling2Flag** 0 0.0007

In this example, '0' is the random seed number; '0.0007' is the RMS value of the rotation angle of the quadrupole with the unit [rad].

After setting the errors in the lattice, the code will generate a file at the current working directory, and the file name is “flat\_file\_errcoupling.dat”, user can check the error setting in this file.

This command is delicated for soleil lattice in which each quadrupole is cut into two half quadrupoles.

### Calculate natural emittance in the lattice

To calculate natural emmttice of the lattice, use command:

**CouplingFlag**

The natural emittance and coupling factor of the lattice will be printed on the screen.

### Momentum acceptance

The following command calculate momentum acceptance:

**MomentumAccFlag** MomAccFile TrackDim istart istop deltaminp

Deltamaxp nstepp deltaminn deltamaxn nstepn turns zinitial

For example:

**MomentumAccFlag**  momacc.out 4D 1 209 0.01 0.05 100 -0.01 -0.05 100 1026 0.0001

The meaning of parameters and default values are shown in Table 6. If user only use **MomentumAccFlag**,then the code will use the default values.

Table 6 Parameters of the command to calculate momentum acceptance

|  |  |  |
| --- | --- | --- |
| **parameter** | **meaning** | **Default value** |
| MomAccFile | File to save the tracked momentum acceptance at each elements; saved in the current directory. | momentumacceptance.out |
| TrackDim | 4D/6D tracking to get the momentum acceptance | 6D |
| istart | start element in the lattice for the tracking | 1 |
| istop | last element in the lattice for tracking | 108 |
| nstepp | number of steps to do the tracking in the positive energy range | 100 |
| nstepn | number of steps to do the tracking in the negative energy range | 100 |
| Deltaminp | Positive start energy of the tracking | 0.01 |
| Deltamaxp | Positive end energy of the tracking | 0.05 |
| Deltaminn | negative start energy of the tracking | -0.01 |
| Deltamaxn | end negative energy of the tracking | -0.05 |
| nturns | number of turn | 1026 |
| zinitial | the initial vertical coordinate which is used to search for 6D closed orbit. This value should be a small value. | 0.0003 [m] |

### Read multipole error from a file

To read multipole error from a file and set them to the lattice, use the command:

**ReadMultipoleFlag**

Currently this command only works for Soleil lattice, and the errors of correctors and skew quadrupoles are added on the thick sextupoles which are associated with them. The format of multipole errors file is given in the file section 1.5.

After setting the multipole error in the lattice, the code will generate a file at the current working directory, and the file name is “flat\_file\_errmultipole.dat”, user can check this file to verify the error setting.

### Fit tune for the lattice with full quadrupole

To fit tunes, use the command as the following example:

**FitTuneFlag** q7 q9 18.202 10.317

In this example, 'q7' and 'q9' are the pair of quadrupoles used to fit the tunes, '18.202' and '10.317' are the targeted horizontal(nux) and vertical tunes(nuz).

After fitting the tunes, field strengths of the fitted quadrupoles before and after the fitting are printed on the screen, user can copy the new field strengths of quadrupoles to the lattice file for further analysis.

**FitTuneFlag** is a generic command; it works for the lattices with full qudrupoles.

If user does not specify the target nux and nuz, then the code use the default values: (0, 0).

### Fit tune for the lattice with half quadrupoles

To fit tunes for the lattice with half quadrupoles, use the command as the following example:

**FitTune4Flag** qp7a qp7b qp9a qp9b 18.202 10.317

In this example, all the variables have the same meaning as the ones in the command 'FitTuneFlag', except 'qp7a' and 'qp7b' are the two half of quadrupoles 'qp7', and 'qp9a' and ‘qp9b’ are the two half of 'qp9'.

After fitting the tunes, the field strengths of fitted quadrupole before and after the fitting are printed on the screen, user can copy the new field strengths of quadrupoles to the lattice file for further analysis.

**FitTune4Flag** is a command that works for the lattices in which each quadrupole are cut into two halves.

If user does not specify the target nux and nuz, then the code use the default values: (0, 0).

### Fit chromaticity

To fit chromaticity, use the command as the following example:

**FitChromFlag**  sx9 sx10 2.0 2.6

In this example, 'sx9' and 'sx10' are the pair of sextupoles used to fit the chromaticities, '2.0' and '2.6' are the targeted horizontal(ksix) and vertical chromaticities(ksiz). If user does not specify the target ksix and ksiz, then the code use the values: (0, 0).

After fitting the chromaticites, the field strengths of fitted sextupoles before and after the fitting are printed on the screen, user can copy the new field strengths of sextupoles to the lattice file for further analysis.

### Touschek lifetime determined by RF acceptance

To calculate Touschek lifetime based on the RF acceptance, use the following command:

**TouschekFlag**

### Intra Beam Scattering(IBS)

To calculate Intra Beam Scattering with command:

**IBSFlag**

### Touschek lifetime determined by RF acceptance and momentum accpetance

To track momentum acceptance and then based on the tracked momentum acceptance, then to get Touschek lifetime:

**TousTrackFlag**

In this case, the chamber file should be defined in the user script.

### Phase space

To calculate phase space, use the command:

**PhaseSpaceFlag** Phase\_phase\_file Phase\_Dim Phase\_X Phase\_Px

Phase\_Y Phase\_Py

Phase\_delta Phase\_ctau Phase\_nturn damping\_flag

Example:

**PhaseSpaceFlag** 6D 1e-6 0.0 1e-6 0.0 0.012 0.0 1000 false

The meaning of parameters and defaults values of **PhaseSpaceFlag** are shown in table Table 7. If user only use **PhaseSpaceFlag**,then the code will use the default values.

Table 7 Parameters of the command to get phase space

|  |  |  |
| --- | --- | --- |
| **parameter** | **meaning** | **default value** |
| Phase\_phase\_file | File to save tracked phase space; saved in the current directory. | phase.out |
| Phase\_Dim | 4D/6D tracking | 4D |
| Phase\_X | Horizontal coordinate at the start point of tracking | 0.0 |
| Phase\_Px | Horizontal derivative at the start point of tracking | 0.0 |
| Phase\_Y | vertical coordinate at the start point of tracking | 0.0 |
| Phase\_Py | vertical derivative at the start point of tracking | 0.0 |
| Phase\_delta | Energy at the start point of tracking | 0.0 |
| Phase\_ctau | Longitudinal position at the start point of tracking | 0.0 |
| Phase\_nturn | number of turns for tracking | 512 |
| Damping\_flag | Boolean flag to turn on/off the radiation damping during the tracking | false |

User Defined Files

## Lattice file

The following are the rules to define a lattice in the lattice file of Tracy 3.

### Syntax

* Every line embraced by “{}” is comment line, for example:

**{\*\*\*\*\*drift space \*\*\*\*\*}**

* Each sentence is ended by ‘;’ or no punctuation. Tracy is not sensitive to capital/small letters.
* User can define any lattice element with the name they want, but the element type is fixed, and the type of the element doesn’t depend on the capital or small letter.
* For the lattice of the ring, the definition of RF cavity is mandatory, and the harmonic number of the RF cavity is also mandatory; for the lattice of the linac, the definition of the RF cavity is optional.

### variables

User can define the variable in the lattice file, such as:

**Intmeth = 4;**

so in the lattice file, everywhere of ‘intmeth’ will be replaced by ‘4’.

### Start line

The lattice file must begin with the sentence:

**define lattice;**

This definition is mandatory.

### Global variables

After define the ring, user needs to define the system parameters of the lattice. **Energy** is the beam energy with unit [GeV], **dP** is the relative momentum offset of the particle, and **CODeps** is the convergence for the algorism to find the closed orbit.

For example:

**Energy =** 2.739;

**dP =** 1.0d-10;

**CODeps=** 1.0d-15;

These definitions are mandatory.

### Drift space

To define a drift element, the format is:

Symbol : **drift** , **L** = length ;

‘symbol’ is the user defined element name, ‘drift’ is a keyword to denote this element is a drift, and using keyword ‘L’, user can define the drift length of the element with unit [m]. For example:

SD1a **: drift, L**= 0.900000;

The length of the drift is mandatory.

### dipole

To define a bending magnet, the format is:

Symbol : **bending** , **L** = length, **T** = total bending angle, **T1** = entrance angle, **T2**=exit angle, **K**=quadrupole component field strength**, method** = integration\_method, **N**=Number of slice, **gap** = gap between two poles;

|  |  |  |
| --- | --- | --- |
| Symbol | Parameter | Units |
| L | length | m |
| T | total bending magnet | ° |
| T1 | Entrance angle | ° |
| T2 | Exit Angle | ° |
| K | gradient | m-1 |
| Method | Integration method | 1,2,4 |
| Gap | Fringe field |  |
| N | number of slices |  |

‘symbol’ is the user defined element name; ‘bending’ is a keyword to denote this element is a dipole; and using keyword ‘L’, user can define the length of the element with unit [m]; ‘T’ is the total bending angle of the magnet with unit [degree]; ‘T1’ is the entrance angle when the electron enters the dipole, the unit is [degree]; ‘T2’ is the exit angle when the electron exits the dipole, the unit is [degree]; ‘K’ is the field strength with unit [m-1] or integrated field strength(unitless) of the quadrupole components if this dipole is a combined magnet; keyword ‘method’ define the order of symplectic integrated method in the code, the value ‘1’ means 1st order, ‘2’ means 2nd order, and ‘4’ means 4th order; keyword ‘N’ means how many pieces of this element is cut when it is treated in the code, ‘gap’ is the distance between two poles of the dipole’with unit [m], the gap size determine the fringe field, if the gap size is 0, then the dipole has no fringe field.

For example:

**{\*\* bending \*\*}**

**beta\_gap=37e-3;**

**tracy\_gap=beta\_gap\*2\*0.724;**

**BEND1 : bending, L= 1.05243, T= 11.25, T1=5.5906, T2=5.67658, K=0.00204, N=4,**

**method=intmeth,gap=tracy\_gap;**

The parameters of ‘bending’ are optional, the default values for the missing parameters are 0, and **the default value for ‘method’ is 0.**

### quadrupole

To define a quadrupole element, the format is:

Symbol : **quadrupole** , **L** = length, **tilt** = tilt angle, **K** = field strength, **FF1** = 1[0], **FF2**=1[0],

**FFscaling** = scaling factor of the field, **method** = integration\_method, **N**=Number of slice;

‘symbol’ is the user defined element name; ‘quadrupole’ is a keyword to denote this element is a quadrupole or parts of a quadrupole; and using key word ‘L’, user can define the length of the element with unit [m]; ‘tilt’ is the tilt angle of the quadrupole with unit [degree], if ‘tilt’ is nonzero, then the quadrupole is a skew quadruple; ‘K’ is the field strength with unit [m-2] or integrated field strength with unit [m-1] of this quadrupole component depending on the length ‘L’ is 0 or not; the value of ‘FF1’ is either 1 or 0, 1 means taking into account the fringe field at the left edge of this component, 0 means not taking into account the fringe field at the left edge; the value of ‘FF2’ is also 1 or 0, 1 means taking into account the fringe field at the right edge, and 0 means not taking into account the fringe field at the right edge(user should notice, in order to take into account, user need also to set the flag “QuadFringeOnFlag” in the user input script); keyword ‘method’ define the order of symplectic integration method used in the code, the value ‘1’ means 1st order, ‘2’ means 2nd order, and ‘4’ means 4th order; keyword ‘N’ means how many pieces of this element is cut when it is treated in the code.

Example:

**{\*\* Quadrupole \*\*}**

**Nq=8/2; {Number of slices}**

**dgsurg=1.00;**

**dgsurgL=1.00;**

**quadfringe=1.0;**

**LQC=0.3602;**

**QP1a : quadrupole, L=LQC/2,K= -1.073038\*dgsurg, FF1=quadfringe, FF2=0, FFscaling =1,**

**method=intmeth, N=Nq;**

The parameters of ‘quadrupole’ are optional, the default value for ‘method’ is 4, the default value for ‘FFscaling’ is 1, the default value for the other parameters are 0.

### Skew quadrupole

The skew quadrupole is a special type of quadrupole, with a non-zero tilt angle. For example:

**QT: quadrupole, tilt=45.0, K= 0.0, method=intmeth, N=1;**

**Notice:**

For lattice with skew quadrupoles,

* User must specify the name of skew quadrupole in the input file “\*.prm” with the commands:

**qt** skewquad

Here ‘qt’ is the name of the skew quadrupoles defined in the lattice.

### sextupole

To define one sextupole, the format is:

Symbol : **sextupole** , **L** = length, **K** = field strength, **FF1** = 1[0], **FF2**=1[0],

**method** = integration\_method, **N**=Number of slice;

‘symbol’ is the user defined element name; ‘sextupole’ is a keyword to denote this element is a sextupole; and using keyword ‘L’, user can define the length of the element with unit [m]; ‘K’ is the field strength([m-2 ]) or integrated field strength([m-1 ]) of this sextupole components depending on the length ‘L’ is 0 or not, the value of ‘FF1’ is either 1 or 0, 1 means taking into account the fringe field at the left edge of this component, 0 means not taking into account the fringe field at the left edge; the value of ‘FF2’ is also 1 or 0, 1 means taking into account the fringe field at the right edge, and 0 means not taking into account the fringe field at the right edge; keyword ‘method’ define the order of symplectic integration method in the code, the value ‘1’ means 1st order, ‘2’ means 2nd order, and ‘4’ means 4th order; keyword ‘N’ means how many pieces of this element is cut when it is treated in the code. For example:

**NqSx=1; {Number of slices}**

**coef=1.0/0.16;**

**method4sextu = 4;**

**sextfringe = 0;**

**SX1 : sextupole, L=0.16, K = 1.719190\*coef, method=method4sextu, N = NqSx,**

**FF1=sextfringe, FF2=sextfringe;**

The parameters of ‘sextupole’ are optional, the default value for ‘method’ is 4, the default value for the other parameters is 0.

### multipole

To define a multipole, the format is:

symbol: **Multipole, L**=<length>, **T** =<bending angle>, **T1**=<entrance angle>,

**T2**=<exit angle>, **tilt**=<roll angle>, **N** =<# of kicks>,

**method**=<method> ,

**HOM=(**i, <Bi>, <Ai>, j, <Bj>, <Aj>,….n, <Bn>, <An>);

‘L’ is the element length with unit [m], ‘T’ is the total bending angle of the element with unit [degree], ‘T1’ is the entrance angle with unit [degree], ‘T2’ is the exit angle with unit [degree], ‘tilt’ is the roll angle of the element with unit [degree], ‘N’ is the number of slices of the element when it is treated in the code, ‘method’ is symplectic integration method, 1 is 1st order, 2 is 2nd order, 4 is 4th order; ‘HOM’ is the magnetic component of the element, ‘n’ means n-th components, ‘Bn’ is the n-th upright magnetic component with unit [m-(n/2-1)], and ‘An’ is the n-th skew magnetic component with unit [m-(n-1)].

**Example 1:**

B**: multipole, L**=0.70, **T**=10.0, **T1**=5.0, **T2**=5.0, **HOM** = (2, -1.0, 0), **N**=8, **Method**=2;

In this example, the multipole is a dipole with field strength -1.0.

**Example 2:**

QF**: multipole, L**=0.70, **HOM** = (2, 2.50, 0.0, 4, 1.01e7, 0.0), **N**=8, **Method**=4;

In this example, the multipole is a dipole with quadrupole filed errors.

The parameters of ‘multipole’ are optional, the default value of ‘method’ is 0, and the default values for the other parameters are 0.

### wiggler

To define a wiggler, the format is: (To be updated……..)

symbol: **Wiggler**, **L** = <length>,**BoBrhoV** = <B/Brho [1/m]>,**BoBrhoH** = <B/Brho [1/m]>,

**Lambda** = <period [m]>,**kxV** = <[m]>,**kxH** = <[m]>,**phi** = <phase [deg]>,

**harm**(n, kxV, BoBrhoV, kxH, BoBrhoH, phi)...**N** = <no of integration steps>,

**Method** = <method>;

‘L’ is the wiggler length with unit [m], ‘BoBrhoV’ is the normalized vertical field with unit [m-1], ‘BoBrhoH’ is the normalized horizontal field with unit [m-1], ‘Labmda’ is the period length with unit [m]. kxV

**Example 1:**

U143: **wiggler**, **L**=4.80, **K**=0.5, **Lambda**=0.15, **N**=20, **Method**=0;

**Example 2:**

EPU: **wiggler**, **L**=4.80, **Lambda**=0.15, **N**=20, **Method**=0,

**harm**=(3, kxV\_3, BoBrhoV\_3, kxH\_3, BoBrhoH\_3, phi\_3);

### field map

To read field map from a file, use the format: (To be updated……..)

<symbol> : **Fieldmap**,

**L** = <length [m]>,

**N** = <no of integration steps>,

**file1** = <file name (lower case)>

‘L’ is the length of the element, ‘N’ is the number of integration steps in the code, ‘file1’ is the field map file.

**Example:**

FM: **Fieldmap**, **L** = 1.0, **N** = 20, **file1** = "U19\_Bxyz.dat";

### Insertion device

To define the insertion device, the format is:

Symbol : **insertion** , **scaling1** = 1/0, **scaling2**=1/0,**method** = interpolation\_method,

**N**=Number of slice, **file1** = name of the file with 1st order radia map,

**file2** = name of the file with 2nd order radia map;

‘symbol’ is the user defined element name; ‘insertion’ is a keyword to denote this element is an insertion device; ‘scaling1’ is the scaling factor for the 1st order field map, ‘scaling2’ is the scaling factor for the 2nd order map; keyword ‘method’ define the order of symplectic interpolation method in the code, the value of 1 is linear interpolation, 3 is spline interpolation; keyword ‘N’ means how many pieces of this element is cut when it is treated in the code. The 1st and 2nd order of insertion device field are read from the files generated by RADIA, user need to specify the files use ‘file1’ and ‘file2’. If user does not specify the file name with the file path, then the code will look for the files in the current working directory. The path of the Radia map file must be in small letters, otherwise the code can’t find the file.

**example**:

WIGSLIC**:insertion, N** = 10**, scaling1**=1.0**, scaling2**=1.0**, method**=2**,**

**file1=** "/home/sources/physmach/brunelle/tracy2.7/w150g11pole60\_oppose\_radia\_pour\_tracy.txt",

**file2=** "/home/sources/physmach/brunelle/tracy-2.7/w150g11pole20\_fin.dat";

All the parameters for ‘insertion’ is optional, the default value for scaling1 and scaling2 are 1, the default ‘method’ is 3 which means spline interpolation, the default ‘N’ is 1, the default values for all the other parameters are 0.

### RF cavity

To define the RF cavity, use the command:

Symbol: **cavity**, **Frequency** = RF frequency, **Voltage** = RF voltage, **Phase** = synchrotron phase,

**harnum** = harmonic number of the RF cavity;

‘symbol’ is the element name, ‘cavity’ means that this element is RF cavity; ‘frequency’ defines the RF frequency with unit [Hz]; ‘voltage’ defines the RF voltage with unit [Volt]; ‘phase’ is the synchrotron phase with unit [degree]; ‘harnum’ is the harmonic number.

**Example:**

CAV: **Cavity**, **Frequency** = 499.95e6, **Voltage**=1.22e6, **phase** = 30, **harnum**=328;

The harmonic number of the RF cavity is mandatory, and the other parameters of ‘cavity’ are optional, the default value are 0.

### corrector

To define the corrector, use the command:

Symbol: **corrector**, **horizontal/vertical**, **method** = integrated method;

‘symbol’ is the element name, ‘corrector’ means that this element is a corrector, ‘horizontal’ means the element is a horizontal corrector; ‘vertical’ means the element is a vertical corrector; keyword ‘method’ define the order of symplectic integration method in the code, the value ‘1’ means 1st order, ‘2’ means 2nd order, and ‘4’ means 4th order.

**Example:**

{\*\* Horizontal correctors \*\*}

CH : **corrector**, **horizontal**, **method**=intmeth;

{\*\* Vertical correctors \*\*}

CV : **corrector**, **vertical**, **method**=intmeth;

The parameter of ‘corrector’ is optional, the default value for ‘method’ is **0.**

**Notice:**

**The ‘symbol’ of correctors are special!!!!**

For lattice with correctors,

* User must specify the name of corrector in the input file “\*.prm” with the commands:

**h\_corr** HCM

or

**v\_corr** VCM

Here ‘HCM’ is the name of the corrector defined in the lattice for horizontal orbit correction; ‘VCM’ is the name of the corrector defined in the lattice for vertical orbit correction.

### marker

To define a marker, use the command:

Symbol: **marker**;

‘symbol’ is the name of the element, ‘marker’ means the element is a marker in the lattice, it does not have length or field strength, etc.

### BPM

BPM is a special marker in the lattice, the ‘symbol’ name must be ‘BPM’. User can

define the BPM as:

**BPM : type;**

Normally It’s type is defined as ‘Marker’, but in order to include the misalignment error of BPM into the lattice, it must be defined as ‘Beam Position Monitor’ which is in fact multipole type, since only the element with multipole type is saved with displacement error, field error, etc.

**Notice:**

For lattice with BPMs,

* User must specify the name of BPM in the input file “\*.prm” with the commands:

**bpm**  beaPosMonitor

Here ‘beaPosMonitor’ is the name of the BPMs defined in the lattice.

### Girder

Girder is a special element, it’s the girder used in the real machine to support the magnetic elements and other elements.

Symbol: **type**;

Normally It’s type is defined as ‘Marker’, but in order to include the misalignment error of girder into the lattice, it must be defined as ‘multipole’, since only the element with multipole type is saved with displacement error, field error, etc.

For convenience, it’s better to define the beginning of the girder and also the end of the girder, and the elements between the beginning and end of the girders are the elements who are put on the girder in the real machine.

**Notice:**

For lattice with girders,

* User must specify the name of girder in the input file “\*.prm” with the commands:

**gs** Girder\_Start

**ge** Girder\_End

Here ‘Girder\_Start’ is the name of the start of girder defined in the lattice, ‘Girder\_End’ is the name of the end of girder defined in the lattice,

### Element block

To construct the element block, use the following format:

Symbol: elem1, elem2,…., block1,block2;

‘symbol’ is the name of the element block, and ‘elem1’, ‘elem2’, ‘block1’, block2’ are the element or sub element blocks in this element block. If there are N the same element/block subsequently, user can use ‘N\*element/block’ to simply the definition. For example:

SINJ: SD1a,ssep,3\*SEP,esep,SD1c,eHU600,SD1d;

In the element block, there are 9 elements/blocks, and 3 elements/blocks ‘SEP’ subsequently.

### Cell

user can define the cell structure using the command:

**CELL** : <block name>, **SYMMETRY**=<symmetry>;

<block name> is the name of a block; <symmetry> is the number of super symmetry or the number of the block in the ring. For example:

**CELL** : BL1, Symmetry=12;

This example defines the cell with block ‘BL1’, and the number of super symmetry is 12.

### ring

To define the ring, use the command:

**RING**: elem,…,block….

It’s similar to define a element block, but must with the fixed symbol name ‘RING’. For example:

**RING**: DEBUT,SUP1,SUP2,SUP3,SUP4,CAV,FIN;

### End line

To end the lattice file, user need to use the following command at the end of the lattice file:

**end;**

This command is mandatory.

## Multipole error file

The multipole error of the lattice elements can be defined in a file, and then the file is read into the lattice. User can define the systematic or random multipole error of the lattice elements.

There are two ways to define the multipole error, one way is to define the error for all the families in one type, for example, the error for all the quadrupoles; another way is to define the error for one family, for example, the “Q1” family of the quadrupoles.

This multipole error file only works for soleil lattice, in which all the multipole errors are added on the corresponding components of the magnets, since the sextupoles are combined magnets, they function as sextupole, horizontal/vertical correctors, and skew quadrupoles. The multipole errors of horizontal/vertical correctors and skew quadrupoles are added on the sextupoles which have their function.

### Define systematic multipole errors

To define the systematic multipole error of the element, the user just need to follow the rules as below.

* input format of multipole error:

keywords/name sys r0(radius where is error is measured)

n(order of error) Bn(n-th B component of the field) An(n-th A component of the field)

n, Bn, An,......

* The "keywords" means one type of lattice elements or the name of the family, and keywords of the type of lattice elements are:

dip dipole

quad quadrupole

sext sextupole

hcorr horizontal corrector

vcorr vorizontal corrector

qt skew quadrupole

* “sys” is a keyword to denote that user are setting the systematic multipole error.
* Bn defines the upright component of the magnetic field, then for the component of a skew quadrupole or a vertical corrector, Bn = 0
* An defines the skew component of the magnetic field, then for the component of a dipole or upright quadrupole, An = 0.
* The line start with ‘#’ is comment line.
* The blank line in the multipole definition file is neglected by the code.

For the soleil lattice, the use can define the multipole errors for the type or each family. But to define the multipole error for quadrupole type, user can define the multipole errors for each quadrupole family; or first define the quadrupole type, and then define the multipole errors on Q2 and Q7 families for the lattice with full quadrupoles or QP2a, QP2b, QP7a and QP7b families for the lattice with quadrupoles which are cut into two halves. This is due to that Q2/QP2a/QP2b and Q7/QP7a/QP7b are the long quadrupoles in the lattice, the multipole errors on them are different from the ones on the other short quadrupoles.

The following is an example file to define systematic multipole errors on Soleil lattice:

#dipole

dip sys 20e-3 2 2.2e-40 0.0 3 -3.0e-4 0.0 4 2.0e-5 0.0 5 -1.0e-4 0.0 6 -6.0e-5 0.0 7 -1.0e-4 0.0

#quadrupole

#for all short quadrupoles

quad sys 30e-3 6 2.4e-4 0.0 10 0.7e-4 0.0 14 0.9e-4 0.0

#for all long quadrupoles qp2 and qp7

qp2a sys 30e-3 6 0.7e-4 0.0 10 1.9e-4 0.0 14 1.0e-4 0.0

qp2b sys 30e-3 6 0.7e-4 0.0 10 1.9e-4 0.0 14 1.0e-4 0.0

qp7a sys 30e-3 6 0.7e-4 0.0 10 1.9e-4 0.0 14 1.0e-4 0.0

qp7b sys 30e-3 6 0.7e-4 0.0 10 1.9e-4 0.0 14 1.0e-4 0.0

#for all short quadrupoles,sextupole mesure quadrupoles longs

quad sys 30e-3 3 -1.6e-4 0.0 4 -3.4e-4 0.0

#for long quadrupoles qp2 and qp7

qp2a sys 30e-3 3 2.9e-4 0.0 4 -8.6e-4 0.0

qp2b sys 30e-3 3 2.9e-4 0.0 4 -8.6e-4 0.0

qp7a sys 30e-3 3 2.9e-4 0.0 4 -8.6e-4 0.0

qp7b sys 30e-3 3 2.9e-4 0.0 4 -8.6e-4 0.0

# for sextupoles

sext sys 32e-3 5 5.4e-4 0.0 7 3.3e-4 0.0 9 -4.7e-4 0.0 15 -9.0e-4 0.0 21 -20.9e-4 0.0 27 0.8e-4 0.0

# for horizontal correctors, all An=0

hcorr sys 35e-3 5 0.430 0.0 7 0.063 0.0 11 -0.037 0.0

# for vertical correctors, all Bn=0

vcorr sys 35e-3 5 0.0 -0.430 7 0.0 0.063 11 0.0 0.037

# for sextupole associated skew quadrupole, all Bn=0

# qt sys 35e-3 4 0.0 -0.0

qt sys 35e-3 4 0.0 -0.680

### Define ramdom multipole error

To define random multipole errors on the lattice elements, user need to follow the same rule as the ones to define systematic multipole error, and user need to replace “sys” by “rms”. For exampole:

seed seed\_number

quad rms 30e-3 6 2.4e-4 0.0 10 0.7e-4 0.0 14 0.9e-4 0.0

The random multipole error is multiplied by the random scale factor; the new value is added to the corresponding components of the magnetic field. The random scale factor is generated by a random function which follows the normal distribution( mean value is 0 and standard deviation is 1), The cut value for the normal distribution function is 2. If user does not define seed for the random function before the setting of errors, then the code will stop and give an error message.

Here is example file to define random multipole error in the lattice:

#define seed for the ramdom multipole error

seed 1000000

#dipole

# dip 20e-3 2 2.2e-4 0.0 3 -3.0e-4 0.0 4 2.0e-5 0.0 5 -1.0e-4 0.0 6 -6.0e-5 0.0 7 -1.0e-4 0.0

dip rms 20e-3 2 2.2e-40 0.0 3 -3.0e-4 0.0 4 2.0e-5 0.0 5 -1.0e-4 0.0 6 -6.0e-5 0.0 7 -1.0e-4 0.0

#quadrupole

quad rms 30e-3 3 -1.6e-4 0.0 4 -3.4e-4 0.0 6 2.4e-4 0.0 10 0.7e-4 0.0 14 0.9e-4 0.0

Q2 rms 30e-3 3 2.9e-4 0.0 4 -8.6e-4 0.0 6 0.7e-4 0.0 10 1.9e-4 0.0 14 1.0e-4 0.0

Q7 rms 30e-3 3 2.9e-4 0.0 4 -8.6e-4 0.0 6 0.7e-4 0.0 10 1.9e-4 0.0 14 1.0e-4 0.0

## Misalignment error file

The misalignment error of the lattice elements can be defined in a file, and then the file is read into the lattice. User can define the systematic or random misalignment error of the lattice elements.

There are two ways to define the misalignment error, one way is to define the error for all the families in one type, for example, the error for all the quadrupoles; another way is to define the error for one family, for example, the “Q1” family of the quadrupoles.

The systermatic misalignment error file only works for lattice with both full and half quadrupoles; The random misalignment error file only works for lattice with full quadrupoles.

### Define systematic misalignment errors

To define the systematic misalignment error of the element, user just need to follow the rules as below.

* input format of misalignment error:

type/family name sys dx dy dr

* The "keywords" means one type of lattice elements or the name of the family, and keywords of the type of lattice elements are:

All all the elements in the lattice

girder girder

dipole dipole

quad quadrupole

sext sextupole

bpm beam position monitor

family name family name of the elements

* “sys” is a keyword to denote that user are setting the systematic displacement error.
* dx defines the displacement in x direction with unit [m].
* dy defines the displacement in y direction with unit [m].
* dr defines the rotation angle with unit [rad]).
* The line start with ‘#’ is comment line.
* The blank line in the misalignment error file is neglected by the code.

The following is an example file to define systematic multipole error on Soleil lattice:

#-----------------------------------------------------------------------

# systematic alignment error for SOLEIL

# name x(m) y(m) r (rad)

#-----------------------------------------------------------------------

girder sys 100.0e-6 100.0e-6 0.5e-03

quad sys 30.0e-6 30.0e-6 80.0e-06

sext sys 30.0e-6 30.0e-6 100.0e-06

dipole sys 500.0e-6 500.0e-6 0.2e-03

### Define random misalignment errors

To define random misalignment errors on the lattice elements, user need to follow the same rule as the ones to define systematic misalignment error, and user need to replace “sys” by “rms”. i.e.:

* input format of misalignment error:

seed seed\_number

type/family name rms dx dy dr

The random misalignment error is multiplied by the random scale factor; the new value is added to the corresponding components of the misalignment components. The random scale factor is generated by a random function which follows the normal distribution (mean value is 0 and standard deviation is 1), The cut value for the normal distribution function is 2.

If user does not define seed for the random function before the setting of errors, then the code will stop and give an error message.

Here is example file to define random misalignment error in the lattice:

#-----------------------------------------------------------------------

# random alignment error for SOLEIL

# name x(m) y(m) r (rad)

#-----------------------------------------------------------------------

girder rms 100.0e-6 100.0e-6 0.5e-03

quad rms 30.0e-6 30.0e-6 80.0e-06

sext rms 30.0e-6 30.0e-6 100.0e-06

dipole rms 500.0e-6 500.0e-6 0.2e-03

## Vacuum chamber file

User can define the script to set the vacuum chamber limitation around the ring. The characteristic for the vacuum chamber script are:

* Lines start with "#" are comment.
* The format of the vacuum chamber definition is

MK1, MK2, minimum x, maximum x, minimum y, maximum y.

* To set the vacuum chamber, it is needed to add two markers in the lattice, such as MK1 and MK2, MK1 is before the first element and MK2 is after the end element of the vacuum chamber region.
* The numbers of MK1 and MK2 are the same in the lattice.
* The units are [meter] for minimum x, maximum x, minimum y, maximum y.
* The first line is to define the global vacuum chamber limit around the ring, and the key words should be "Start","All".
* The following is one example of the user vacuum chamber script:

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Script to set the vacuum chamber

#

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# MK1 MK2 dxmin dxmax dymin dymax (Apertures in meter)

Start All -35e-3 35e-3 -12.5e-3 12.5e-3

#sdm1 esdm -21e-3 21e-3 -5e-3 5e-3

debut ehu600 -35e-3 35e-3 -7e-3 7e-3

ssep esep -20e-3 35e-3 -7e-3 7e-3

ssdm esdm -21e-3 21e-3 -5e-3 5e-3

ssdac esdac -35e-3 25e-3 -2.5e-3 2.5e-3