**December**

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

2 Non-parallel version Tracy 4

2.1 Compile 4

2.2 Run 4

3 Parallel version Tracy 5

3.1 Compile 5

3.2 Run 6

4 User input script 6

4.1 File path 6

4.2 File names 7

4.2.1 Lattice file name 7

4.2.2 Multipole field error file name (SOLEIL lattice) 8

4.2.3 Files of multipole field errors of correctors and skew quadrupoles (SOLEIL lattice) 8

4.2.4 File to define field strength of virtual coupling source elements (SOLEIL lattice) 8

4.2.5 Cut off value 9

4.3 Commands 9

4.3.1 Activate quadrupole fringe field 9

4.3.2 Deactivate quadrupole fringe field 10

4.3.3 Set voltage of RF cavity 10

4.3.4 Print the tracked coordinates at each element to a file 10

4.3.5 Print Twiss parameters to a user defined file 11

4.3.6 Print COD (Close Orbit Distortion) to a user defined file 11

4.3.7 Read vacuum chamber setting from a file 12

4.3.8 Read lattice element multipole field errors from a file 12

4.3.9 Read lattice element misalignment errors from a file 12

4.3.10 Closed orbit (COD) correction 12

4.3.11 Get tunes by tracking 13

4.3.12 Get chromaticities by tracking 13

4.3.13 Tune shift with amplitude 14

4.3.14 Tune shift with energy 14

4.3.15 Frequency map analysis for on momentum particle 14

4.3.16 Frequency map analysis for off momentum particle 15

4.3.17 Add coupling by the random rotation of the full quadrupoles 16

4.3.18 Add coupling by random rotation of the half quadrupoles 16

4.3.19 Calculate coupling and emittance 16

4.3.20 Calculate momentum acceptance by tracking 16

4.3.21 Read multipole field error from a file (SOLEIL lattice) 17

4.3.22 Read the sources of coupling from a file (SOLEIL lattice) 18

4.3.23 Fit tunes for the lattice with full quadrupole 18

4.3.24 Fit tunes for the lattice with half quadrupoles 18

4.3.25 Fit chromaticity 19

4.3.26 Touschek lifetime determined by RF acceptance (TO BE UPDATED) 19

4.3.27 Intra Beam Scattering (IBS)(TO BE UPDATED) 19

4.3.28 Touschek lifetime determined by the minimum value of RF acceptance and momentum acceptance (TO BE UPDATED) 20

4.3.29 Obtain phase space by tracking 20

4.3.30 Insertion device (ID) compensation (Tested for TaiWan light source; TO BE CONTINUE DEVELOPPED.) 20

5 User Defined Files 22

5.1 Lattice file 22

5.1.1 Lattice element 22

5.1.2 Syntax 22

5.1.3 Variables 22

5.1.4 Start line 22

5.1.5 Global variables 23

5.1.6 Drift space 23

5.1.7 Dipole 23

5.1.8 quadrupole 24

5.1.9 Skew quadrupole 25

5.1.10 sextupole 26

5.1.11 multipole 27

5.1.12 Wiggler (To be updated.) 28

5.1.13 field map (To be updated……..) 29

5.1.14 Insertion device 29

5.1.15 RF cavity 30

5.1.16 corrector 31

5.1.17 Marker 32

5.1.18 BPM (To be updated) 32

5.1.19 Girder 32

5.1.20 Element block 33

5.1.21 Cell 33

5.1.22 Ring 33

5.1.23 End line 33

5.2 Multipole field error file 33

5.2.1 Define systematic multipole errors 34

5.2.2 Define ramdom multipole error 35

5.3 Misalignment error file 35

5.3.1 Define systematic misalignment errors 36

5.3.2 Define random misalignment errors 36

5.4 Vacuum chamber file 37

# Introduction

Tracy is a code to do long term tracking, and is written in the mixture of c and c++. This code is kept on developing. Soleil version of Tracy 3 is a code with more flexibility and easy to use. User does not need to know the structure of the code, what they need to do is to write an input script, and then run the code.

Based on the need, user can write the files to define multipole field errors, misalignment errors of the lattice elements, and vacuum chamber, and then provide the file name in the user input script, in order to set the field errors of the lattice elements and the vacuum limit for the different region of the lattice.

Attention:

* In the user defined file, such as the user input script, the file to define multipole field errors or alignment errors, or the file to define vacuum chambers, the maximum numbers of column is 130 (Not including comment line which starts with symbol #.), and the spaces between each parameters or variables can’t contain “TAB” key, otherwise the code can’t work properly. This is because the routine to read the user defined input file is written in C. These limits will be improved in the future development.
* Although the user can define the file name whatever they want, it is suggested to name the file which is used to define multipole field error with the extension “.fe” and name the file which is used to define the misalignment error with the extension “.ae”.

There are two versions of Tracy 3, non-parallel version and parallel version. Non-parallel version Tracy is for the single computer*,* parallel version of Tracy is for the the cluster. The user defined input script “\*.prm” can be used for both parallel and non-parallel version Tracy.

# Non-parallel version Tracy

## Compile

The make file of Tracy is generated by “automake”. Based on the compilers used, user needs to update “make\_for\_gcc.sh” under path “$HOME/TracyIII/” or the “Makefile.am” under path “$HOME/TracyIII/tracy/tracy/src” and “$HOME/TracyIII/tracy/tools”. The compilers used on the server “metis” of SOLEIL Synchrotron are “gfortran”, and “gcc”.

To compile the code, run

make\_for\_gcc.sh opt

under the shell terminal, an executive file “soltracy” is automatically generated under path “$HOME/TracyIII/tracy/tools”.

## Run

To execute Tracy, user needs to provide an input script with the file extension “.prm”. For example, “Input\_test.prm” is a user defined script, user can type the command line

sol**tracy**  Input\_test.prm

under the shell terminal, and then press “return” key to execute the code. The input script file name can be defined with any valid string except it must be ended with the file extension “.prm”.

# Parallel version Tracy

In order to reduce the tracking time, the parallel version Tracy can be used on the cluster. Until now, the following three features of Tracy are parallelized:

* Frequency map analysis for on momentum particle, command “FmapFlag”
* Frequency map analysis for off momentum particle, command “FmapdpFlag”
* Track momentum acceptance at lattice elements, command “MomentumAccFlag”

## Compile

The commonly used compilers for parallel computation are MPI 2, and Intel MPI which is based on MPI 2. For the cluster of SOLEIL Synchrotron, Intel MPI is installed. To get the parallel Tracy work, three files of the non-parallel version Tracy need to be modified.

The details are shown in the following steps.

1. The path of included files of Intel MPI is added in “Makefile.am” under path “$HOME/TracyIII/tracy/tracy/src” (shown in BLUE color):

INCLUDES = -I../inc -I$(NUM\_REC)/inc

-I/opt/intel/impi/3.2.2.006/include64

1. The execute file, source file, paths of included files and library of Intel MPI are modified in “Makefile.am” under path “$HOME/TracyIII/tracy/tools” (shown in BLUE color):
   * + bin\_PROGRAMS = psoltracy
     + soltracy\_SOURCES = soltracy.cc nrutil.c nrcheck.c

nrlinwww.c nrframe.c ../tracy/src/tracy\_lib.cc ->

psoltracy\_SOURCES = psoltracy.cc nrutil.c nrcheck.c

nrlinwww.c nrframe.c ../tracy/src/tracy\_lib.cc

* + - LIBS = -L$(NUM\_REC)/lib

-L/opt/intel/impi/3.2.2.006/lib64 -L$(LIBPATH)

-lrecipes\_c\_icc -lstdc++ -lgfortran -lmpichcxx

* + - INCLUDES = -I$(TRACY\_LIB)/tracy/inc -I$(NUM\_REC)/inc -I/opt/intel/impi/3.2.2.006/include64

1. The compilers used in the parallel computing are defined in the “make\_for\_psoltracy.sh” located in path “$HOME/TracyIII” as (shown in BLUE color):

CC = mpiicc

CXX = mpiicpc

F77 = mpiifort

Depending on the compilers used to do parallel computation, user needs to update the compilers in “make\_for\_psoltracy.sh” and the paths of included files and library which are shown above with blue color.

After updating compilers, paths of included files and library for parallel computation, user can run

make\_for\_psoltracy.sh

under the shell script to compile the parallel Tracy. After compilation, the execute file “psoltracy” is automatically generated under the path “$HOME/TracyIII/tracy/tools”.

## Run

As the non-parallel version Tracy, user needs to write the commands in a script which must be with the file extension “.prm”. The syntaxes to define the script “\*.prm” are the same for both non-parallel and parallel Tracy.

To run the parallel Tracy, user need to contact the administrator of their cluster to know how to run parallel programs on the cluster. For the cluster on Synchrotron SOLEIL, the nodes used to do parallel computation are assigned by PBS (Portable Batch System), so a script is need. For example, user define the input script “test.prm” to tell Tracy what jobs are need to be done on the cluster, define script “lance\_tracy3\_parallel.sh” to assign the numbers of CPUs to do parallel computation and lance job to the SOLEIL cluster. and then type the following command under the bash shell to submit the job and run the parallel Tracy on the cluster:

lance\_tracy3\_parallel.sh test.prm

# User input script

There are two types of keywords in the user input script. The first type is to set the file, the file names, and define parameters for the related calculations; the key words for such definitions are ended with the characters “Flag”. The second type is to define Boolean commands with or without parameters to do different calculations. The rules of the definition of input scripts are:

* The blank lines and lines starting with "#" (comment line) are ignored by the code.
* Keywords without “Flag” as the final 4 characters are NOT executed according to the defined sequence in user input script. If the same command keywords are defined many times in the user script, only the last defined keyword is executed.
* The commands with the last 4 characters as “Flag” are executed according to the sequence in the code.
* The definition of lattice file at the beginning of the user script is mandatory. If the lattice file is with any of the following elements: horizontal correctors, vertical correctors, girders, BPM, skew quadrupoles, user must declare the element name at the beginning of input script.
* One keyword command uses one line. User can not define more than one command at the same line.
* Except explanation, all commands with Boolean flag are the generic commands, and can be used on all the lattice of the ring.

## File path

In the user input script, user can specify the name of the files which are used in the calculation, such as the lattice file, multipole field error file, misalignment error file, vacuum chamber file. When specifying the file name, user can provide the absolute path for the file which is not located at the current path, such as: “/home/physmach/Tracy3/tracy/soleil.lat”; or for convenience, user provides the filename without absolute path, but put the files 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 declares the files without absolute path and does not set the directory through command in\_dir; or the files specified in the script are not found at the current working path, the code will give an error message and stop 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

In Tracy 3, after reading the lattice, Twiss parameters are automatically printed to the file “linlat.out”.

If one of these elements (Horizontal/vertical correctors or beam position monitors, or girders, or skew quadrupoles.) are defined in the lattice file, for example to read the misalignment errors of the lattice element and then do orbit correction, to read multipole field errors (SOLEIL lattice), etc.; user MUST specify the names of these elements using the corresponding commands in the “\*.prm” script:

**h\_corr** HCM

**v\_corr** VCM

**gs** GS

**ge** GE

**bpm\_name** BPM

**qt** QT

Here “HCM” is the name of horizontal correctors used for horizontal orbit correction, or the horizontal correctors on which the multipole field errors are added in SOLEIL lattice; “VCM” is the name of vertical correctors used for vertical orbit correction, or the vertical correctors on which the multipole field errors are added in SOLEIL lattice; “GS” is the name of the start girder; ‘GE’ is the name of the end girder; “BPM” is the name of beam position monitors defined in the lattice file; “QT” is the name of the skew quadrupoles defined in the lattice. Generally, user needs to define horizontal, vertical correctors and BPMs when reading the misalignment errors or correcting closed orbit distortion

### Multipole field error file name (SOLEIL lattice)

User can read the multipole field errors on SOLEIL lattice from an external file; the file is specified in the user script using the following command:

**multipole\_file** multipole\_file\_name

Here “multipole\_file\_name” is the user defined multipole field error file, the format of this file is given in section 5.2.

If the multipole field errors of horizontal, vertical correctors and skew quadrupoles are defined in the “multipole\_file”, user **MUST** specify the names of these lattice elements in the “\*.prm” file as the following example:

**h\_corr** CH

**v\_corr** CV

**qt**  QT

Here “CH”, “CV” and “QT” are the names of horizontal, vertical correctors, and skew quadrupoles respectively which are defined in the lattice file.

### Files of multipole field errors of correctors and skew quadrupoles (SOLEIL lattice)

Horizontal, vertical correctors and skew quadrupoles are integrated with the sextupole quadrupoles on SOLEIL storage ring. To define the multipole field errors on these elements, user need to define the orders and relative strengths of multipole field errors in the multipole field error file (section 5.2) and specify the file name in the user input script “\*.prm” (section 4.2.2), then specify the file names as the following example:

**fic\_hcorr** corh.txt

**fic\_vcorr** corv.txt

**fic\_skew** corqt.txt

Here “corh.txt” and “corv.txt” are the files with measured current values corh, corv, corqt (with unit [Ampere]) for the horizontal, vertical correctors and skew quadrupoles respectively. Based on the measured current values, user can get the integrated field strength as:

Hcorr\_strength [*T.m*] = corh \*\_convHcorr /brho; (**horizontal correctors**)

Vcorr\_strength [*T.m*] = corv \*\_convVcorr /brho; (**vertical correctors**)

Qt\_strength [*T.m*] = corqt \*\_convQt /brho; (**skew quadrupoles**)

here brho is the momentum rigidity, and the conversion constants between current and field are \_convHcorr = 8.14e-4, \_convVcorr = 4.642e-4, \_convQt = 93.83e-4.

For SOLEIL lattice, the SAME ORDER of multipole field errors on the same elements are added together, so the SAME ORDER of horizontal/vertical, skew quadrupoles, and sextupoles are added together since these elements are integrated at the same magnets.

### File to define field strength of virtual coupling source elements (SOLEIL lattice)

On SOLEIL storage ring, the coupling is thought to come from the rotation of quadrupoles and vertical displacements of sextupoles. The strengths of these coupling sources are written in a file, then read by the following command:

**virtualskewquad\_file** virtual\_skew\_quad\_currents.txt

Here “virtual\_skew\_quad\_currents.txt” is the user defined file with strength of vertical coupling source.

In order to use this command to read the virtual sources of coupling, user needs to define the virtual coupling element as a skew quadrupole in the lattice file:

SQ: quadrupole, tilt=45.0, K= 0.0, method=4, N=1;

and this virtual coupling element MUST be with the name “SQ”. Now there 152 elements defined as the virtual coupling sources in the SOLEIL lattice. The syntaxs to define skew quadrupole are explained in section 5.1.9.

The measured current value qtcorr[i] of the ith coupling source is converted to the corresponding integrated skew quadrupole field strength corr\_strength as

corr\_strength [*m-1*] = qtcorr[i]\*conv/brho;

here brho is momentum rigidity, and the conversion constant conv is 93.88e-4 [*A-1.T*].

### Cut off value

Set the cut off value of all random distribution (Gaussian distribution) to “n” times of the RMS value sigma:

**normalcut** n

## Commands

The following commands turn on the boolean flags in the code to set the machine parameters and carry on different calculations. All these commands are optional; user can choose whichever they need. If user wants 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 (add “#” at the beginning of command line) 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 execute the command according to the sequence defined in the input script “\*.prm”. For example,

**FitTune4Flag**     qp7a qp7b  qp9a qp9b  18.18 10.28

**ReadMultipoleFlag**

FitTune4Flag     qp7a qp7b  qp9a qp9b  18.202 10.317

The code will fit tunes to the target tunes (18.18 10.28), and then read multipole field errors into the lattice, then fit the tunes to a new set of values (18.202 10.317).

* User can define the same Boolean commands as often as they want in the same input script; but user can only define maximum 500 commands in one input script.
* The user defined script can be used both for the non-parallel and parallel version Tracy; the output files are the same for both versions.

### Activate quadrupole fringe field

To activate quadrupole fringe field, use the command:

**QuadFringeOnFlag**

in the “\*.prm” script. With this command, user can define the fringe field at the entrance or exit of the quadrupole together with the command FF1 = 1 or FF2 = 1 of the quadrupoles which are defined in the lattice file; if FF1 or FF2 not equals to 1, then there is no fringe field at the entrance or exit of the quadrupoles even if **QuadFringeOnFlag** is active in the “\*.prm” file.

This flag is a global flag, if user set this flag in the input script, it will always have effects until this flag is reset.

### Deactivate quadrupole fringe field

To deactivate quadrupole fringe field, use the command:

**QuadFringeOffFlag**

With this command, user can deactivate the fringe field at the entrance and exit of the quadrupole, even if FF1 = 1 or FF2 = 1 for the quadrupole in the lattice file.

This flag is a global flag, if user set this flag in the input script, it will always have effects until this flag is reset.

### Set voltage of RF cavity

User can reset the RF voltage by setting “RFvoltageFlag” to replace the value of RF voltage which is defined in the lattice. For example:

**RFvoltageFlag** 3000000

Here “RFvoltageFlag” is the name of the keyword command, ‘3000000’ is the value of RF voltage with the unit [volt].

If the ring has more than one RF cavities, the related parameters are defined as the total values for one RF cavity.

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

To print the coordinates tracked around COD at each element, use the command:

**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 values of “PrintTrackFlag” are shown in Table 1.

Table 1 Parameters of command "PrintTrackFlag".

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Default value** | **Unit** |
| track\_file | File to save the tracked coordinates around each lattice element | track.out | - |
| x | Start tracking horizontal coordinate | 0.001 | [m] |
| px | Start tracking horizontal canonical momentum px normalized by reference momentum p0, that is: px = px/p0. Normally px is approximate as horizontal deviation with unit [rad]. | 0.0 | - |
| y | Start tracking vertical coordinate | 0.0 | [m] |
| py | Start tracking vertical canonical momentum py normalized by reference momentum p0, that is: py = py/p0. Normally px is approximate as vertical deviation with unit [rad]. | 0.0 | - |
| delta | Start tracking relative energy offset | 0.0 | - |
| ctau | Start tracking longitudinal coordinate | 0.0 | [m] |
| nturn | Number of turn for tracking | 50 | - |

### Print Twiss parameters to a user defined file

With the command “PrintTwissFlag”, Tracy 3 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 parameters to a default file “twiss.out”.

### Print COD (Close Orbit Distortion) to a user defined file

With the command “PrintCODFlag”, the code will print the close orbit distortion 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 outuput file of the command **PrintCODFlag**, the data is saved in the following format:

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

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

#

Where “#” denotes the command line, the meanings of the above parameters are shown in Table 2.

Table 2 Parameters in the output file of close orbit.

|  |  |  |
| --- | --- | --- |
| Name | Description | Unit |
| i | Number of the element | - |
| name | Element name defined in lattice | - |
| s | Longitudinal length | [m] |
| 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 | [m] |
| nux | Horizontal tune | - |
| betay | Vertical beta function | [m] |
| nuy | Vertical tune | - |
| xcod | Horizontal closed orbit distortion | [mm] |
| ycod | Vertical closed orbit distortion | [mm] |
| dSx | Horizontal displacement of the element | [mm] |
| dSy | vertical displacement of the element | [mm[ |
| dipx | Horizontal dipole strength of the element | [mrad] |
| dipy | vertical dipole strength of the element | [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 format of the chamber file is given in section 5.4.

### Read lattice element multipole field errors from a file

To read the multipole field errors of the lattice elements from the user defined file, use the command:

**ReadfefileFlag dip.fe**

Tracy will read the systematic and random multipole field errors of the lattice elements defined in the file “dip.fe”, and then replace the corresponding field components of the elements with the new field errors. The formats to specify the systematic and random multipole field errors in a file are given in section 5.2.

### Read lattice element misalignment errors from a file

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

**ReadaefileFlag** dip.ae

Tracy will read the systematic and random misalignment errors of the lattice elements from the file “dip.fe”, and replace the misalignment errors of the corresponding components of the elements. The formats to define the systematic and random misalignment errors of the lattice elements in a file are given in section 5.3.

### Closed orbit (COD) correction

The orbit distortion is corrected using SVD (Singular Value Decomposition) method in Tracy 3. In order to do orbit correction, user needs to call the command

**CODCorrectFlag**

and then specify the following parameters in the user defined “\*.prm” file.

* Specify the element names of horizontal, vertical correctors, and beam position monitors used in the orbit correction as the following examples:

**h\_corr** HC

**v\_corr** VC

**BPM**  bpm

* User also need to specify the states of the correctors to trigger on/off the correction using the following parameters:

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

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

In the file “hcorr\_56nom.state”, a list of numbers (1 or 0) are given to the horizontal correctors, according to the sequence in the lattice; “1” means the corresponding corrector is used for horizontal orbit correction, “0” means this corrector is not used in the horizontal orbit correction. The definition rules of vertical corrector states in “vcorr\_56nom.state” are the same as “vcorr\_56nom.state”.

* This parameter defines number of iterations to correct the orbit distortion, this value should be an integer number not smaller than 1.

**n\_orbit** 3

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

**nwh** 60

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

**nwv** 60

In Tracy 3, during the closed orbit correction:

1. Beam response matrices between beam position monitors and horizontal/vertical correctors are calculated and written to the files “svdh.out”/ “svdv.out”, respectively. The maximum number of horizontal/vertical correctors used for orbit correction is 250.
2. The code corrects the closed orbit distortion.
3. Horizontal and vertical orbits at the locations of all beam position monitors during the correction are saved to the files “horbit.out” and “vorbit.out”, respectively.
4. A file “OrScanFile.out” will be saved with the summaries of the mean and RMS values of the orbits before and after correction.
5. Finally, Twiss parameters, closed orbit distortion at the lattice elements are saved to a summary file “summary\_miserr\_codcorr.out”, the format of this file is explained in Table 2.

### Get tunes by tracking

To get tunes by tracking, use command:

**TuneTracFlag**

The tunes obtained by tracking are printed on the screen.

### Get chromaticities by tracking

To get chromaticity by tracking, use command:

**ChromTracFlag**

The chromaticities obtained by tracking are printed on the screen.

### Tune shift with amplitude

To calculate tune shift with amplitude, one needs 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 meanings of parameters and default values of command “AmplitudeTuneShiftFlag” are shown in Table 3. If user uses the command AmplitudeTuneShiftFlagwithout parameters, then the code will use all the default values.

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

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Meaning** | **Default values** |
| 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 needs to use the command:

**EnergyTuneShiftFlag**  nudp\_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 4.

If user uses command EnergyTuneShiftFlagwithout parameters, then the code will use all the default values.

Table 4 Parameters of the command to calculate tune shift with energy

|  |  |  |
| --- | --- | --- |
| **parameter** | **meaning** | **default value** |
| nudp\_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 analysis for the on momentum particle, use the command:

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

For example:

**FmapFlag** fmap.out 31 21 516 0.025 0.005 0.0 true

The meaning of parameters and default values of this command are shown in Table 5. If user uses command FmapFlagwithout parameters, then the code will use all the default values.

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

|  |  |  |
| --- | --- | --- |
| **Parameters** | **meaning** | **Default values** |
| 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 analysis for the 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 6. If user only uses the command **FmapdpFlag** but without defining all the parameters, then the code uses the default values.

Table 6 Parameters of the command “FmapdpFlag”.

|  |  |  |
| --- | --- | --- |
| **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 coupling by the random rotation of the full quadrupoles

To simulate coupling in the lattice, use can add the random rotation error to all the full quadrupole, using 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 angles of all the quadrupoles with the unit [rad].

After setting the rotation error in the lattice, the code will generate a file with the file name “flat\_file\_errcoupling\_full.dat” at the current working directory, user can check the error setting of quadrupoles in this file; then the coupling will be calculated and Twiss parameters after adding the random rotation errors will be saved to the file “linlat\_errcoupling.out”.

### Add coupling by random rotation of the half quadrupoles

In order to get the beam parameters in the middle of the quadrupoles, each quadrupole in the lattice can be cut into two parts. In such case, the coupling of the lattice can be generated by random rotation of all the half quadrupoles in the lattice, using 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 with the file name “flat\_file\_errcoupling\_half.dat”, user can check the error setting of quadrupoles in this file. After adding the random rotation errors, the coupling will be calculated and Twiss parameters will be saved to the file “linlat\_errcoupling2.out”.

This command is dedicated for Soleil lattice in which each quadrupole is cut into two half quadrupoles.

### Calculate coupling and emittance

To calculate coupling and emittance, use command:

**CouplingFlag**

After calculation, the coupling and the emittance will be printed on the screen, and the Twiss parameters will be automatically saved to the file “linlat\_coupling.out”.

### Calculate momentum acceptance by tracking

The following command calculate momentum acceptance at a predefined lattice region by tracking:

**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 7. If user uses **MomentumAccFlag** without parameters,then the code will use the default values.

Table 7 Parameters of the command to calculate momentum acceptance

|  |  |  |
| --- | --- | --- |
| **parameter** | **meaning** | **Default value** |
| MomAccFile | File to save the tracked momentum acceptance at each element; 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 | End element in the lattice for tracking | 108 |
| nstepp | Number of steps to do tracking in the positive energy range | 100 |
| nstepn | Number of steps to do 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 | Negative end 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 field error from a file (SOLEIL lattice)

After defining the file names of multipole field errors on SOLEIL storage ring (section 4.2.2 and 4.2.3), use the command:

**ReadMultipoleFlag**

to read multipole field errors and set the corresponding values to SOLEIL lattice. The multipole field errors of correctors and skew quadrupoles are added on the thick sextupoles which are integrated at the same magnets. The format of multipole errors file is given in section 5.2.

After setting the multipole field errors 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 the field components of the lattice elements in this file to verify the multipole field errors.

### Read the sources of coupling from a file (SOLEIL lattice)

The sources of coupling on SOLEIL storage ring can be read from an external file. Use the command:

**ReadVirtualSkewquadFlag**

to read and set the field strength to the virtual skew quadrupoles. Currently this command only works for Soleil lattice.

The coupling sources MUST be defined as the skew quadrupoles with the name “SQ”, the rules and related information are explained in section 4.2.4.

### Fit tunes for the lattice with full quadrupole

Betatron tunes can be fit using two families of quadrupoles. The command is:

**FitTuneFlag** Quad1 Quad2 nux nuz

For example:

**FitTuneFlag** q7 q9 18.202 10.317

The parameters of this command are shown in Table 8.

Table 8 Parameters of the command “FitTuneFlag”.

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Meaning** | **Default values** |
| Quad1 | Quadrupole family used to fit the tunes | - |
| Quad2 | Quadrupole family used to fit the tunes | - |
| nux | Target horizontal tune | 0.0 |
| nuz | Target vertical tune | 0.0 |

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

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

### Fit tunes for the lattice with half quadrupoles

For the lattice with each quadrupole cut into two pieces, betatron tunes can be fit using two families of quadrupoles. The command is:

**FitTune4Flag** Q1a Q1b Q2a Q2b nux nuz

The parameters of this command are shown in Table 9.

Table 9 Parameters of the command “FitTune4Flag”.

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Meaning** | **Default values** |
| Q1a | First half of the quadrupole family used to fit the tunes | - |
| Q1b | Second half of the quadrupole family used to fit the tunes | - |
| Q2a | First half of the quadrupole family used to fit the tunes | - |
| Q2b | Second half of the quadrupole family used to fit the tunes | - |
| nux | Target horizontal tune | 0.0 |
| nuz | Target vertical tune | 0.0 |

For 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 pieces of the full quadrupole “qp7”, and “qp9a” and “qp9b” are the two half pieces of the full quadrupole 'qp9'.

After fitting the tunes, the field strengths of fitted quadrupole before and after the fitting are printed to 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.

### Fit chromaticity

Chromaticities can be fit using two families of sextupoles, the command is:

**FitChromFlag**  SX1 SX2 epsilon\_x epsilon\_z

The parameters of this command are shown in Table 10.

Table 10 Parameters of the command “FitChromFlag”.

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Meaning** | **Default values** |
| SX1 | First sextupole family used to fit the chromaticities | - |
| SX2 | Second sextupole family used to fit the chromaticities | - |
| epsilon\_x | Target horizontal chromaticity | 0.0 |
| epsilon\_z | Target vertical chromaticity | 0.0 |

For example:

**FitChromFlag**  sx9 sx10 2.0 2.6

After fitting the chromaticites, the field strengths of fitted sextupoles before and after the fitting are printed to 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 BE UPDATED)

To calculate Touschek lifetime, use the following command:

**TouschekFlag**

Here the momentum acceptance is limited by the RF acceptance.

### Intra Beam Scattering (IBS)(TO BE UPDATED)

To calculate Intra Beam Scattering, use the command:

**IBSFlag**

### Touschek lifetime determined by the minimum value of RF acceptance and momentum acceptance (TO BE UPDATED)

Touschek lifetime can be calculated by

**TousTrackFlag**

In this case, the energy acceptance at each lattice element is limited by the minimum value of RF acceptance and momentum acceptance obtained by tracking, andthe chamber file MUST be defined in the user script.

### Obtain phase space by tracking

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

For example:

**PhaseSpaceFlag** phasespace.out 6D 1e-6 0.0 1e-6 0.0 0.012 0.0 1000 false

The meanings of parameters and defaults values of PhaseSpaceFlag are shown in table Table 11. If user uses **PhaseSpaceFlag** without parameters,then the code will use the default values.

Table 11 Parameters of the command PhaseSpaceFlag to calculate 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 canonical momentum/derivative at the start point of tracking | 0.0 |
| Phase\_Y | vertical coordinate at the start point of tracking | 0.0 |
| Phase\_Py | vertical canonical momentum/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 |

### Insertion device (ID) compensation (Tested for TaiWan light source; TO BE CONTINUE DEVELOPPED.)

To compensate the beta beat introduced by the insertion devices, several families of Quadruoples can be used. Defining the following command in the “\*.prm” can active this action:

**IDCorrFlag**

User also needs to define the following parameters used for the compensation of insertion device:

**N\_calls** 1

**N\_steps** 1

**N\_Fam** 11

**IDCquads** qs1 qs2 qs3 qs4 qs5 ql1 ql2 ql3 q1 q2 q3

**scl\_dbetax** 5e-1

**scl\_dbetay** 5e-1

**scl\_dnux** 0.1

**scl\_dnuy** 0.1

**scl\_nux** 1e1

**scl\_nuy** 1e1

**ID\_step** 0.7

The meanings of the above commands and the default values used to do ID compensation using quadrupoles are shown in Table 12.

Table 12 Parameters of commands to do ID compensation using quadrupoles.

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Meanings** | **Default values** |
| N\_calls | Number of calls to do ID compensation | 1 |
| N\_steps | Number of steps. | 1 |
| N\_Fam | Number of quadrupole families used to do ID correction. | 15 |
| IDCquads | Name of quadrupole families used to do ID correction. | - |
| scl\_dbetax | Scaling weight factor of the change step of horizontal beta function during the ID correction. | 1 |
| scl\_dbetay | Scaling weight factor of the change step of vertical beta function during the ID correction. | 1 |
| scl\_dnux | Scaling weight factor of the change step of horizontal tune during the ID correction. | 0.1 |
| scl\_dnuy | Scaling weight factor of the change step of vertical tune during the ID correction. | 0.1 |
| scl\_nux | Scaling weight factor of horizontal tune during the ID correction. | 100 |
| scl\_nuy | Scaling weight factor of vertical tune during the ID correction. | 100 |
| ID\_step |  | 0.7 |

# User Defined Files

## Lattice file

The followings are the rules to define a lattice file used in Tracy 3.

### Lattice element

The nth normalized field strength of the lattice element is defined as in Tracy:

,

where  is magnet rigidity with the nominal momentum  and the electron charge . For example, for sextupole,  is defined as

.

**Notes:**

In AT (Accelerator Toolbox) and Beta code, the definition of are the same as in Tracy. In MADX and ELEGANT, the nth normalized field strength of the lattice element is defined as



### 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 in the lattice.
* User can define any lattice element with any valid name (but must start with a character) they want, but the element type is fixed.
* 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 variables in the lattice file. For example:

**Intmeth = 4;**

so when the code is running, each “intmeth” in the lattice file 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**, the beam energy with unit [GeV].
* **dP**, the relative momentum offset of the particle.
* **CODeps**, 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 definition of the length of the drift element 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;

Here “Symbol” is the user defined element name; “bending” is a keyword to denote this element is a dipole; all the other parameters are explained in Table 13.

Table 13 Parameters of dipole in the lattice file.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Parameter** | **Units** |
| L | length | [m] |
| T | total bending angle | degree |
| T1 | Entrance angle | degree |
| T2 | Exit Angle | degree |
| K | (For combined dipoles) quadrupole field strength if magnet length L not equal to 0 or integrated field strength if L=0. | m1 (L!=0)/unitless (L=0) |
| Method | Symplectic integration method.  The value ‘1’ means 1st order, ‘2’ means 2nd order, and ‘4’ means 4th order. | 1,2,4 |
| Gap | Distance between two poles of the dipole, the gap size determine the fringe field. If the gap size is 0, then the dipole has no fringe field. | [m], |
| N | number of slices when the dipole is treated in the code | - |

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 also 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;

Here “Symbol” is the user defined element name; “quadrupole” is a keyword to denote this element is a quadrupole or parts of a quadrupole; all the other parameters are explained in Table 14.

Table 14 Parameters of quadrupoles in the lattice file.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Parameter** | **Units** |
| L | Length of the quadrupole | [m] |
| Tilt | tilt angle of the quadrupole;  if ‘tilt’ is non-zero, then the quadrupole is a skew quadruple. | [degree] |
| K | If L ≠ 0,  then Gradient, .  If L= 0,  integrated field strength of this quadrupole component. | [m-2](If L ≠ 0)  [m-1] (If L = 0) |
| FF1 | 1 or 0.  1 means taking into account the fringe field at the left edge.  0 means not taking into account the fringe field at the left edge(user should notice, in order to take into account the fringe field of the quadrupoles, user also need to use “QuadFringeOnFlag” in the user input script) | - |
| FF2 | 1 or 0.  1 means taking into account the fringe field at the right edge.  0 means not taking into account the fringe field at the right edge(user should notice, in order to take into account the quadrupole fringe field, user need also to set “QuadFringeOnFlag” in the user input script) | - |
| Method | Order of symplectic integration method.  Value “1” means 1st order, “2” means 2nd order, and “4” means 4th order. | 1,2,4 |
| N | Pieces of the quadrupole to be 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 “skewquad” 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;

Here “symbol” is the user defined element name; “sextupole” is a keyword to denote this element is a sextupole; all other parameters are explained in Table 15.

Table 15 Parameters of sextupoles in the lattice file.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Parameter** | **Units** |
| L | Length of the element | [m] |
| K | If L ≠ 0,  Gradient, .  If L= 0,  integrated field strength with unit [m-2] of this quadrupole component. | [m-3](If L ≠ 0)  [m-2](If L = 0) |
| FF1 | 1 or 0.  1 means taking into account the fringe field at the left edge.  0 means not taking into account the fringe field at the left edge |  |
| FF2 | 1 or 0.  1 means taking into account the fringe field at the right edge.  0 means not taking into account the fringe field at the right edge |  |
| Method | Order of symplectic integration method.  Value “1” means 1st order, “2” means 2nd order, and “4” means 4th order. | 1,2,4 |
| N | pieces of this element to be cut |  |

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>,

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

**N** =<# of kicks>, **method** = <method>;

Here “symbol” is the user defined element name; “multipole” is a keyword to denote this element is a multipole; all other parameters are explained in Table 16.

Table 16 Parameters of the multipoles in the lattice file.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Parameter** | **Units** |
| L | Length of the multipole | [m] |
| T | total bending angle | [degree] |
| T1 | Entrance angle | [degree] |
| T2 | Exit Angle | [degree] |
| Tilt | tilt angle of the quadrupole;  if ‘tilt’ is non-zero, then the quadrupole is a skew quadruple. | [degree] |
| HOM | Multipole field components of the element.  The format is n, <Bn>, <An>, etc. Here:  n is nth component of the multipole field;  Bn is nth component of the upright multipole field with the unit [m-(n/2-1)] ???;  An is nth component of the skew multipole field with the unit [m-(n-1)] ???. |  |
| N | pieces of this element to be cut |  |
| Method | Symplectic integration method.  1 is 1st order, 2 is 2nd order, 4 is 4th order. | 1,2,4 |

**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 4th order upright multipole 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 be updated.)

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

symbol: **Wiggler**, **L** = <length>,**BoBrhoV** = <B/Brho>,**BoBrhoH** = <B/Brho>,

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

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

**Method** = <method>;

Here “symbol” is the user defined element name; “wiggler” is a keyword to denote this element is a wiggler; all other parameters are explained in Table 17.

Table 17 The parameters of wigglers in a lattice file.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Parameter** | **Units** |
| L | Length of the wiggler | [m] |
| BoBrhoV | the normalized vertical field | [m-1] |
| BoBrhoH | the normalized horizontal field | [m-1] |
| Labmda | period length | [m] |
| kxV |  | [m] |
| kxH |  | [m] |
| phi |  | [degree] |
| harm |  |  |
| N | No of integration steps |  |
| Method | Symplectic integration method.  1 is 1st order, 2 is 2nd order, 4 is 4th order. |  |

**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 be updated……..)

To read field map from a file, use the format:

<symbol> : **Fieldmap**, **L** = <length [m]>, **N** = <no of integration steps>,

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

Here “symbol” is the user defined element name; “Fieldmap” is a keyword to denote this element is a Fieldmap; all other parameters are explained in Table 18.

Table 18 Parameters of field map in the lattice file.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Parameter** | **Units** |
| L | Length of the field map | [m] |
| file1 | field map file |  |
| N | No of integration steps |  |

‘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;

Here “symbol” is the user defined element name; “insertion” is a keyword to denote this element is an insertion device; all other parameters are explained in Table 19.

Table 19 parameters of the insertion devices in the lattice file.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Parameter** | **Units** |
| scaling1 | scaling factor for the 1st order field map |  |
| scaling2 | scaling factor for the 2nd order map |  |
| method | the order of symplectic interpolation method in the code, the value of 1 is linear interpolation, 3 is spline interpolation. |  |
| N | pieces of this element is cut when it is treated in the code |  |
| file1 | The 1st order of insertion device field are read from the files generated by RADIA.  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. |  |
| file1 | The 2nd order of insertion device field are read from the files generated by RADIA.  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/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;

Here “symbol” is the element name, “cavity” means that this element is a RF cavity; all the other parameters are explained in Table 20.

Table 20 Parameters of RF cacity in the lattice file.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Parameter** | **Units** |
| frequency | RF frequency | [Hz] |
| voltage | RF voltage | [Volt] |
| phase | synchrotron phase | [degree] |
| harnum | 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 values are 0.

### corrector

To define the corrector, use the command:

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

Here “symbol” is the element name, “corrector” means that this element is a corrector; all the other parameters are explained in

Table 21 parameters of correctors in the lattice file.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Parameter** | **Units** |
| Horizontal / vertical | “horizontal” means the element is a horizontal corrector; “vertical” means the element is a vertical corrector | - |
| method | Order of symplectic integration method.  Value ‘1’ means 1st order, ‘2’ means 2nd order, and ‘4’ means 4th order | 1,2,4 |

**Example:**

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

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

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

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

The parameters of “corrector” are optional, the default value for ‘method’ is **0.**

**Notice:**

**The “symbol” of correctors are special!!!!**

For a 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**;

Here “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 (To be updated)

BPM is a special marker in the lattice; the “symbol” name must be “BPM” (???). User can

define the BPM as:

**BPM : type;**

Normally Its type is defined as “Marker” type, but in order to include the misalignment error of BPM into the lattice, it must be defined as “Beam Position Monitor” type 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. It is defined as:

Symbol: **type;**

Normally Its type is defined as “Marker” type, but in order to include the misalignment error of girder into the lattice, it must be defined as “multipole” type, 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;

Here “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 this example 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.

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 an 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 needs to use the following command at the end of the lattice file:

**End;**

This command is mandatory.

## Multipole field error file

The multipole field errors 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 field error of the lattice elements.

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

### Define systematic multipole errors

To define the systematic multipole field 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 the multipole field error is measured)

n(order of multipole field error) Bn(nth B component of the

multipole field) An(n-th A component of the multipole field)

m, Bm, Am,......

* The "keywords" means one type of lattice elements or the family name; the 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 all the quadrupoles, user can NOT define the multipole errors by the type. There are two choice: one is to define the multipole errors for each quadrupole family; second is to define the field errors by quadrupole type, and then define the multipole errors on Q2 and Q7 families (the lattice with full quadrupoles) or QP2a, QP2b, QP7a and QP7b families (lattice with quadrupoles which are cut into two halves). This is due to that Q2/QP2a/QP2b and Q7/QP7a/QP7b are the quadrupoles which have lengths larger than other quadrupoles in the lattice, and 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 field errors on the lattice elements, user needs to define the seed of the random errors, and then follow the same rule as the ones to define systematic multipole error except replacing “sys” by “rms”. For example:

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 off value for the normal distribution function is 2 times of the RMS value. 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 each family, for example, the “Q1” family of the quadrupoles.

The systermatic misalignment error file works for the lattices with full or 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 needs 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 except replacing “sys” by “rms”. That is:

* 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 off value for the normal distribution function is 2 times of the RMS value.

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