European Organization for Nuclear Research
CERN BE/ABP

CERN/SL/94-56
Updated December 2019

SixTrack

Version 5.4.3

Single Particle Tracking Code Treating Transverse Motion with Synchrotron Oscillations in a Symplectic Manner

User's Reference Manual

F. Schmidt, A. Alekou, V.K. Berglyd Olsen, R. De Maria, M. D'Andrea, M. Fitterer, S. Kostoglou, A. Mereghetti, T. Persson, K. Sjobak, J.F. Wagner, and S.J. Wretborn,

Abstract

The aim of SixTrack is to track two nearby particles taking into account the full six–dimensional phase space including synchrotron oscillations in a symplectic manner. It allows to predict the long–term dynamic aperture which is defined as the border between regular and chaotic motion. This border can be found by studying the evolution of the distance in phase space of two initially nearby particles. Parameters of interest like nonlinear detuning and smear are determined via a post processing of the tracking data. An analysis of the first order resonances can be done and correction schemes for several of those resonances can be calculated. Moreover, there is the feature to calculate a one–turn map to very high order and the full six–dimensional case, using LBL differential algebra. This map allows a subsequent theoretical analysis like normal form procedures which are provided by É. Forest [1].

The linear elements are usually treated as thick elements in SixTrack. In that case there is at least one non–zero length element in the structure file which is not a drift element. If the accelerator, however, is modelled exclusively with drifts and kicks, SixTrack automatically uses the thin lens formalism according to G. Ripken [2]. A common header of output data and the format of these data has been found for MAD and SixTrack tracking data.

Geneva, Switzerland
December 19, 2019

Acknowledgement

I would like to thank my colleagues at DESY and CERN to help to find nasty bugs and for a thorough check of the program. I would like to thank Mikko Vaenttinen who helped to vectorise the program. He also did most of the typing of the manuscript. Moreover, I want to express my gratitude to F. Zimmermann who helped to finish the differential–algebra part in endless night sessions. Additions concerning Normal Forms have been contributed by M. Giovannozzi. J. Miles helped with the calculation of the 6D Courant–Snyder matrix and its use to transform the tracking data in the post–processing. W. Herr is thanked for providing a software package used for the orbit correction. L.H.A. Leunissen extracted and adapted the 6D beam–beam code of Hirata [20].

– F. Schmidt, for the version 3.x and 4.x manual

Contents

Chapter 1

Introduction

The Single Particle Tracking Code SixTrack is optimised to carry two particles [1] through an accelerator structure over a large number of turns. It is an offspring of RACETRACK [3] written by Albin Wrulich. The input structure is based on RACETRACK, but has been significantly enhanced in SixTrack 4 and 5.

The main features of SixTrack are:

1. Treatment of the full six-dimensional motion including synchrotron motion in a symplectic manner [5]. The energy can be ramped at the same time considering the relativistic change of the velocity [6].
2. Detection of the onset of chaotic motion and thereby the long term dynamic aperture by evaluating the Lyapunov exponent.
3. Post processing procedure allowing:
   - calculation of the Lyapunov exponent,
   - calculation of the average phase advance per turn,
   - FFT analysis,
   - resonance analysis,
   - calculation of the average, maximum and minimum values of the Courant-Snyder emittance and the invariants of linearly coupled motion,
   - calculation of smear, and
   - plotting using the CERN packages HBOOK, HPLOT and HIGZ [7, 8, 9]
4. Calculation of first order resonances and of correction schemes for the resonances [10].
5. Calculation of the one turn map using the differential algebra techniques. The original DA package by M.Berz [11] has been replaced by the package of LBL [1]. The Fortran code is transferred into a Map producing via the (slightly modified) "DAFOR" code [12].
6. The code is vectorised, with two particles, the number of amplitudes, the different relative momentum deviations $\Delta p/p_0$ in parallel [13].
7. Operational improvements:
   - free format input,
   - optimisation of the calculation of multipole kicks,
   - improved treatment of random errors,
   - each binary data file has a header describing the history of the run (Appendix C)

[1] Two particles are needed for the detection of chaotic behaviour.

1.1 Versions and Service

There are two versions: for element by element tracking there is a vector version, and there is a version to produce a one turn map using the LBL Differential Algebra package. In both cases the input structure file `fort.2` is used to determine if the thick or thin linear element mode has to be used.

To use the power of the Differential Algebra, for instance to calculate the 6D closed orbit in an elegant fashion, the tracking versions may also be equipped with a low order map facility to avoid the otherwise huge demand on memory.

It must be mentioned that in the linear thin lens version dipoles have to be treated in a special way. See section 5.1.3 for details.

To convert MAD-X files into SixTrack input, a special conversion command exists the MAD-X. See the MAD-X documentation for further details.

The following subroutines are taken from various packages:

Table 1.1: External Routines

Package	Routine	Purpose
HBOOK	HBOOK2, HDELET, HLIMIT, HTITLE	Graphic basics
HPLOT	HPLAX, HPLCAP, HPLEND, HPLINT	Graphic options
	HPLOPT, HPLSET, HPLSIZ, HPLSOF	
HIGZ	IGMETA, ISELNT, IPM, IPL	Graphic output

All versions can be downloaded from the web. The project webpage is found at http://sixtrack.web.cern.ch/, and primary source repository is located at https://github.com/SixTrack/SixTrack. Older versions can be found at http://cern.ch/Frank.Schmidt/Source.

In case of problems, please see the CERN SixTrack egroups "sixtrack-users" and "sixtrack-developers". If these are not accessible to you, you are welcome to contact the coordinators: Riccardo De Maria and Kyrre Sjobak, as well as the original developer Frank Schmidt. Our contact details are available from the CERN phonebook.

If you think you have found a defect in the program, please create a report on the issue tracker at https://github.com/SixTrack/SixTrack/issues. Note that for this to be useful, you need to describe what the program is doing, what you expected it to do, and an example which demonstrates the unwanted behaviour. Please also look through the issues that are already listed to see if it has already been reported. If so, you are welcome to add a comment to the issue, which may influence its priority and give additional and useful information to the developers.

The most up to date version of the documentation can always be found on the SixTrack website, and the latest features will be in the documentation in the source on GitHub. Additionally, various older documentation can be found at http://cern.ch/Frank.Schmidt/Documentation/doc.html.

1.2 Evolution of SixTrack

Following, is a short historical overview of how the versions of SixTrack have evolved.

- **Version 1** The first version has been an upgrade of RACETRACK [3] to include the full 6D formalism for long linear elements by G. Ripken [5].
- **Version 2** The DA package and the Normal Form techniques [11, 18] have been added to allow the production of high order one turn Taylor maps and their analysis. The 6D thin lens formalism [2] has also been included to speed up the tracking without appreciable deterioration of the accelerator model for very large Hadron colliders like the LHC.

- **Version 3** The beam–beam kick à la Bassetti and Erskine [19] has been included together with the 6D part by Hirata et al. [20]. Moreover, this 6D part has been upgraded to include the full 6D linear coupling [21]. Lastly, the LBL DA package has replaced the original one by Berz, and all operations needed to set up the accelerator structure are now performed with the help of Forest's LieLib package [1].
- **Version 5** Machine size and particle numbers are no longer defined by compiler flags but allocated dynamically based on the input files. Merges the Collimation version of SixTrack into the main code. Code-wide implementation of rounding of input float values from text, as well as output of float values to text in critical parts, provided by the CRLIBM library. This also includes a wrapper for consistent rounding of values from mathematical functions. Adds experimental support for HDF5 and ROOT output files. Extensive rewrite of many sections of the code to increase flexibility and modularity, as well as bug fixes to the physics. Includes a full rewrite of the input parsing and upgrade of the source to Fortran 2008. Support for 32, 64 and 128 bit floating point precision. Replaces the astuce preprocessor with the c preprocessor.

For a more detailed list of changes, see `CHANGELOG.md` in the repository.

1.3 SixTrack Input Structure

The SixTrack input is line oriented. Each line is treated as one string of input in which a certain sequence of numbers and character strings is expected to be found. The numbers and character strings must be separated by at least one blank space. Floating point numbers can be given in multiple formats, see below for further details. Comments can be added starting with the character "!". SixTrack 5 also supports in-line comments, older versions do not. Empty lines and lines starting with the comment character will be ignored and are not counted towards the line number requirement that apply to many of the input blocks. For beckwards compatibility, lines starting with a slash "/" in the first column will also be ignored by the program.

For detailed questions concerning rounding errors, calculation of the Lyapunov exponent and determination of the long term dynamic aperture, see [14].

1.3.1 Input Format

The input format used in SixTrack has been inherited from RACETRACK, but significantly extended in version 4 and 5 to shift to a keyword/value style format. Many of the older input blocks still use the old format, or support both RACETRACK and keyword/value types arguments.

The idea of the input format is to use a sequence of input blocks, each block with a specific keyword in the first line. The block is terminated by the keyword `NEXT` in the last line. The input data goes in the lines in between. The keyword `ENDE` ends the input sequence, and anything after this keyword is ignored. This system makes it easy to read input and allows easy change and addition of input blocks.

Values inside a block can be indented, but the opening and closing keywords cannot. In some blocks, more than 4 space indentation signify a line continuation of the previous line. It is therefore advisable to keep indentation at less than 5 spaces to avoid unpredictable results.

In blocks using a keyword/value format, the keywords are in general case sensitive, and should be provided in upper case. Values are as a rule not case sensitive, but there are some exceptions. Please consult the documentation of the relevant block for further details.

From version 5, SixTrack enforces the proper closing of input blocks, which in the past have been somewhat inconsistent. This does not apply to the first line, which contains the flag to determine which file to read the lattice from. If a `NEXT` keyword is added after this, an error is raised. Multiple occurrences of the same input block is, as a general rule, not allowed and will cause an error. However, there are a few exceptions. String input variables can be wrapped in either single or double quotes. SixTrack will also accept strings without quotes, provided they do not contain tabs, spaces or the comment character.

Input errors in `fort.3` will provide a descriptive error message, the line number and file in which the error was encountered, and a printout of the line where the error was found. Errors in other input files may be less descriptive.

A new `SETTINGS` block has been added in SixTrack 5 where a `DEBUG` flag can be added. When this flag is set, most values read from the main input file will be printed back with the value it was converted to internally in SixTrack, and if it was modified, what it was changed to.

In the following chapters, the input structure of SixTrack is discussed in detail. To facilitate the use of the program, a list of default values in Appendix A, the input and output files are described in Appendix B, and a description of the data structure of the binary data files in Appendix C.

1.3.2 Input Values

Integers must be entered as plain integers.

Floats can be entered as integers (converted during parsing) and standard Fortran floats. Examples `1, 1.0, 1.0e3, 1e3, 1.0d3, 1d3, 1.0q3, 1q3`.

Strings can be entered with both single and double quotes, and without quotes if the string contains no tabs, spaces or comment characters.

Flags accept the following values: `.true./.false.`, `true/false`, `yes/no`, `on/off`, and `1/0`. These are not case sensitive.

1.3.3 Command Line Arguments

SixTrack does not require any command line arguments, but can optionally take the file name for the main input file as well as the geometry file. The first file name encountered is taken as the input file, and the second is taken as the geometry file. See also Sections 3.1 and 3.2.

In addition, SixTrack can take the following command line arguments:

`--notrack`	SixTrack will run the initialisation of the job, but skip the entire tracking loop. This can be useful for checking initial simulation parameters without having to run the full job.
`-v, --version`	Echo program name and version as a single line, and exit.
`-V, --VERSION`	Echo program name, version, release date, and git hash on four lines, and exit.
`-nv, --numvers`	Echo the numerical version as an integer, and exit,

Chapter 2

Conventions

2.1 Tracking

The main particle tracking arrays used in SixTrack are listed in Table 2.2. Some of them are relative to the reference particle main values, which are listed in Table 2.1.

Table 2.1: An overview of the reference particle variables used in SixTrack.

Name	Variable	Unit	Description
m_0	`numc0`	[MeV]	Reference mass
E_0	`e0`	[MeV]	Reference energy
P_0	`e0f`	[MeV/c]	Reference momentum
β_0	`beta0`	[1]	Reference relativistic beta factor
γ_0	`gamma0`	[1]	Reference relativistic gamma factor

Table 2.2: An overview of the particle arrays used in SixTrack, and their definition.

Name	Variable	Unit	Definition	Description
x	`xv1(j)`	[mm]	–	Horizontal position
y	`xv2(j)`	[mm]	–	Vertical position
x'	`yv1(j)`	[1/1000]	$\frac{P_x}{P}$	Approximate horizontal angle
y'	`yv2(j)`	[1/1000]	$\frac{P_y}{P}$	Approximate vertical angle
σ	`sigmv(j)`	[mm]	$s - \beta_0 ct$	Longitudinal offset
p_σ	n/a	[1]	$\frac{E-E_0}{\beta_0 P_0 c}$	Canonical conjugate of σ
δ	`dpsv(j)`	[1]	$\frac{P-P_0}{P_0}$	Canonical conjugate of σ
r_v	`rvv(j)`	[1]	$\frac{\beta_0}{\beta}$	Velocity ratio
r_p	`oidpsv(j)`	[1]	$\frac{P_0}{P}$	Momentum ratio
ζ	n/a	[mm]	σ / r_v	Longitudinal offset conjugate with δ
m	`nucm(j)`	[MeV/c^2]	–	Mass
m/c	`mtc(j)`	[1]	$\frac{q}{q_0}\frac{m_0}{m}$	Mass-to-charge ratio
P	`ejfv(j)`	[MeV/c]	–	Momentum
E	`ejv(j)`	[MeV]	–	Energy

2.1.1 Normalisation Matrix

The normalisation matrix, referred to in this manual as the *T-matrix* and in the source code with the variable `tas`, is a 6×6 matrix calculated from the eigenvectors of the one-turn map. The T-matrix is used to convert normalised coordinates to physical coordinates, and its inverse converts physical to normaliased.

The T-matrix and its inverse is printed to the tracking files (see Appendix B) and to the particle state files dumped from the settings block (see Section 3.4).

It is also used for writing normalised particle dumps (see Section 10.1) and to read normalised input beam distributions (see Section 4.7).

Chapter 3

General Input

3.1 Main Input Files

SixTrack requires a main input file, which by default must be named `fort.3`. Alternatively, if a different file name is desired, it can be given to the SixTrack executable as the first command line argument. If a geometry file is requested in the main input file (see Section 3.2), it can be given as the second command line argument. If none is provided, SixTrack will look for a file named `fort.2`. These files will be referred to as `fort.3` and `fort.2` in the rest of this document.

Note that you can always add the global `DEBUG` flag to the `SETTINGS` block to enable echoing back most of the input parameters set in the `fort.3` file. The flag is described in Section 3.4. This can not only verify that SixTrack understood and received the values correctly, but it also often echoes back other parameters computed from the input values. The echoed back lines will start with "`INPUT> DEBUG`", so it is sometimes easier to grep for these lines in the output. They contain the name of the block where they were parsed, and their line number within the block if available or relevant.

3.2 Program Version

The *Program Version* input block determines if all of the input will be in the input file `fort.3`, or if the geometry part of the machine (see 5) will be in a separate file: `fort.2`. The latter option is useful if tracking parameters are changed, but the geometry part of the input is left as it is. The geometry part can be produced directly from a MAD-X input file (see 3.7). Note that this line should not have a `NEXT` keyword after it, and must always be the first line of the file.

Keyword	`FREE` or `GEOM`
Data lines	None
Format	`keyword comment title`

Format Description

`keyword`	The first four characters of the first line of the `fort.3` input file are reserved for the keyword. `FREE` for free format input with all input in `fort.3`, and `GEOM` if the geometry part is in file `fort.2`.
`comment`	Following the first four characters, 8 characters are reserved for comments
`title`	The next 60 characters are interpreted as the title printed at the top of the output file `fort.6`.

3.3 Print Selection

The `PRIN` flag is deprecated, and replaced by the `PRINT` flag in the `SETTINGS` block.

3.4 Settings

The *Print Selection* input block available in earlier version has been replaced with the `SETTINGS` block. This was done to allow for more options related to what output SixTrack produces in `fort.6`. The `PRIN` flag is available as one of several options in this block. However, for backwards compatibility, the `PRIN` flag is still accepted.

Keyword `SETT`
Data lines Variable

Format Description

`PRINT` This causes the printing of the input data to the output file `fort.6`.

`DEBUG` A global debug flag that causes the majority of the blocks to echo back the value read from the input file after parsing. It may also print out secondary values set based on input values read, or modification made to input values based on other dependencies and criteria.

`QUIET` Followed by an integer specifying how "quiet" the output should be. A higher value causes less information to be printed back out. If the keyword is not present, the default value is 0, which means it is disabled. If it is present, but the integer value is omitted, it is set to be 1. This flag does not interfere with the `PRINT` flag.

`PRINT_DCUM` This will cause the calculated s-coordinate of each structure element to be printed to the file `machine_length.dat`.

`PARTSUMMARY` Enable or disable the printing of a particle summary after tracking. The flag takes an optional parameter to explicitly state whether it is ON or OFF. If omitted, it is assumed the user requests it to be ON. If the flag is omitted entirely, the default behaviour is determined by the particle count. If SixTrack is running with 64 particles or less, it is ON by default, otherwise OFF.

`WRITEFORT12` Enable or disable the writing of `fort.12` after tracking. The flag takes an optional parameter to explicitly state whether it is ON or OFF. If omitted, it is assumed the user requests it to be ON. If the flag is omitted entirely, the default behaviour is determined by the particle count. If SixTrack is running with 64 particles or less, it is ON by default, otherwise OFF.

`INITIALSTATE` Followed by either "binary" or "text". This will write a file before tracking containing the initial state of all particles to either a binary or a text file. Adding "ions" as a second keyword will also dump the additional ion columns (see Section 4.5). The file header also contains the settings of the reference particle, the 4D and 6D closed orbit, the tunes, and the TA matrix. (Note that the $\mathrm{d}p/p_0$ values are not scaled by a factor 1000 in this file.)

`FINALSTATE` Followed by either "binary" or "text". This will write a file after tracking containing the final state of all particles, including those lost during tracking, to either a binary or a text file. Adding "ions" as a second keyword will also dump the additional ion columns (see Section 4.5). The file header also contains the settings of the reference particle, the 4D and 6D closed orbit, and the TA matrix. (Note that the $\mathrm{d}p/p_0$ values are not scaled by a factor 1000 in this file.)

3.5 Comment Line

An additional comment can be specified with the *Comment* block. The comment will be written to the binary data files (Appendix C), and will appear in the post processing output as well.

Keyword COMM
Data lines 1
Format A string of up to 80 characters.

3.6 Iteration Errors

For the processing procedures, the number of iterations and the precision to which the processing is to be performed are chosen with the *Iteration Errors* input block. If the input block is left out, default values will be used.

Keyword ITER
Data lines 1 to 4
Format Each data line holds three values as in table 3.1, except for the fourth line where the horizontal and vertical aperture limits can be additionally specified. This has been added to avoid artificial crashes for special machines.

Table 3.1: Iteration Errors

Variable	Type	Default	Description
Data Line 1			
ITCO	int	50	Number of Iterations for closed orbit calculation.
DMA	dbl	1e-12	Demanded Precision of closed orbit displacements.
DMAP	dbl	1e-15	Demanded Precision of derivative of closed orbit displacements.
Data Line 2			
ITQV	int	10	Number of Iterations for Q Adjustment.
DKQ	dbl	1e-10	Variations of quadrupole strengths.
DQQ	dbl	1e-10	Demanded Precision of tunes.
Data Line 3			
ITCRO	int	10	Number of Iterations for chromaticity correction.
DSMO	dbl	1e-10	Variations of sextupole strengths.
DECH	dbl	1e-10	Demanded Precision of chromaticity correction.
Data Line 4			
DEO	dbl	1e-9	Variations of momentum spread for chromaticity calculation.
DED	dbl	1e-9	Variations of momentum spread for evaluation of dispersion.
DSI	dbl	1e-9	Demanded Precision of desired orbit r.m.s. value; compensation of resonance width.
APER(1)	dbl	1000 [mm]	Demanded Precision of horizontal aperture limit.
APER(2)	dbl	1000 [mm]	Demanded Precision of vertical aperture limit.

3.7 MAD-X to SixTrack Conversion

A converter has been developed [15], which is directly linked to MAD-X. It produces the geometry file `fort.2`; an appendix to the parameter file `fort.3`, which defines which of the multipole errors are switched on; the error file `fort.16`, and the file `fort.8` which holds the transverse misalignments and the tilt of the non-linear kick elements. It also produce a file `fort.34` with linear lattice functions, phase advances and multipole strengths needed for resonance calculations for the program *SODD* [22].

In addition, the flag `aperture` will produce an aperture limitations file. The flag `multicol` will produce an alternative `fort.2` file with more information on the machine structure (see Section 5.2.1.2).

Chapter 4

Initial Conditions for Tracking

For the study of non-linear systems, the choice of initial conditions is of crucial importance. The input structure for the initial conditions was therefore organised in such a way as to allow for maximum flexibility. SixTrack is optimised to reach the largest possible number of turns. In order to derive the Lyapunov exponent, and thereby to distinguish between regular and chaotic motion, the particle has a close by companion particle. Moreover, experience has shown that varying only the amplitude while keeping the phases constant is sufficient to understand the non-linear dynamics, as a subsequent detailed post-processing allows to find the dependence of the parameter of interest on these phases.

A number of features have over time been deprecated or replaced by other modules in SixTrack. Therefore there are a number of parameters that are no longer in use, but nevertheless have to be set to dummy values. The Simulation (`SIMU`) and Distribution (`DIST`) blocks are intended to replace all of the blocks in this chapter. These blocks take keyword/value sets instead of blocks of numbers, and are therefore easier to maintain. The `SIMU` block (Section 4.1) can currently be used to replace the `TRAC`, `INIT`, and `HION` blocks (Section 4.2), but note that the implementation may still have bugs, and is therefore considered experimental.

4.1 Simulation Parameters

Note: This input block is experimental. It provides an alternative interface to the most used parameters of the `TRAC`, `INIT`, and `HION` blocks, and is intended to be used in combination with the `DIST` block.

The Simulation block (`SIMU`) is intended to take the main simulation parameters, and replaces the `TRAC`, `INTI`, and `HION` blocks. If the `SIMU` block is present in `fort.3`, these blocks cannot be present.

If the reference particle mass is set in the `SIMU` block, the value provided in the `SYNC` block is ignored.

Keyword	`SIMU`
Data lines	Variable
Format	Keyword/value format. See Table 4.1.

Some settings provided by the `TRAC` and `INIT` blocks are not supported by the `SIMU` block. These are listed below. The option to add closed orbit to generated particles is only supported for the amplitude scan, which is not supported by the `SIMU` block. If you need this feature, please use the old input format.

`TRAC`	`ntwin`	Fixed to a value of 2.
`TRAC`	`idy(1), idy(2)`	Fixed to a value of 1.
`TRAC`	`idfor`	Fixed to a value of 1.
`TRAC`	`amp(1), amp0`	Fixed to a value of 0.

Table 4.1: Available arguments in the SIMU block.

Keyword	Argument(s)	Default
Particles and Turns		
PARTICLES	n_particles(int)	0
The number of particles to be tracked. The value must be an even number. This is due to several parts of SixTrack dealing with particles as pairs.		
TURNS	forward(int) [backward(int)]	0 0
The number of turns in the forward, and optionally, backward direction.		
CRPOINT	interval(int)	1000
How often to write checkpoint files. This parameter is ignored if SixTrack was not built with checkpoint/restart functionality. Checkpoint files are always written on turn 1, then with the interval specified here, and then a last time at the end of tracking.		
Reference Particle		
REF_ENERGY	energy(float,MeV)	0.0
The reference particle energy in MeV.		
REF_PARTICLE	mass(float,MeV) [charge A Z]	938.271998 1 1 1
REF_PARTICLE	name [charge A Z]	proton 1 1 1
The reference mass can either be provided as a value in MeV, or as a named particle. Currently this can only be set to "proton". This value defaults to the proton mass set in the SixTrack physical constants module in source/constants.f90. Optionally, the reference particle charge and atomic mass (A), and atomic number (Z) can be set. If A is set, Z must also be set. If only charge is set, Z is set to the same value. The default values are all 1 (proton).		
PDG_YEAR	year(int)	2002
This can be used to set the PDG year to use for the mass if a name is provided in REF_PARTICLE. The default value is the 2002 proton mass, the other value currently supported is 2018. More will be added in the future. Note that this value affects how the proton radius is calculated as it uses the PDG year to select the relevant constant for calculating this. Even when setting a reference particle mass in MeV, the PDG year is used for this.		
Lattice and Optics		
LATTICE	thin|thick 4D|6D	thin 4D
The first argument must be either thick or thin, and this must match the content of the geometry file. The second argument must be either 4D or 6D. These arguments are not case sensitive. When 6D tracking is requested, closed orbit and optical functions at the starting point are calculated using the differential algebra package.		
OPTICS	first(int) last(int)	1 nblz
Start and stop structure element index for optics calculation. If set to 0 or omitted, the optics calculation defaults to the full machine.		
Closed Orbit		
6D_CLORB	on|off	off
Compute the 6D closed orbit. If the simulation is running 4D, this option is ignored.		
INIT_CLORB	on|off	off
INIT_CLORB	x xp y yp [sigma dpsv] (float)	6 * 0.0
Initialise closed orbit. This keyword can be called either with a flag, in which case it turns on or off the reading of a closed orbit suggestion from file fort.33, or it can provide 4 or 6 values for the closed orbit suggestion. If omitted, the values are initialised to zero, and the 4D closed orbit is used to seed the first four values of the 6D closed orbit. These settings are ignored when running in 4D.		

(The table continues on the next page)

Keyword	Argument(s)	Default
Particle and Track Files		
READ_FORT13	on|off	off
Read the particle distribution from file `fort.13`. This file is not intended for reading an initial distribution, but for continuing tracking from a previous simulation from a `fort.12` file. Note that if the file is used as an input file for the initial distribution, the closed orbit is not added, even if requested with the `ADD_CLORB` flag.		
WRITE_FORT12	interval(int)	10000
How often, in terms of turns, to write the particle distribution to file `fort.12`. This file can be renamed to `fort.13` and used as an input file for continued tracking.		
WRITE_TRACKS	interval(int) [rewind(flag)]	nturn+1 on
How often, in terms of turns, to write to the tracking file `singletrackfile.dat` (see Appendix B). The optional `rewind` flag specifies whether or not to rewind the tracking files on each write.		
Various Flags and Options		
EXACT	on|off	off
Switch to enable exact solution of the equation of motion into tracking and 6D (no 4D) optics calculations. `off:` $x' \simeq \frac{P_x}{P_0(1+\delta)}, \quad y' \simeq \frac{P_y}{P_0(1+\delta)};$ `on:` $x' \simeq \frac{P_x}{P_0\sqrt{(1+\delta)^2 - P_x^2 - P_y^2}}, \quad y' \simeq \frac{P_y}{P_0\sqrt{(1+\delta)^2 - P_x^2 - P_y^2}}.$		
CURVEFF	on|off	off
Enable or disable the effect of the curvature in a combined function magnet (bending + quadrupole). Note that the weak focusing effect is always included.		

4.2 Tracking Parameters

All tracking parameters are defined with this input block. The initial coordinates are generally also set here. A fine tuning of the initial condition is done with *Initial Coordinates* block (4.3), and the parameters for the synchrotron oscillation are given in block (4.4).

Keyword TRAC
Data lines 3
Format
Line 1: `numl numlr napx amp(1) amp0 ird imc`
`niu(1) niu(2) numlcp numlmax`
Line 2: `idy(1) idy(2) idfor irew iclo6`
Line 3: `nde(1) nde(2) nwr(1) nwr(2) nwr(3) nwr(4)`
`ntwin ibidu iexact curveff`

Format Description

`numl`	integer	Number of turns in the forward direction.
`numlr`	integer	Number of turns in the backward direction.
`napx`	integer	Number of amplitude variations (i.e. particle pairs).

`amp(1),amp0`	floats	Start and end amplitude (any sign) in the horizontal phase space plane for the amplitude variations. The vertical amplitude is calculated using the ratio between the horizontal and vertical emittance set in the *Initial Coordinates* block (4.3), where the initial phase in phase space are also set. Additional information can be found in the *Remarks*.
`imc`	integer	Number of variations of the relative momentum deviation has been removed. This value must be 1.
`niu(1),niu(2)`	integer	Start and stop structure element index for optics calculation. If 0, defaults to the full machine.
`numlcp`	integer	Checkpoint/restart version: How often to write checkpointing files.
`numlmax`	integer	No longer in use.
`idz(1),idz(2)`	integers	A tracking where one of the transversal motion planes shall be ignored is only possible when all coupling terms are switched off. The part of the coupling that is due to closed orbit and other effects can be turned off with these switches. `idz(1), idz(2) = 1`: coupling on. `idz(1), idz(2) = 0`: coupling to the horizontal and vertical motion plane respectively switched off.
`idfor`	integer	Usually the closed orbit is added to the initial coordinates. This can be turned off using `idfor`, for instance when a run is to be prolonged. `idfor = 0`: closed orbit added. `idfor = 1`: initial coordinates unchanged. `idfor = 2`: prolongation of a run, taken the initial coordinates from `fort.13`.
`irew`	integer	To reduce the amount of tracking data after each amplitude and relative momentum deviation iteration $\Delta p/p_0$ the binary track file `singletrackfile.dat` (see Appendix B) are rewound. This is always done when the post-processing is activated (9.12). For certain applications it may be useful to store all data. The switch `irew` allows for that. `irew = 0`: track file rewound. `irew = 1`: all data on track file.
`iclo6`	integer	This switch allows to calculate the 6D closed orbit and optical functions at the starting point, using the differential algebra package. It is active in all versions that link to the Differential Algebra package. Note that `iclo6 > 0` is mandatory for 6D simulations, and that `iclo6 = 0` is mandatory for 4D simulations. `iclo6 = 0`: switched off. `iclo6 = 1`: calculated. `iclo6 = 2`: calculated and added to the initial coordinates (4.3). `iclo6 = 5 or 6`: like for 1 and 2, but in addition a guess closed orbit is read (in free format) from file `fort.33`.
`nde(1)`	integer	Number of turns at flat bottom, useful for energy ramping.
`nde(2)`	integer	Number of turns for the energy ramping. `numl-nde(2)` gives the number of turns on the flat top. For constant energy with $nde(1) = nde(2) = 0$ the particles are considered to be on the flat top.
`nwr(1)`	integer	Every `nwr(1)`'th turn the coordinates will be written to track file in the flat bottom part of the tracking.

`nwr(2)`	integer	Every `nwr(2)`'th turn the coordinates in the ramping region will be written to track file.
`nwr(3)`	integer	Every `nwr(3)`'th turn at the flat top a write out of the coordinates to track file will occur. For constant energy this number controls the amount of data in track file, as the particles are considered on the flat top.
`nwr(4)`	integer	In cases of very long runs it is sometimes useful to save all coordinates for a prolongation of a run after a possible crash of the computer. Every `nwr(4)`'th turn the coordinates are written to unit 6.
`ntwin`	integer	For the analysis of the Lyapunov exponent it is usually sufficient to store the calculated distance of phase space together with the coordinate of the first particle (`ntwin` set to one). You may want to improve the 6D calculation of the distance in phase space with `sigcor, dpscor` (see 4.3) when the 6D closed orbit is not calculated with `iclo6` $\neq 2$. If storage space is no problem, one can store the coordinates of both particles (`ntwin` set to two). The distance in phase space is then calculated in the post-processing procedure (see 9.12). This also allows a subsequent refined Lyapunov analysis using differential algebra and Lie algebra techniques ([29]).
`ibidu`	integer	No longer in use. Value ignored.
`iexact`	integer	Switch to enable exact solution of the equation of motion into tracking and 6D (no 4D) optics calculations. `iexact = 0`: approximated equation $$\text{e.g.} \quad x' \simeq \frac{P_x}{P_0(1+\delta)}, \quad y' \simeq \frac{P_y}{P_0(1+\delta)};$$ `iexact = 1`: exact equation $$\text{e.g.} \quad x' \simeq \frac{P_x}{P_0\sqrt{(1+\delta)^2 - P_x^2 - P_y^2}}, \quad y' \simeq \frac{P_y}{P_0\sqrt{(1+\delta)^2 - P_x^2 - P_y^2}}.$$
`curveff`	integer	`curveff = 0`: the effect of the curvature in a combined function is neglected. Note that the weak focusing effect is always included. `curveff = 1`: switch to enable the curvature effect in a combined function magnet (bending + quadrupole).

Remarks

1. This input data block is usually combined with the *Initial Coordinates* input block (4.3) to allow a flexible choice of the initial coordinates for the tracking.
2. For a prolongation of a run the following parameters have to be set:
   (a) in this input block: `idfor = 1`
   (b) in the *Initial coordinates* input block:
   - `itra = 0`
   - take the end coordinates of the previous run as the initial coordinates (including all digits) for the new run.
3. A feature is installed for a prolongation of a run by using `idfor = 2` and reading the initial data from file `fort.13`. The end coordinates are now written to `fort.12` after each run. Intermediate

coordinates are also written to `fort.12` in case the turn number `nwr(4)` is exceeded in the run. The user takes responsibility to transfer the required data from `fort.12` to `fort.13` if a prolongation is requested. This feature can be used to effectively read in a custom-made beam distribution. The format of the file is one line per pair of particles; the meaning of the columns is exactly that of the *Initial Coordinates* input block (see Sec. 4.3 and Tab. 4.4).

4. As of version 5.2 the particle momentum offset from `fort.13` is re-calculated from the particle energy to ensure these are consistent. The particle momentun offset in `fort.13` is therefore ignored.

5. Some illogical combinations of parameters have been suppressed.

6. The initial coordinates are calculated using a proper linear 6D transformation: `amp(1)` is still the maximum horizontal starting amplitude (excluding the dispersion contribution) from which the emittance of mode 1 e_I is derived, `rat` (see 4.3) is the ratio of e_{II}/e_I of the emittances of the two modes. The momentum deviation $\frac{\Delta p}{p_{0,1}}$ is used to define a longitudinal amplitude. The 6 normalized coordinates read:

   (a) horizontal:

   $$\left[\sqrt{e_I} = \frac{\text{amp}(1)}{\sqrt{\beta_{xI}} + \sqrt{|\text{rat}| \times \beta_{xII}}}, \quad 0.0\right]$$

   (b) vertical:

   $$[sign(\text{rat}) \times \sqrt{e_{II}} \text{ with } e_{II} = |\text{rat}| \times e_I, \quad 0.0]$$

   (c) longitudinal:

   $$\left[0.0, \quad \frac{\Delta p}{p_{0,1}} \times \sqrt{\beta_{sIII}}\right]$$

   and are then transformed with the 6D linear transformation into real space. Note that results may differ from those of older versions.

7. The amplitude scan is performed from `amp(1)` to `amp0` in steps of delta = (amp0−amp(1))/(*napx*−1). For the intermediate amplitudes, `delta` is added up for each step, however the last amplitude is guaranteed to be fixed to the given value. This enables "control calculations" by setting the first amplitude of one simulation equal to the last amplitude of another simulation, and unless there are calculation errors, they shall produce exactly the same results.

8. Note that if `iclo6 = 2` and `idfor = 0` in the input file, then `idfor` is internally set to 1, as is seen in some outputs. This does not mean that the closed orbit is not added; the setting of `iclo6 = 2` simply takes precedence.

4.3 Initial Coordinates

The *Initial Coordinates* input block is meant to manipulate how the initial coordinates are organise, which are generally set in the tracking parameter block (4.2). Number of particles, initial phase, ratio of the horizontal and vertical emittances and increments of 2 × 6 coordinates of the two particles, the reference energy and the starting energy for the two particles.

Keyword	`INIT`
Data lines	16
Format	Line 1: `itra chi0 chid rat iver` Lines 2 to 16: 15 initial coordinates as listed in Table 4.4

Format Description

`itra`	integer	Number of particles: `itra = 0`: Amplitude values of tracking parameter block (4.2) are ignored and coordinates of data line 2–16 are taken. `itra` is set internally to 2 for tracking with two particles. This is necessary in case a run is to be prolonged. `itra = 1`: Tracking of one particle, twin particle ignored. `itra = 2`: Tracking the two twin particles.
`chi0`	float	Starting phase of the initial coordinate in the horizontal and vertical phase space projections.
`chid`	float	Phase difference between first and second particles.
`rat`	float	Denotes the emittance ratio (e_{II}/e_I) of horizontal and vertical motion. For further information see the *Remarks* of the `TRAC` input block in Section 4.2.
`iver`	integer	In tracking with coupling it is sometimes desired to start with zero vertical amplitude which can be painful if the emittance ratio `rat` is used to achieve it. For this purpose the switch `iver` has been introduced: `iver = 0`: Vertical coordinates unchanged. `iver = 1`: Vertical coordinates set to zero.

Table 4.4: Initial Coordinates of the 2 Particles

Line	Contents
2	x_1 [mm] coordinate of particle 1
3	x'_1 [mrad] coordinate of particle 1
4	y_1 [mm] coordinate of particle 1
5	y'_1 [mrad] coordinate of particle 1
6	path length difference 1 ($\sigma_1 = s - v_0 \times t$) [mm] of particle 1
7	$\Delta p/p_{0,1}$ of particle 1
8	x_2 [mm] coordinate of particle 2
9	x'_2 [mrad] coordinate of particle 2
10	y_2 [mm] coordinate of particle 2
11	y'_2 [mrad] coordinate of particle 2
12	path length difference ($\sigma_2 = s - v_0 \times t$) [mm] of particle 2
13	$\Delta p/p_{0,2}$ of particle 2
14	energy [MeV] of the reference particle
15	energy [MeV] of particle 1
16	energy [MeV] of particle 2

Remarks

- These 15 coordinates are taken as the initial coordinates if `itra` is set to zero (see above). If `itra` is 1 or 2 these coordinates are added to the initial coordinates generally defined in the tracking parameter block (4.2). This procedure seems complicated but it allows freely to define the initial difference between the two twin particles. It also allows in case a tracking run should be prolonged to continue with precisely the same coordinates. This is important as small difference may lead to largely different results.

- The reference particle is the particle in the centre of the bucket which performs no synchrotron oscillations.
- The energy of the first and second particles is given explicitly, again to make possible a continuation that leads precisely to the same results as if the run would not have been interrupted.
- There is a refined way of prolonging a run, see the *Tracking Parameters* input block (4.2).

4.4 Synchrotron Oscillation

The parameters needed for treating the synchrotron oscillation in a symplectic manner are given in the *Synchrotron Oscillation* input block.

Keyword	`SYNC`
Data lines	2
Format	Line 1: `harm alc u0 phag tlen pma ition dppoff`
	Line 2: `dpscor sigcor`

Format Description

`harm`	integer	Harmonic number.
`alc`	float	Momentum compaction factor, used here only to calculate the linear synchrotron tune Q_S.
`u0`	float	Circumference voltage in [MV].
`phag`	float	Acceleration phase in degrees.
`tlen`	float	Length of the accelerator in meters.
`pma`	float	Rest mass of the particle in MeV/c^2.
`ition`	integer	Transition energy switch: `ition = 0`: for no synchrotron oscillation (energy ramping still possible). `ition = 1`: for above transition energy. `ition = -1`: for below transition energy.
`dppoff`	float	Offset Relative Momentum Deviation $\Delta p/p_0$: a fixpoint with respect to synchrotron oscillations. It becomes active when the 6D closed orbit is calculated (see item `iclo6` in section 4.2).
`dpscor,sigcor`	floats	Scaling factor for relative momentum deviation $\Delta p/p_0$ and the path length difference ($\sigma = s - v_0 \times t$) respectively. They can be used to improve the calculation of the 6D distance in phase space, but is only used when `ntwin = 1` in the tracking parameter input block (4.2). Please set to 1 when the 6D closed is calculated.

Note: The value of `tlen` is also calculated internally by SixTrack (in `dcum`), and a warning is issued if the given value is different from the calculated value.

4.5 Tracking with Ions

The default tracking in SixTrack is for protons. In case tracking of ions is wanted the following input block should be used. The `HION` block only specifies the reference particle. By default, all particles

are initialised to the same values, but if multiple ion species are needed, these can be provided by an input file in the DIST block.

Keyword	HION
Data lines	1
Format	Line 1: A Z m_a Q

Format Description

A	integer	Total number of nucleons (atomic mass number).
Z	integer	Total number of protons.
m_a	float	Mass of the ion [GeV/c^2].
Q	integer	Electrical charge.

4.6 Random Numbers

The RAND block configures and initialises the pseudo-random number generator used by some parts of SixTrack. The block allows the user to set a master seed that generates a reproducible sequence of numbers, or alternatively seed the generator with a clock seed by providing the word TIME in place of a seed number. The SEED keyword is required, and if it is a number, it must be larger than 0.

The main random number generator in this module is RANLUX [52]. The RANECU [53, 54] generator used by the magnet error code is also available. It is possible to override the default generator for the block with the OVERRIDE keyword. This will force the initialisation of all random number series to set the generator provided.

The "Luxury level" of the RANLUX generator is by default set to 3. This can be overridden by the LUXLEVEL keyword.

Some modules in SixTrack uses its own seeds and initialisation of random numbers. Please refer to the sections for the individual blocks for which feature requires this module. The original magnet error block is one such module (see Secyion 8.1).

Keyword	RAND
Data lines	Variable
Format	Keyword/value

Example

```
RANDOM NUMBERS
  SEED 42
  LUXLEVEL 3
  OVERRIDE ranlux
NEXT
```

4.7 Initial Distribution

The DIST block adds the ability to read a beam distribution from file, or generate it internally in SixTrack. The file format is very flexible and can be specified column-wise with the FORMAT keyword to support many file layouts. It is also possible to specify the unit of the data in the input file, within a limited range. If no format is specified, the DIST block falls back to the fixed 14 column format read by the original simple DIST block prior to version 5.3.1 (see Table 4.10).

Keyword DIST
Data lines Variable
Format Keyword/value

Format Description

There are several approaches available for initiating the beam distribution. It can be read in its entirety from a file, selecting a set of many available conventions for describing the particle coordinates, ion values and meta data. For a small number of particles, the coordinates can be set directly in the DIST block as well.

The DIST block is linked to the external DISTlib library for beam distributions, and the filled particle coordinates can be passed on to this library for further processing, like applying the T-matrix (see Section 2.1.1). In the future, more features will be added to this library, and made available through the DIST block interface.

Table 4.7 lists the currently available keywords of the DIST block. Table 4.8 lists all the column formats the block supports. Table 4.9 lists all the available fill methods for populating the particle arrays without having to read from file.

For an overview of the definitions used for the particle tracking variables in SixTrack, see Section 2.1.

Default Behaviour

Since there are multiple ways to set the particle coordinates, a few default behaviours and precedences have been coded into the parsing.

- All particle coordinates are initially set to 0, with the exception of particle energy, which defaults to the reference energy. That is, the particle δ momentum is 0.
- The particle mass and ion parameters also default to those of the reference particle.
- The particle ID, if not provided, is set as a range from 1 to the number of particles as specified in the TRAC or SIMU block.
- The particle parent ID is set to equal that of its ID. This is the correct way to indicate that a particle is a primary particle.

It is only possible to set each particle coordinate using one method. If conflicting methods are selected in the FORMAT keyword and in FILL methods, an error will be raised. Since SixTrack uses multiple arrays for different values related to the particle energy, these are calculated after initialisation from the input format chosen.

The default normalisation method is to use the internal T-matrix in SixTrack (see Section 2.1.1). Alternatively, Twiss and dispersion can be set in the block, or a new T-matrix provided. These will then take precedence over the internal matrix.

All emittances default to 0, so if these are not set, the normalisation will also return arrays of zeros.

Table 4.7: Available keyword/value sets in the DIST block.

Keyword	Argument(s)	Default
Input, Output and Format		
FORMAT	[list_of_columns]	OLD_DIST
(The table continues on the next page)		

Keyword	Argument(s)	Default
A list of column formats for the input. The available column values are listed in Table 4.8. This format is applied to either the input file or to particles specified directly in the DIST block. The number format columns must match the file columns or particle entries. If no format is specified, the parser assumes it will receive a 14 column file matching the format described in Table 4.10.		
READ	filename(char) [use_distlib(flag)]	
The filename of the file to read. An optional logical flag sets whether the filename is passed on to the external DISTlib, in which case the file must conform to the DISTlib file format. This is not covered here. If the file contains more particles than requested in the TRAC or SIMU block, the remaining particles will be ignored. If the file contains less particles, an error will be raised.		
PARTICLE	[list_of_values]	
A list of values to be parsed as a particle. This requires a format to be specified. It provides the option to add particles to the simulation without having to use the INIT block or a distribution file. Although not intended for initialising a large number of particles, there is no limit on how many times this keyword can be used.		
ECHO	[filename]	echo_distribution.dat
Echos the distribution back to a file. The format of the file is described in Table 4.11. This keyword is kept for legacy support, but a much more detailed file is written by the INITIALSTATE keyword in the SETTINGS block.		
Beam Parameters		
EMITTANCE	emit1[mm mrad] emit2[mm mrad]	0.0 0.0
The transverse beam emittance values in units of mm mrad.		
LONGEMIT	emit3 unit[eVs|um]	0.0
Longitudinal emittance and its unit. The emittance can either be provided in μm or in eVs.		
TWISS	betaX[m] alphaX[1] betaY[m] alphaY[1]	1.0 0.0 1.0 0.0
The horizontal and vertical twiss parameters.		
DISPERSION	dx dpx dy dpy	0.0 0.0 0.0 0.0
Beam dispersion		
TMATRIX	val1 val2 val3 val4 val5 val6	
Specify the 6×6 normalisation matrix in its entirety. The keyword needs to be repeated 6 times, once for each row.		
Internal Generator		
FILL	[list_of_parameters]	
The different columns can also be filled by a set of fill functions controlled with this keyword. The settings provided by this keyword are applied after the file is read or the arrays are populated by the PARTICLES keyword. The FILL feature can therefore be used to overwrite the data read from the file. The various fill methods available are listed in Table 4.9. If a fill method using random numbers is used, note that the random number generator also must be initialised with the RAND block. See Section 4.6.		

4.7.1 Column Formats

The FORMAT keyword allows the user to specify their own file format by providing a list of columns it contains. The available formats and how they are converted to internal SixTrack particle coordinates are listed in Table 4.8.

Each column can optionally take a unit in square brackets, appended to the column name itself with no space in between. The available units are also listed in the table for the column formats that

support units. If no unit in square brackets is provided, the parser defaults to the internal SixTrack units which are mm, mrad and MeV.

For units of energy $c = 1$ such that for instance MeV, MeV/c and MeV/c^2 are equivalent. The parser accepts the following notation: `MeV`, `MeV/c`, `MeV/c^2`, and `MeV/c**2`. Units are not case sensitive.

For units of length, the parser accepts the character `u` as an alternative to μ.

Multi-Column Formats

To avoid the need for specifiying common combinations of columns, a set of multi-column keywords are also available. They are translated directly into a group of columns in a pre-defined order. using these keywords does not prevent the user from adding more columns to extend the format.

Note, however, that conflicting columns cannot be provided. Only one column for each of the 6 particle coordinates is allowed at the same time. If the file contains multiple columns for the same coordinate, the columns not in use can be disabled with the `SKIP` flag.

For backwards compatibility with the old `DIST` block, a format that matches the old 14 colum file is also provided. For the recent addition of charge and PDGID, these columns must be added to this format. If no format is specified, the default is the old 14 column format described in Table 4.10.

Example

Below is an example of a `DIST` block using a 7-column input file with the length unit in millimetres.

```
DIST
  FORMAT ID X[mm] PX Y[mm] PY ZETA[mm] DELTA
  READ   partDist.dat
NEXT
```

Table 4.8: Available column formats in the DIST block.

Column Name	Units
Meta Columns	
`SKIP`	N/A
Disables the column in the file, that is, during parsing, the column is skipped.	
`ID`	N/A
The particle ID. Currently, this number must be in the range 1 to number of particles in the simulation, and they must be unique. There is no restriction on the order.	
`PARENT`	N/A
The particle's parent ID. If the parent ID is the same as the particle ID, the particle is considerd a primary particle.	
Transverse Coordinates	
`X, Y`	m or mm
The particle coordinate in the horizontal and vertical plane, respectively. These are the internal values used for tracking in SixTrack, and are read in as provides.	
`XP, YP`	[1]
The particle transverse momentum ratio relative to its total momentum, $p_x/p \approx x'$, $p_y/p \approx y'$. These are the internal values used for tracking in SixTrack, and are read in as provides.	
`PX, PY`	eV, keV, MeV, GeV or TeV
The particle transverse momentum. These values will be converted to SixTrack internal values.	
(The table continues on the next page)	

Column Name	Units
PX/P0, PXP0, PY/P0, PYP0	[1]
The particle transverse momentum relative to the reference momentum, p_x/p_0, p_y/p_0. The slash in the column name is optional. These values will be converted to SixTrack internal values.	
Longitudinal Position	
SIGMA	m or mm
The particle offset relative to the reference particle, σ. This is the internal value used for tracking in SixTrack, and is read in as provided.	
ZETA	m or mm
The particle offset relative to the reference particle, with relative velocity correction, $\zeta = \frac{\beta_0}{\beta}\sigma$. This value will be converted to SixTrack internal value SIGMA (σ).	
DT	ps, ns, μs, ms or s
The particle time delay, $\sigma = -\beta_0 \cdot \mathrm{d}t \cdot c$. This value will be converted to SixTrack internal value.	
Energy and Momentum	
E, P	eV, keV, MeV, GeV or TeV
The particle total energy or momentum, respectively. These are values used by SixTrack for tracking. Only one can be set, and the other is computed from the first.	
DE/E0, DEE0	[1]
The particle total energy relative to reference energy, $\Delta E/E_0$. This is converted to SixTrack internal value E after input.	
DP/P0, DPP0, DELTA	[1]
The particle total momentum relative to reference momentum, $\Delta P/P_0$. This is a value used by SixTrack for tracking. If provided as input, particle total energy and momentum is calculated from this value.	
PT	[1]
The particle total momentum relative to reference energy, $\Delta E/P_0 c$. This is converted to SixTrack internal value E after input.	
PSIGMA	[1]
The particle total momentum relative to reference energy and relativistic velocity, $\Delta E/\beta_0 P_0 c$. This is converted to SixTrack internal value E after input.	
Normalised Coordinates	
XN, YN, ZN, PXN, PYN, PZN	N/A
The six coordinates in nurmalised coordinates in units of $\sqrt{m}$. These are transformed by the beam parameters given in the DIST block after input.	
JX, JY, JZ, PHIX, PHIY, PHIZ	N/A
NOT YET IMPLEMENTED! The six coordinates in action coordinates. These are transformed by the beam parameters given in the DIST block after input.	
Ion Parameters	
MASS, M	eV, keV, MeV, GeV or TeV
The mass of the particle. The default value is the reference particle mass set in the SIMU, SYNC or HION block.	
CHARGE, Q	N/A
The charge of the particle in units of elementary charge. The default value is the reference particle mass set in the SIMU or HION block.	
ION_A, ION_Z	N/A
(The table continues on the next page)	

Column Name	Units
The ion A and Z values (atomic mass and atomic charge). The default value is the reference particle mass set in the SIMU or HION block. Both columns must be provided if one of them is.	
PDGID	N/A
The Particle Data Group ID of the particle. If it is not provided, it is either calculated from A and Z if given, or set to that of the reference particle in the SIMU or HION block.	
Spin Vector	
SX, SY, SZ	N/A
The three components of the particle spin vector. This information is currently not used for tracking, but is available for future additions to SixTrack.	
Multi-Columns Keywords	
4D	N/A
Equivalent to setting X PX Y PY with default units.	
6D	N/A
Equivalent to setting X PX Y PY ZETA DELTA with default units.	
NORM	N/A
Equivalent to setting XN PXN YN PYN ZN PZN.	
ACTION	N/A
Equivalent to setting JX PHIX JY PHIY JZ PHIZ.	
IONS	N/A
Equivalent to setting MASS CHARGE ION_A ION_Z PDGID with mass in units GeV.	
SPIN	N/A
Equivalent to setting SX SY SZ.	
OLD_DIST	N/A
Gives the old file format as described in Table 4.10. That is, it's equivalent to ID PARENT SKIP X[M] Y[M] SKIP XP[RAD] YP[RAD] SKIP ION_A ION_Z MASS[GEV] P[GEV] DT.	

4.7.2 Filling the Columns

The FILL keyword allows the user to fill the particle arrays with values generated based on a set of parameters. These can be both pseudo-random distributions, value ranges, or fixed values.

The fills are processed *after* particles are read from file or from direct PARTICLE declarations in the DIST block. This makes it possible to overwrite certain values after reading from file. The FILL feature can also be used to populate the arrays from scratch, which can be useful for filling the normalised arrays (see example below).

The different fill methods are listed in Table 4.9.

Note that the PARENT column cannot be filled. The SPIN columns are also currently disabled with this feature.

Also note that the FILL methods cannot create corrolated distributions. This can be achieved by using the normalised column formats, which triggers a normalisation after they have been filled.

If a fill method using random numbers is used, the random number generator also must be initialised with the RAND block. See Section 4.6.

The general format for this feature is:

```
FILL column method param1 ... paramN [firstIDX lastIDX]
```

`FILL`	The keyword selecting this feature.
`column`	The target column format. Must be one of the columns described in Table 4.8.
`method`	The fill method. Must be one of the methods described in Table 4.9.
`params`	The fill method parameters. These vary from method to method. See Table 4.9.
`firstIDX`	Optional: The first paricle index for this fill method. Defaults to particle 1.
`lastIDX`	Optional: The last paricle index for this fill method, where -1 indicates the last particle as request in the `SIMU` or `TRAC` block. Defaults to particle -1.

Example

The example below fills the normalised coordinate arrays with Gaussian distributions with a sigma cut, and fills the particle ID and ion columns. Writing to the normalised coordinate arrays in this manner triggers the normalisation routine to be called, here defaulting to use the internal T-matrix since no `TWISS` or `TMATRIX` keywords are set.

```
DIST
  EMITTANCE 2.5 2.5
  SEED 12
  FILL ID     COUNT  1   1
  FILL XN     GAUSS  1.0 0.0 5.0
  FILL PXN    GAUSS  1.0 0.0 5.0
  FILL YN     GAUSS  1.0 0.0 5.0
  FILL PYN    GAUSS  1.0 0.0 5.0
  FILL ZN     GAUSS  0.8 0.0 3.0
  FILL PZN    GAUSS  0.5 0.0 3.0
  FILL ION_A  INT    1
  FILL ION_Z  INT    1
  FILL CHARGE INT    1
  FILL MASS   FLOAT  938.272046
NEXT
```

Table 4.9: Available fill methods in the DIST block.

Method	Argumnent(s)
`INT`	`value [first last]`
Sets all values to a fixed integer. Can be used with column formats `ION_A`, `ION_Z`, `CHARGE`, and `PDGID`.	
`FLOAT`	`value [first last]`
Sets all values to a fixed floating point value. Can be used with all floating point column formats.	
`GAUSS`	`sigma mu [cut] [first last]`
Generates a normal random distribution with width `sigma` and offset `mu`, with an optional sigma `cut`. Can be used with all floating point column formats except `MASS`, `JX`,`JY`, and `JZ`.	
`RAYLEIGH`	`sigma [maxcut] [mincut] [first last]`
Generates a Rayleigh random distribution with width `sigma`, with an optional sigma `maxcut` and sigma `mincut`. Can be used with floating point column formats `JX`,`JY`, and `JZ`.	
`UNIFORM`	`lower upper [first last]`
Generates a uniform random distribution between the values `lower` and `upper`. Can be used with all floating point column formats except `MASS`.	
`LINEAR`	`lower upper [first last]`
Fills the array with floating point values ranging between the values `lower` and `upper` in equal steps. Can be used with all floating point column formats except `MASS`.	
`COUNT`	`start step [first last]`
Fills the array with integer values starting from `start`, with a given `step`. Can be used with column format `ID`.	

4.7.3 Support for the Old DIST Format

Tables 4.10 and 4.11 describe the old input file and echo file for the `DIST` block. These formats are still supported for backwards compatibility.

Table 4.10: Format of the ASCII file containing the distribution to be read by the `DIST` block.

#	Description
1	particle id
2	parent particle id
3	statistical weight (unused)
4	x [m]
5	y [m]
6	z (unused)
7	x' [1e-3]
8	y' [1e-3]
9	z' (unused)
10	mass number
11	atomic number
12	mass [GeV/c^2]
13	linear momentum [GeV/c]
14	time lag [s]

Table 4.11: The format of the ASCII file where the distribution read by the `DIST` block is echoed. See also Table 10.2 in `DUMP` for a more detailed description of the variables.

#	Variable	Unit
1	`xv1`	[mm]
2	`yv1`	[1e-3]
3	`xv2`	[mm]
4	`yv2`	[1e-3]
5	`sigmv`	[mm]
6	`ejfv`	[MeV/c]

Chapter 5

Machine Geometry

5.1 Single Elements

The *Single Elements* input block defines the name and type of linear and non-linear elements, the inverse bending radius or multipole strength respectively, and the strength and length of the elements. Linear and non-linear elements are distinguished by length – linear elements have a non-zero length and non-linear elements have zero length. Both kinds of elements can appear in the input block in arbitrary order. The input line has a different format for linear and non-linear elements. Moreover, the multipoles, being a set of non-linear elements, are treated in a special way. The maximum number of elements is set as a parameter (see Appendix A.2).

Keyword	`SING`
Data lines	Variable
Format	Described in the following sections.

5.1.1 Linear Elements

Each linear single element has a name, type, inverse bending radius, focusing and a non-zero length.

Format `name type` ϱ^{-1} `K length`

`name`	May contain up to 47 characters.
`type`	As shown in the table 5.1 .
ϱ^{-1}	Inverse bending radius in m^{-1}.
`K`	Focusing strength in m^{-2}.
`length`	Magnet length in meters.

Remarks

1. For the horizontal plane the bending radius is defined to be negative ($\varrho < 0$). This is different from other programs like MAD-X [23].
2. $K < 0$ corresponds to a horizontal focusing quadrupole.
3. For the length of an edge focusing element (type=8) the same value must be used as for the corresponding bending magnet. A sector bending magnet is transformed into a rectangular magnet with an edge focusing element of positive length on either side, while for the opposite transformation a negative length is required.

Table 5.1: Different Types of Linear Elements

Type	ϱ^{-1}	K	Description
0	0	0	Drift length magnet
1	X	0	Horizontal (rectangular) bending
2	0	X	Quadrupole (– focusing, + defocusing)
3	X	0	Horizontal (sector) bending
4	X	0	Vertical (rectangular) bending
5	X	0	Vertical (sector) bending
6	X	X	Horizontal combined function magnet
7	X	X	Vertical combined function magnet
8	X	0	Edge focusing

4. It is important to note that the splitting of a rectangular magnet, which is sometimes necessary if multipole errors are to be introduced, does change the linear optics. It is therefore advisable to replace the rectangular magnet with a sector magnet, which can be split without affecting the linear optics, and make an overall transformation into a rectangular magnet via edge focusing elements. Do not forget to use the total length of dipole as the length of the edge focusing element.

5.1.2 Non-Linear Elements

Format `name type` K_n`-strength rms-strength length`

`name`	May contain up to 47 characters.
`type`	As shown in table 5.2.
K_n`-strength`	Average multipole strength.
`rms-strength`	Random multipole strength.
`length`	Must be 0.

Table 5.2: Different Types of Non-linear Elements

Type	Strength	Description
0	–	Observation point (for instance for aperture limitations).
1	$b_1[\text{rad} \cdot \text{m}^0]$	Horizontal bending kick.
-1	a_1	Vertical bending kick.
2	$b_2[\text{rad} \cdot \text{m}^{-1}]$	Normal quadrupole kick.
-2	a_2	Skew quadrupole kick.
⋮		
10	$b_{10}[\text{rad} \cdot \text{m}^{-9}]$	Normal 20^{th} pole.
-10	a_{10}	Skew 20^{th} pole.

Remarks

1. Because the horizontal bending magnet is defined to have a negative bending radius, the sign for normal elements is different from other programs like MAD-X, while skew elements have the same sign.

2. Again contrary to other programs the factor $(n-1)!$ is already included in the multipole strength, which is defined as follows:

   - for normal elements:

   $$b_n(\text{SixTrack}) = \frac{-1}{(n-1)!}\, L_{\text{element}}\, b_n(\text{MAD})$$

   - for skew elements:

   $$a_n(\text{SixTrack}) = \frac{1}{(n-1)!}\, L_{\text{element}}\, a_n(\text{MAD})$$

3. Unlike in RACETRACK, the horizontal and vertical displacements do not fit into the 80 character input lines of SixTrack. They have to be introduced in a separate *Displacements of Elements* input block (5.2.2).

5.1.3 Multipole Blocks

A set of normal, normal-r.m.s., skew, and skew-r.m.s. errors can be combined effectively. The actual values for the strengths have to be given in a separate *Multipole Coefficient* input block (6.1) which must have the same name. To consider the curvature of dipoles which are replaced by drifts and dipole kicks this block is used in two different ways.

Format `name type cstr cref length`

Marker for high order kick (default)

`name`	May contain up to 47 characters.
`type`	Must be 11.
`cstr`	The bending strength given in the *Multipole Coefficient* input block (6.1) is multiplied with this factor.
`cref`	The reference radius given in the *Multipole Coefficient* input block (6.1) will be multiplied by this factor. If it is zero the multipole block will be ignored.
`length`	Must be 0.

Default + dipole curvature

`name`	May contain up to 47 characters.
`type`	Must be 11.
`cstr`	The bending strength [rad] of horizontal or vertical dipole. Internally the value is set to one to allow the processing of a multipole block (6.1).
`cref`	The length [m] of the dipole that is approximated by a kick. Internally this value is set to one to allow the processing of a multipole block (6.1).
`length`	length = -1: horizontal dipole. length = -2: vertical dipole.

Remarks The definition of the multipole strength in a block will be given in (6.1).

5.1.4 Generalized RF-Multipoles

A set of normal and skew RF-multipoles. The actual values for the strengths have to be given in a separate *Generalized RF-multipoles* input block (6.2) which must have the same name.

Format `name type`

`name` May contain up to 47 characters.

`type` Must be 11.

5.1.5 Solenoid

Format `name type` k_s $ks * l$

`name` May contain up to 47 characters.

`type` Type identifier is 25.

`ks` The strength k_s of the solenoid

`ks*l` The strength k_s of the solenoid multiplied with the length of the corresponding thick solenoid.

Remarks The solenoid is modeled as thin solenoid but a length the length of the real solenoid is still needed. This is different from the case with a multipole.

5.1.6 Cavities

Format `name type u0 harm lag`

`name` May contain up to 47 characters.

`type` Type identifier is +12 and −12 for above and below transition energy respectively.

`u0` Circumference voltage in [MV].

`harm` Harmonic number.

`lag` Lag angle [degrees] in the cavity (zero is default).

5.1.7 Beam–Beam Lens

Depending on the setting in the `BEAM` block of `fort.3` (Section 6.6), there are two ways to define a beam beam lens in the SINGLE ELEMENTS list.

When the `EXPERT` flag is set in the `BEAM` block: The parameters of the beam–beam lens is defined there. In this case, only the element name and type and the location within the lattice remain in the `fort.2` element definition.

Format `name type 0 0 0 0 0 0`

`name` May contain up to 47 characters.

`type` 20

The rest of the parameters are ignored and should be set to zero.

When the `EXPERT` flag is not set: The "traditional" format is used.

Format `name type h-sep v-sep strength-ratio` σ_h^2 σ_v^2 σ_hv^2

`name`	May contain up to 47 characters.
`type`	20
`h-sep`	Horizontal beam–beam separation [mm].
`v-sep`	Vertical beam–beam separation [mm].
`strength-ratio`	Strength ratio with respect to the nominal beam–beam kick strength. This is useful, in particular for 4D, to allow for splitting one beam–beam kick into several (longitudinally close by) kicks.
σ_h^2	When the flag `lhc=2` is set in the `BEAM` block of the `fort.3` file, this column represent the horizontal σ for the strong beam [mm^2].
σ_v^2	When the flag `lhc=2` is set in the `BEAM` block of the `fort.3` file, this column represent the vertical σ for the strong beam [mm^2].
σ_hv^2	When the flag `lhc=2` and `ibbc=1` is set in `BEAM` block of the `fort.3` file, this column represent the coupled σ for the strong beam [mm^2].

Remarks These beam–beam elements become active when the "Beam–Beam" input block 6.6 is used.

5.1.8 Wire

Format `name type`

`name`	May contain up to 47 characters.
`type`	15

Remarks The "wire" elements become active when the `WIRE` input block 6.7 is used. All parameters except name and type have to be set to zero, otherwise SixTrack aborts. The parameters for the wire are defined in the `WIRE` input block.

5.1.9 "Phase-trombone" or Matrix Element

Format `name type`

`name`	May contain up to 47 characters
`type`	22

Remarks These "trombone" elements become active when the "Phase Trombone Element" input block 6.8 is used.

5.1.10 AC Dipole

Format `name type ACdipAmp Qd ACdipPhase`

`name`	May contain up to 47 characters.
`type`	Type identifier is +16 and −16 for horizontal and vertical AC dipoles respectively.
`ACdipAmp`	Maximum excitation amplitude [Tm].
`Qd`	Excitation frequency in units of $[2 \times \pi]$.
`ACdipPhase`	Phase of the harmonic excitation in radians.

Remarks The length of the ramps and the flat top are specified in the "Displacement" block 5.2.2. The energy introduced in the "Initial coordinates" block 4.3 is used to compute the deflection angle.

5.1.11 Dipole Edge

Format `name type` r_{21} r_{43}

`name`	May contain up to 47 characters.
`type`	24
r_{21}	Horizontal edge focusing.
r_{43}	Vertical edge focusing.

Remarks MAD-X is outputting the correct format when using the dipedge element. An example of the hard edge model is described in the physics guide [16], which gives $r_{21} = -r_{43}$. Note that the values of the vertical edge focusing is dependent on the modeling of the fringe fields [24]. A particle with position x_1, y_1 and angle x'_1, y'_1 will have the angle x'_2, y'_2 after passing through the dipedge element. The following equations describe their relation:

$$x'_2 = x'_1 + x_1 \frac{r_{21}}{1+\delta} \tag{5.1}$$

$$y'_2 = y'_1 + y_1 \frac{r_{43}}{1+\delta} \tag{5.2}$$

5.1.12 Crab Cavity

Format `name type voltage frequency phase`

`name`	May contain up to 47 characters.
`type`	Type identifier is +23 and −23 for horizontal and vertical crab cavities respectively.
`voltage`	Crab Cavity voltage [MV].
`frequency`	Crab Cavity frequency [MHz].
`phase`	Phase of the excitation in radians.

Remarks

How to use the crab cavity from MAD-X (using rfmultipole) to SixTrack:
In the MAD-X script write:

```
MULT.1, FREQ=<freq in MHz>., KNL=\{V [MV]/E0[MeV]\}, PNL=\{phase\}, TILT=<H: 0; V:PI/2.>;
```

where phase is 0.25 (phase for multipoles in SixTrack). As an example, to have the effect of a vertical Crab Cavity of $f = 400$ MHz, $V = 6$ MV, beam energy [MeV]: `BEAM -> PC/1e3`, use the following line:

```
MULT.1, FREQ=400., KNL={6./BEAM -> PC/1e3}, PNL={0.25}, TILT=PI/2.;
```

This creates the following line in `fort.2`:

```
mult.1d  -23  6.00000000e+00  4.00000000e+02  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
```

If you dont want to have a vertical Crab Cavity then just remove the `TILT`. If you dont want to have CC but a simple dipole field then remove the `FREQ` parameter.

5.1.13 RF Multipole

Provides a kick in the form of

$$\Delta x' + i\Delta y' = \frac{k}{1+\delta}(x+iy)^n \cos(\phi - 2\pi f t) \tag{5.3}$$

$$\Delta\delta = P_0 \frac{k}{1+\delta} \frac{(x+iy)^{n+1}}{(n+1)!} \cos(\phi - 2\pi f t) \tag{5.4}$$

Format `name type name kick frequency phase`

Check the phase keyword

`name`	May contain up to 47 characters.
`type`	26: normal quadrupole, -26 skew quadrupole,
	27: normal sextupole, -27 skew sextupole,
	28: normal octupole, -28 skew octupole.
`kick`	maximum normalized kick k.
`frequency`	frequency f in [MHz].

Remarks
How to use the RF multipoles (from MAD-X to SixTrack):

2nd order multipole (quadrupole):
In the MAD-X script write:

```
MULT.1, KNL=\{0,-0.06*1e-3/brho\}, PNL=\{0, 0.25\};
```

where `-0.06*1e-3` is the b_2 value in units of $1/\mathrm{m}^{n-1}$.
This gives the following single element in `fort.2`:

```
mult.1q  26  6.00000000e-05  400.00000000e+00  -1.570796327e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
```

3rd order multipole (sextupole):
In the MAD-X script write:

```
MULT.1, FREQ=400., KNL=\{0,0,1159.*1e-3/brho\}, PNL=\{0,0,0.25\};
```

where `1159.*1e-3` is the b_3 value in units of $1/\mathrm{m}^{n-1}$.
This gives the following single element in `fort.2`:

```
mult.1s  27 -5.79500000e-01  4.00000000e+02  -1.570796327e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
```

4th order multipole (octupole):
In the MAD-X script write:

```
MULT.1, FREQ=400., KNL=\{0,0,0,-4.*1e-3/brho\}, PNL=\{0,0,0,0.25\};
```

where `-4.*1e-3` is the b_4 value in units of $1/\mathrm{m}^{n-1}$.
This gives the following single element in `fort.2`:

```
mult.1o  28  6.666666667e-04  4.00000000e+02  -1.570796327e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
```

The values of b_2, b_3, and b_4 used in the above examples were taken from Table II of paper [37] and normalized by the beam rigidity.

The effect of these multipoles was checked on a beam of particles with $x = x' = y' = 0$, and $y = 1, 2$, and 3 mm, with different z positions. The effect on y' was linear, quadratic and

cubic with y when using b_2, b_3, or b_4, respectively, as expected. Furthermore, the amplitude of the y' agrees with the analytical formulas found in the appendix of this paper [37] under "Normal quadrupole/sextupole/octupole".

Important note: $B\rho$ and the factorial $(n-1)!$ are already included in K2, K3 etc of MAD-X, i.e. $b_3 = 1159 \cdot 10^{-3}$ in MAD-X results in a kick as if b_3 is $1159 \cdot 10^{-3}/(n-1)!$. So in order for this paper's [37] analytical equations to be compatible with MAD-X, the equations for normal quadrupole should read as

$$\Delta x' = -\frac{b_2}{(2-1)!\ B\rho} \cdots .$$

5.1.14 Electron Lens

Format `name type`

`name` May contain up to 47 characters.
`type` 29

Remarks The "e-lens" elements become active when the `ELEN` input block 6.10 is used. All parameters except name and type have to be set to zero in the list of single elements, otherwise SixTrack aborts. The parameters for the e-lens are defined in the `ELEN` input block.

5.1.15 Scattering Point

Format `name type`

`name` May contain up to 47 characters.
`type` 40

Remarks The "scattering" elements become active when the `SCAT`ter input block 6.11 is used. All parameters except name and type have to be set to zero in the list of single elements, otherwise SixTrack aborts. The parameters of the scattering are defined in the `SCAT`ter input block.

5.1.16 Beam Position Monitor

Format `BPMname 0 0 0 0`

`BPMname` Must start with "BP" and maybe followed by 46 characters.

Remarks This element dumps the coordinates of the 1st particle to the file with name "BPMname". The file contains 7 columns: x,x', y,y', ct,$\delta p/p$ and E. Usual SixTrack units are used. Any number of `BPM` elements can be used but the names must differ.

5.1.17 X-Rotation

Format `name type angle`

`name` May contain up to 47 characters.
`type` 43
`angle` The rotation angle in radians.

Remarks A positive angle rotates the reference system in the clockwise direction.

5.1.18 Y-Rotation

Format `name type angle`

`name`	May contain up to 47 characters.
`type`	44
`angle`	The rotation angle in radians.

Remarks A positive angle rotates the reference system in the clockwise direction.

5.1.19 S-Rotation

Format `name type angle`

`name`	May contain up to 47 characters.
`type`	45
`angle`	The rotation angle in radians.

Remarks A positive angle rotates the reference system in the clockwise direction.

5.2 Block Definitions

In four-dimensional transverse tracking, the linear elements between non-linear elements can be combined to a single linear block to save computing time.

Keyword	`BLOC`
Data lines	> 1
Format	First line: `mper msym(1) ... msym(mper)` (integers) From second line: `block-name` {`element-name`}

Format Description

`mper`	Number of super periods. The following set of blocks is considered a `super-period`. The accelerator consists of `mper` super-periods.
`msym(i)`	$\pm$ 1 for each super-period. If msym(i) = 1, the i'th super-period will be built up in the order in which linear elements appear in the blocks below. If msym(i) = -1, the super-period will be built up in reverse order.
`block-name`	The name of the block with up to 47 characters.
`element-name`	The element names have to appear as a linear element in the list of "single elements" (5.1.1). If one line is too short to contain all the elements of a block, a line with additional elements to the same block can be added. At least 5 (five) blanks must appear at the beginning of the extra line so that names of blocks and names of linear elements in a block can be distinguished.

Remarks

1. When synchrotron oscillation is introduced, the linear elements can no longer be lumped into one block, because in that case even a drift length magnet is a non-linear element with respect to the longitudinal plane. However, the block structure is still kept to make use of the speed-up in case one can restrict the studies to the four-dimensional case.

2. The maximum number of blocks and the maximum number of entries in each block are defined as parameters (Appendix A.2).

3. The inversion of a super-period (msym(i) = −1) is presently no longer allowed.

5.2.1 Structure Input

The model of the accelerator is put together by constructing a sequence of blocks of linear elements, non-linear elements, observation points, and possibly a cavity with the keyword `CAV` used if this name does not appear in the list of single elements (5.1) with type ±12. In that case, its parameters are given in the *Synchrotron Oscillations* input block (4.4).

The Structure Elements block can either be specified as a list of Single Element names with multiple elements per line, or in a multi-column format with one element per line.

5.2.1.1 Format 1: List of Single Elements

The single column format defines a list of Single Elements in the order they appear in the machine. The maximum number of elements per line is 40.

Format { `structure-element` | `CAV` | `GO` }

`structure-element`	Structure elements must appear as non-linear and observation elements in the single element list or in the list of blocks of the *Block Definition* input block (5.2).
`CAV`	A cavity can be introduced by a keyword `CAV`. This element does not appear in the single element list (5.1).
`GO`	Starting point: the keyword `GO` denotes where the tracking is started and where the tracked coordinates are recorded at each turn.

Remarks Repetition of parts of the structure is indicated by parentheses with a multiplying factor N in front of them. If the left parenthesis "(" occurs in a line of input, the factor `N` is expected to be found in the preceding characters. If the characters are blank, `N` is set to 1. The right parenthesis ")" signals the end of the sequence to be repeated.

5.2.1.2 Format 2: Multi-Column List of Elements

This mode is enabled by the `MULTICOL` flag, which has to appear on the first line of the block. The block then takes a list of elements, one per line, with at least three values. The `GO` keyword is supported as in the list format, and has to appear alone on a single line. Note: Generating these imput files requires MadX version 5.05 or higher.

Format `ElemName FamName S`

`ElemName`	string	The unique element name (uniqueness is not enforced).
`FamName`	string	The single element name, or "family name".
`S`	float	The s-coordinate of the centre of the element.

The multi-column Structure block is generated automatically by the SixTrack converter in MadX when the flag `multicol` is present. The `ElemName` column is populated with the lower case version of the element name as defined in MadX. The `FamName` column is populated with the Single Element

name as defined in the Single Element block. The `S` column is the s-coordinate at the centre of the element as defined in MadX.

5.2.2 Displacement of Elements

This block allows to displace nonlinear elements in horizontal and vertical positions. With the r.m.s. values of the horizontal and vertical displacements it is possible to achieve a displacement that is different from element to element.

To simulate a measured closed orbit at the position of non-linear elements, it is convenient to use the *Displacement of Elements* input block instead of trying to produce a closed orbit by dipole kicks.

Keyword `DISP`
Data lines Variable

Format `name xd xdrms yd ydrms`

`name`	Name of the element which is displaced.
`xd`	Horizontal displacement [mm].
`xdrms`	r.m.s. of horizontal displacement [mm].
`yd`	Vertical displacement [mm].
`ydrms`	r.m.s. of vertical displacement [mm].

In the case of an AC dipole these variables are not meant for displacing this element but are used for the following AC dipole parameters:

Format `name nfree nramp1 nplato nramp2`

`name`	May contain up to 47 characters.
`nfree`	Number of turns free of excitation at the beginning of the run.
`nramp1`	Number of turns to ramp up the excitation amplitude from 0 to `ACdipAmp`.
`nplato`	Number of turns of constant excitation amplitude.
`nramp2`	Number of turns to ramp down the excitation amplitude.

Remarks In RACETRACK the displacements had been included in the *Single Element* input block (5.1). In SixTrack they must be given in the separate *Displacement of Elements* input block because of the limited length of one line of input.

Chapter 6

Special Elements

One advantage of SixTrack, that has been adopted from RACETRACK, is that it easily allows to define elements for a specific purpose. The special elements implemented util now are found in this section. All Special Elements should be written in the `fort.3` file.

6.1 Multipole Coefficients

Sets of normal and skew multipoles of up to tenth order, each with an r.m.s. value, can be combined with this block. The multipole kick is calculated using a Horner scheme, which saves considerably in computation time. Moreover, using the multipole block reduces the number of elements in the single element list (5.1).

Keyword	`MULT`
Data lines	2 to 12
Format	First line: `name` R_0 δ_0. Lines 2 to 12: B_n `rms-`B_n A_n `rms-`A_n.

Format Description

`name`	Name of the multipole block which must appear in the list of single elements (5.1.3).
R_0	Reference radius (in mm) at which the magnet errors are calculated. This makes it convenient to use values from field measurements.
δ_0	Bending strength of the dipole (in mrad). Field errors of line 2–11 are taken to be relative to the bending strength.

Remarks

1. The B_n and A_n are related to the b_n, a_n of the single nonlinear element (5.1.2) in the following way:
$$b_n = \delta_0 B_n R_0^{1-n} 10^{3n-6}$$
$$a_n = \delta_0 A_n R_0^{1-n} 10^{3n-6}$$
2. The sign convention and the factorial ($n!$) are treated as for the single non-linear elements in (5.1.2).
3. Multipoles of different names can be set to be equal using the `ORG` input block.
4. 22-poles are included ($n = 11$). By enlarging the parameter `MMUL` (Appendix A.2) up to 40-poles (`MMUL=20`) can be treated. To make the change of `MMUL` effective, it is of course necessary to recompile the program.

6.2 Generalized RF-Multipoles

Keyword `RFMU`
Data lines 2 to 21
Format First line: `name frequency` .
Lines 2 to 21: B_n ϕ_{Bn} A_n ϕ_{An}.

Format Description

`name`	Name of the block which must appear in the list of single elements (5.1.4).
B_n	Strength of the normal multipole of order n.
ϕ_{Bn}	Phase (in radians) of the normal multipole of order n.
A_n	Strength of the skew multipole of order n.
ϕ_{An}	Phase (in radians) of the skew multipole of order n.

6.3 Aperture Limitations

The aperture LIMItation block allows to define aperture limitations in the machine and hence describe the mechanical acceptance of the machine. In this way, it is possible to check if particles being tracked still remain inside the machine mechanical aperture or will be lost against the beam pipe.

In addition to the check against the detailed model of the machine, there is also a general (rectangular) aperture check at each non-zero length element. The general aperture check is always on, but in general values of the specifiers are set large enough (A.1) to define the short term dynamic aperture and be outside of any factual machine mechanical aperture.

6.3.1 General Description

Each non-linear (zero length) element defined in the *Single Element* input block (5.1.2) except multipole blocks (5.1.3) can be used to define aperture limitations, but it is highly recommended to use dedicated markers. Several aperture types are available to the user (see later).

The aperture limitations are taken into account during tracking by the online aperture checking algorithm, which verifies that the tracked particles falls within the mechanical acceptance of the machine. When a particle does not fit into the mechanical acceptance, it is removed from tracking; its coordinates at the point of loss are reported in a text file. A back-tracking algorithm finds the actual loss location interpolating the aperture profile by means of a bi-section method; the user can set the precision with which the longitudinal position is found (default is 10 cm). While on by default, the algorithm can be switched off by the user; in this case, the particle coordinates reported in the loss file are those at the aperture marker where the particle is found out of the mechanical aperture of the machine. Particles outside of the mechanical acceptance will be mercilessly killed, unless explicitly requested by the user (see Sec. 6.3.3).

The present implementation extends the functionalities developed in the context of the Fluka-SixTrack coupling [47, 48]. Please note that, if aperture markers are defined in the `LIMI` block, the aperture check is triggered only by the markers. On the other hand, the general aperture check is performed at every element and it is always on. Finally, no matter if the `LIMI` block is present or not, or if the back-tracking algorithm is on or off, SixTrack dumps all particles at their loss point in the `aperture_losses.dat` file (see Sec. 6.3.4).

Format Description

Keyword	`LIMI`
Data lines	Variable
Format	Each aperture marker is fully specified by means of its type and numerical parameters (see Sec. 6.3.2). Other input options are available to the user, headed by specific keywords, to control the respective parameters.

6.3.2 Specifying Aperture Marker

An aperture profile can be assigned to a SINGLE ELEMENT specifying its type and numerical specifiers:

Format: `name type aper_1 aper_2 aper_3 aper_4 x_off y_off angle`

`name`	The name of any non-linear (zero length) element in the *Single Element* input block (5.1.2) except multipole blocks (5.1.3).
`type`	Type of aperture limitation (string). See Tab. 6.2 for the types presently available.
`aper_1` to `aper_4`	Aperture specifiers (floats). Their actual meaning depend on the aperture type. The aperture specifiers are aligned to those of MAD-X [23], with the exception of the Racetrack, that can have an ellyptical corner. See Tab. 6.2 for their meanings. Only the Transition type needs 8 specifiers (hence from `aper_1` to `aper_8`).
`x_off` and `y_off`	Hor. and ver. offsets in mm.
`angle`	Tilt angle is in degrees. The tilt is around the offset point.

The last three numerical specifiers are optional, whereas the others are mandatory, depending on the type. Tab. 6.2 summarises the aperture types presently available and the meaning of the respective numerical specifiers; see also Fig. 6.1 for further geometrical clarifications. The list of aperture markers and specifiers can be given directly in the `fort.3` file, in the `LIMI` block, or via a text file, the name of which must be specified in the `LIMI` block with the `LOAD` keyword (see Sec. 6.3.3). For the convention on signs for the last three (optional) aperture specifiers, please see Fig. 6.2.

Table 6.2: Aperture types and specifiers. Only the mandatory specifiers are reported.

		Aperture specifier	
Name	`Type`	name	meaning
Circle	`CR`	`aper_1`	radius [mm]
Rectangle	`RE`	`aper_1`	hor. half-size [mm]
		`aper_2`	ver. half-size [mm]
Ellipse	`EL`	`aper_1`	hor. semi-axis [mm]
		`aper_2`	ver. semi-axis [mm]
RectEllipse	`RL`	`aper_1`	hor. half-size [mm]
		`aper_2`	ver. half-size [mm]
		`aper_3`	hor. semi-axis [mm]
		`aper_4`	ver. semi-axis [mm]
Octagon	`OC`	`aper_1`	hor. position of ver. side [mm]
		`aper_2`	ver. position of hor. side [mm]
		`aper_3`	angle of first cut corner [degree]
		`aper_4`	angle of second cut corner [degree]

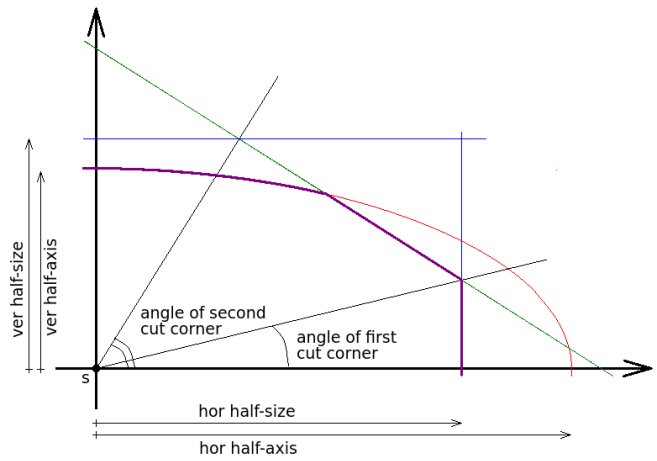

Figure 6.1: Sketch of the most general aperture profile currently treated by the aperture module. The purple line marks the actual aperture restriction as interpreted by the code with the given parameters.

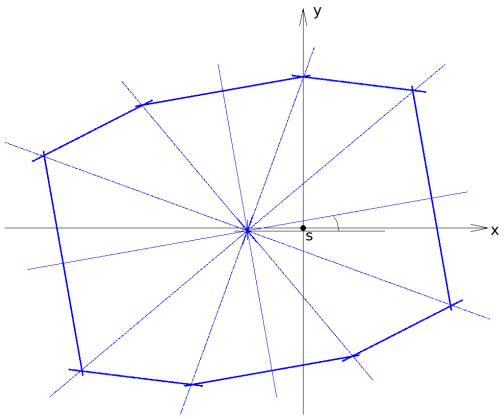

Figure 6.2: Sketch of the octagon aperture shown in the example. While both offsets are negative, the tilt angle is positive.

Ractetrack	`RT`	`aper_1`	hor. displ. of ellypse centre [mm]
		`aper_2`	ver. displ. of ellypse centre [mm]
		`aper_3`	hor. semi-axis [mm]
		`aper_4`	ver. semi-axis [mm]
Transition	`TR`	`aper_1`	hor. half-size [mm]
		`aper_2`	ver. half-size [mm]
		`aper_3`	hor. semi-axis [mm]
		`aper_4`	ver. semi-axis [mm]
		`aper_5`	hor. displ. of ellypse centre (a-la RACETRACK) [mm]
		`aper_6`	hor. displ. of ellypse centre (a-la RACETRACK) [mm]
		`aper_7`	angle of first cut corner [rad]
		`aper_8`	angle of second cut corner [rad]

6.3.3 Other Input Options

The user can customize the behavior of the `LIMI` block concerning the back-tracking algorithm. The syntax is via keywords, followed by positional arguments. Tab. 6.3 lists all available options, their syntax and meaning.

Table 6.3: Other input options of the `LIMI` block. Options are listed in alphabetical order.

keyword	argument	description
`BACKTRKOFF`		to disable back-tracking. Consequently, particle losses will occur only at aperture markers.
`DEBUG`		to enable debugging output for the aperture code. For the moment, this flag only increase the verbosity during parsing of `fort.3`.

LOAD	to load specifications of apertures from a file.	
	filename	ASCII file containing the specifications of apertures
PREC	to set custom accuracy in finding the actual loss location	
	precision	precision of back-tracking [m]
PRIN	to echo the profile of the machine aperture with s to a file. The precision set for back-tracking is used also as precision of the echo.	
	filename	ASCII file where to echo the machine aperture
	MEM	optional flag to dump aperture profile as in memory
SAVE	to avoid killing particles out of the mechanical aperture. Use this option for direct comparisons against BeamLossPattern profiles.	
XSEC	to dump aperture cross sections at specific s-locations (NOT OPERATIONAL YET!)	
	filename	ASCII file where to echo the cross section
	$s_{\min}$	first s-location where to get the cross-section [m]
	$s_{\max}$	last s-location where to get the cross-section [m] (optional)
	Δs	longitudinal steps where to calculate the cross-section [m] (optional)
	N	number of azimuthal angles (optional)

6.3.4 Format of aperture_losses.dat File

The aperture_losses.dat file lists all lost particles, along with their coordinates at the loss point. The file is generated no matter if the LIMI block has been inserted in the fort.3 or not, or if the back-tracking algorithm is on or off. Table 6.4 shows the format of the file. If the code is compiled with the FLUKA compilation flag, the particle ID is complemented by the parent particle ID and the statistical weight.

Table 6.4: Columns of the aperture_losses.dat file. The number between parentheses refers to the case SixTrack is compiled for coupling to Fluka, i.e. if the FLUKA compilation flag is on. A '-' means that a given column is not available when SixTrack is compiled the FLUKA compilation flag.

#	Description
1	turn number
2	index of entry in accelerator sequence
3	index of element in the SINGLE ELEMENT array
4	name of SINGLE ELEMENT
5	s-coordinate of loss point [m]
6	particle id
- (7)	parent particle id
- (8)	statistical weight
7 (9)	x at loss point [m]
8 (10)	x' at loss point []
9 (11)	y at loss point [m]
10 (12)	y' at loss point []
11 (13)	linear momentum [GeV/c]
12 (14)	energy deviation [eV]
13 (15)	time lag [s]

6.3.5 Example

The following example shows a typical use of the LIMI block. Debug messages are requested by the user. The description of the aperture markers is provided by the fort3.limi file, and the aperture profile is echoed and saved in the ape_dump.dat file. The back-tracking algorithm is kept on (BACKTRKOFF option being commented out). Precision of the back-tracking alghoritm is set at 1 mm.

```
LIMITATION OF APERTURE-------------------------------------------------
DEBUG
/ SAVE
PREC        0.001
PRIN  ape_dump.dat
LOAD  fort3.limi
/ BACKTRKOFF
NEXT
```

In the above example, the content of fort3.limi is the following:

```
aper.1        EL     29.0      29.0
aper.2        OC     40.0      30.0       0.5236    1.0472       -10.0       -2.0        0.174533
aper.3        RL     18.95     23.85      23.85      23.85
```

- aper.1 specifies an ellyptical aperture with both axes of 29 mm. Effectively, it is a circular aperture;
- aper.2 specifies an octagon aperture, 40 mm wide horizontally (half-width), and 30 mm height (half-height). The cut corner angles are 30 and 60 degrees. The aperture maker is offset by 10 mm and 2 mm on the horizontal and vertical planes, respectively. The aperture is tilted by 10 degrees;
- aper.3 specifies a rectellipse aperture. It is actually equivalent to a circular aperture of 23.85 mm of radius, and vertical bars at x=±18.95 mm.

Fig. 6.2 shows a sketch of the ocatogon aperture described by the aper.2 marker, to show the convention on signs.

6.4 Power Supply Ripple

Note: The RIPP block is deprecated since release 4.5.20, and the functionality is now provided by the DYNK block (6.5). A fort.3 file containing a RIPP block is therefore no longer valid, and will result in an error message. The description below is therefore only provided as a reference for those who need to convert old input files.

If power supply ripple is to be considered this input data block can be used. A non-linear quadrupole is expected as a ripple element (type=2 and zero length in the single element list (5.1.2)), but in principle other non-linear elements are also allowed. Ripple depth, ripple frequency and starting phase of the ripple frequency are the input parameters.

Keyword	RIPP
Data lines	Variable
Format	name depth frequency start-phase nrturn

Format Description

`name`	Name of the non-linear element in the *single element* block (5.1.2).
`depth`	Maximum kick strength of the ripple element. A quadrupole kick is usually expected.
`frequency`	Given in number of turns (a real value is allowed) of one ripple period.
`start-phase`	Initial phase of the ripple element.
`nrturn`	Initial number of turns, for prolongation runs the number of turn already done.

6.5 Dynamic Kicks

The DYNamic Kicks module [38, 39] allows time-dependent modification of the settings of single elements. The supported elements and attributes are listed in Table 6.7. The settings can be computed on-the fly using several functions, loaded from input files or a combination, as described in Table 6.6.

If the `DYNKSETS` flag is present, `DYNK` produces an output file `dynksets.dat`. This file contains the setting of all elements and attributes for which `DYNK` is active. It is written in all turns of the simulation, even if `DYNK` is not active in that exact turn.

Keyword	`DYNK`
Data lines	Variable
Format	There are four types of statements possible in a `DYNK` block, listed in the following subsections.

6.5.1 FUN Statements

Format: `FUN function-name function-type arg1 arg2 arg3 ...`

This statement defines a function, i.e. something which when evaluated, produces a numerical value, which can be used to set the value of an element attribute. The functions in `DYNK` all have a unique name, and they may take up to 7 arguments (a limitation imposed by the internal parameter `getfields_n_max_fields`). The function type must be one of those listed in Table 6.6.

A function may be defined so that it uses the result of another function, which must be defined above it in the `DYNK` block. This requirement avoids any possibility for infinite recursion. The functions are only evaluated when needed, i.e. when used by a `SET` statement in that turn (6.5.2). The functions may thus be evaluated multiple times in one turn (if used by multiple `SET` statements which are active in that turn, or referenced by multiple other `FUN` statements which are themselves used more than once in that turn), or it may not be evaluated at all. The functions are always evaluated as a function of the current turn number t, which may be shifted by a turn-shift specified in a `SET` statement (6.5.2).

Function names have a maximum length of 20 characters.

Table 6.6: Available function types in DYNK.

Type name	Arguments	Description
"System" functions		
`GET`	`element-name[string]` `attribute-name[string]`	Extracts the original value of a setting, i.e. as specified in the SINGLE ELEMENT section (Sec. 5.1). Attributes as used for `SET`, see Table 6.7.
`FILE`	`filename[string]`	Loads the settings from file; the file is expected to be an ascii file with two columns where the first column is the turn number (should start at 1 and include all turns up to as long as is wanted), and the second column is the value for that turn number.

(The table continues on the next page)

Type name	Arguments	Description
FILELIN	filename[string]	Similar to FILE, but any double can be used as the turn number as long as they are monotonically rising. When evaluated, the function interpolates from the line-segments specified in the file.
PIPE	inPipeName[string] outPipeName[string] ID[string]	Uses a pair of UNIX FIFOs, through which it can communicate with an external program. When evaluated, it sends a message through the outpipe, and then waits for a message on the in-pipe which should contain the value the FUN should returned. The ID is used in case several DYNK PIPE FUNs are using the same outPipe and inPipe, so that the controlling external program can choose what to calculate. For more details, see the example below. Also note that PIPE is not available in the checkpoint/restart version of SixTrack.
RANDG	seed1[int] seed2[int] mu[real] sigma[real] mcut[int]	Returns a pseudorandom number generated from a Gaussian distribution. The mean value and width is controlled by mu and sigma, while mcut is the maximum number of sigmas to generate numbers up to; set to 0 to disable this cut. The integers seed1 and seed2 are the seed used to initialize the RANECU generator. Note that every RANDG function defined in DYNK uses its own separate random number stream.
RANDU	seed1[int] seed2[int]	Returns a pseudorandom number generated from a uniform distribution. The integers seed1 and seed2 are the seed used to initialize the RANECU generator. Note that every RANDU function defined in DYNK uses its own separate random number stream.
RANDON	seed1[int] seed2[int] P[float]	Returns the value of 1.0 or 0.0 resulting of the weighting with the probability P of a pseudorandom number generated from a uniform distribution . The integers seed1 and seed2 are the seed used to initialize the RANECU generator. Note that every RANDON function defined in DYNK uses its own separate random number stream.
Filters		
FIR	N[int] filename[string] baseFun[string]	Applies a Finite Impulse Response (FIR) filter of order vN to the function baseFun. The output is given as $y[t] = \sum_{i=0}^{N} b_i * x[t-i]$, where t is the current turn and $x[t-0]$ is the result of the most recent call to baseFun. The coefficients $b_0 \dots b_N$ and initial values of $x[t-0] \dots x[t-N]$ are loaded from the given file filename, which is a space-separated ascii file with three columns. These columns are (1) row index [int], (2) coefficients b_i [float] and (3) initial values of the $x[]$ array [float]. The row indices are expected to go from 0 to at least N in steps of 1. Note that the filter is stepped once per call, i.e. the array $x[]$ is shifted once every time the FUN is called. Also note that when called, the filter is first stepped, then the new value is filled into the first position in $x[]$, and finally the sum is evaluated. This means that the last value in the $x[]$ array is never used, while the first value ($x[t-0]$) is immediately pushed into $x[t-1]$ before the first evaluation.

(The table continues on the next page)

Type name	Arguments	Description
IIR	N[int] filename[string] baseFun[string]	Applies an Infinite Impulse Response (IIR) filter of order N to the function baseFun. This is very similar to FIR, except that it also uses its own previous outputs. The sum is thus written as $y[t] = \sum_{i=0}^{N} b_i * x[t-i] + \sum_{i=1}^{N} a_i * y[t-i]$. The file filename is identical to that which is used for FIR, except for adding two more columns. These columns are (4) $a_0 \dots a_N$ [float] and (5) initial values for the $y[]$ array [float]. Note that a_0 is never used, and the value of $y[t-0]$ is pushed back to $y[t-1]$ before the first evaluation of the sum, such that $y[t-N]$ is never used.
2-operand operators		
ADD	function-name-1[string] function-name-2[string]	Evaluate the functions referenced by function-name-1 and function-name-2, and return the sum of the results.
SUB	function-name-1[string] function-name-2[string]	Similar to ADD, but return the result of function1 minus function2.
MUL	function-name-1[string] function-name-2[string]	Similar to ADD, but return the product of the results.
DIV	function-name-1[string] function-name-2[string]	Similar to ADD, but return the result of function1 divided by function2
POW	function-name-1[string] function-name-2[string]	Similar to ADD, but return the result of function1 raised to the power of function2.
1-operand operators		
MINUS	function-name	Returns the value of the named function, with the opposite sign.
SQRT	function-name	Returns the square root of the value generated by the named function.
SIN	function-name	Returns the sine of the value generated by the named function.
COS	function-name	Returns the cosine of the value generated by the named function.
LOG	function-name	Returns the natural logarithm of the value generated by the named function.
LOG10	function-name	Returns the common logarithm of the value generated by the named function.
EXP	function-name	Returns the natural exponential function e^x, where x is the value generated by the named function.
ABS	function-name	Returns the absolute value of the value generated by the named function.
Polynomial and elliptical functions		
CONST	value[real]	Always returns the value specified.
TURN	(none)	Return the turn number, i.e. $y(t) = t$.
LIN	a[real] b[real]	Computed value from the linear function $y(t) = a \cdot t + b$.
LINSEG	x1[real] x2[real] y1[real] y2[real]	The function is defined by a line segment between the points (x_1, y_1) and (x_2, y_2), and undefined for $x < x_1$ and $x > x_2$. It is required that $x_1 < x_2$.
QUAD	a[real] b[real] c[real]	Computed value from the quadratic function $y(t) = a \cdot t^2 + b \cdot t + c$.
QUADSEG	x1[real] x2[real] y1[real] y2[real] deriv1[real]	The quadratic function is defined by overlapping the quadratic curve segment which passes through the points (x_1, y_1) and (x_2, y_2), and $\mathrm{d}y/\mathrm{d}x$ at x_1 is deriv1. The quadratic coefficients a, b, c are calculated as $a = \frac{\texttt{deriv1}}{x_1 - x_2} + \frac{y_2 - y_1}{(x_1 - x_2)^2}$, $b = \frac{y_2 - y_1}{x_2 - x_1} - (x_1 + x_2) \cdot a$ and $c = y_1 + \left(-x_1^2 \cdot a - x_1 \cdot b\right)$.
Trancendental functions		

(The table continues on the next page)

Type name	Arguments	Description
SINF	A[real] omega[real] phi[real]	Computes $y(t) = A\sin(\omega t + \phi)$.
COSF	A[real] omega[real] phi[real]	Computes $y(t) = A\cos(\omega t + \phi)$.
COSF_RIPP	A[real] period[real] phi[real]	Computes $y(t) = A\cos\left(\frac{2\pi(t-1)}{\text{period}} + \phi\right)$. This specialized cosine is provided for compatibility, to be used when replacing old RIPP blocks.
Specialized functions		
PELP	tinj[real] Iinj[real] Inom[real] A[real] D[real] R[real] te[real]	This function describes a patched "Parabolic-Exponential-Linear-Parabolic" function, as used for ramping the LHC dipoles and described in [40, Appendix C] and [41]. The parameters are: • The injection time tinj, which is the time (in turn numbers) when the ramp starts. • The injection value Iinj, which is the value when $t \leq t_{inj}$ • The final value Inom, which is the value after the end of the ramp. • The acceleration parameter A, which describes how fast the current is growing in the first (parabolic) segment. • The decelertation parameter D, which describes how fast the current growths flattens out in the forth (parabolic) segment. • The ramp rate R, which describes the maximum ramp rate, seen in the third (linear) segment. • The start time of the ramp te, which describes at what time it switches from the parabolic (first) to the exponential (second) segment.
ONOFF	p1[int] p2[int]	This function is a periodic "pulse width modulation" with period p2 and pulse length p1. It may be described as $y(t) = \{1.0 \text{ if } \text{mod}(t-1, p2) < p1\}\,;\ \{0.0 \text{ otherwise}\}$. The reason for using $t-1$ is that the modulus is naturally zero-based, while SixTrack counts turns starting from 1. Note that it is expected that $p1 >= 0$, $p2 > 1$, and $p1 <= p2$. Also note that for negative t, the function will always return 1.0.

6.5.2 SET Statement

Format: `SET element-name attr-name func-name first-turn last-turn turn-shift`

This statement defines an element setpoint, which changes an element/attribute, `attr-name`, to the value computed by the given function, `func-name`. The `SET` statement becomes active when the turn number reaches `first-turn`, and switches off once `last-turn` has been passed. When switched off, the value applied in `last-turn` stays for the rest of the simulation, or until overwritten by another `SET`. If `last-turn` equals -1, the `SET` is active until the end of the simulation.

The element type and attribute combinations that can be used in `DYNK` are shown in Table 6.7.

The argument `turn-shift` is an integer (positive, negative, or zero) number which is added to the current turn number before computing the function. Thus, in order to (as an example) apply an exponential decay from the value v_0 starting in turn t_0 using the function defined as $f(t) = V_0 \exp(-t/\tau)$, a `turn-shift` $-t_0$ should be applied.

In addition to changing single element attributes, it is also possible to use `DYNK` to change certain global attributes such as the reference energy. This is done through the "element" `GLOBAL VARS`; for example one may want to simulate an energy ramp following the function `eramp` throughout the whole simulation. For this, one would use the `SET` command

```
SET GLOBAL-VARS E0 eramp 1 -1 0
```

Because of this, SixTrack does not accept a real single element in `fort.2` named `GLOBAL-VARS` if `DYNK` is active.

Element type (idx)	Attribute	Units	Description
Standard thin elements (±1 – ±10), Section 5.1.2	`average_ms`	radians * m^{-n}	See Table 5.2
Multipoles (11), Section 6.1	`scaleall`	-	Multiplies all order by this factor
	`a{ORDER}rms, e.g a1rms`	radians * m^{-n}	Corresponds to the 4th column in the multipole block.
	`b{ORDER}rms, e.g b2rms`	radians * m^{-n}	Corresponds to the 2nd column in the multipole block.
	`a{ORDER}str, e.g a3str`	radians * m^{-n}	Corresponds to the 1th column in the multipole block.
	`b{ORDER}str, e.g b4str`	radians * m^{-n}	Corresponds to the 3rd column in the multipole block.
RF cavities (±12), Section 5.1.6	`voltage`	MV	One-turn accelerating voltage
	`harmonic`	–	Harmonic number of the cavity
	`lag_angle`	degrees	Lag angle of the cavity
Beam-Beam 6d (expert) (20), Section 6.6	`h-sep`	mm	Horizontal offset
	`v-sep`	mm	Vertical offset
	`strength`	-	Strength-ratio
	`Sxx`	-	$\Sigma_{x,x}$
	`Sxxp`	-	$\Sigma_{x,xp}$
	`Sxpxp`	-	$\Sigma_{xp,xp}$
	`Syy`	-	$\Sigma_{y,y}$
	`Syyp`	-	$\Sigma_{y,yp}$
	`Sypyp`	-	$\Sigma_{yp,yp}$
	`Sxy`	-	$\Sigma_{x,y}$
	`Sxyp`	-	$\Sigma_{x,py}$
	`Sxpy`	-	$\Sigma_{xp,y}$
	`Sxpyp`	-	$\Sigma_{xp,yp}$
Beam-Beam 4d (expert) (20), Section 6.6	`h-sep`	mm	Horizontal offset
	`v-sep`	mm	Vertical offset
	`strength`	-	Strength-ratio
	`4dSxx`	-	$\Sigma_{x,x}$
	`4dSyy`	-	$\Sigma_{y,y}$
	`4dSxy`	-	$\Sigma_{x,y}$
RF multipoles (±23, ±26 – ±28), Section 5.1.12	`voltage`	MV	Kick voltage
	`frequency`	MHz	Frequency
	`phase`	radians	Offset between zero-crossing and ideal bunch center
Electron lens (29), Section 6.10	`theta_R2`	mrad	Angular kick at R2
	`elens_I`	A	Electron current
	`elens_Ek`	keV	Electron kinetic energy

Scattering (40), Section 6.11	`scaling`	–	Scaling of probability, see Section 6.11, paragraph about ELEM command.
Generalized RF-multipoles (41), Section 6.2	`a{ORDER}amp, e.g a1amp`	m^{-n}	Corresponds to the 3rd column in the generalized RF-multipole blocks.
	`b{ORDER}amp, e.g b2amp`	m^{-n}	Corresponds to the 1st column in the multipole block.
	`a{ORDER}pha, e.g a3pha`	radians	Corresponds to the 4th column in the multipole block.
	`b{ORDER}pha, e.g b4pha`	radians	Corresponds to the 2nd column in the multipole block.
`GLOBAL-VARS` Not a real element, changes global variable	`E0`	MeV	Reference energy of synchronous particle

Table 6.7: Element types and attributes available in `DYNK` .

6.5.3 Additional Flags

Flag `DYNKSETS`

The presence of this statement in a `DYNK` block switches on the writing of the output file `dynksets.dat` in every line. This can be useful to save disk space in very long simulations.

This flag replaces the `NOFILE` flag that in version 5.3.4 and earlier was used to disable the writing of `dynksets.dat`. The file is now disabled by default, and has to be explicitly requested.

Flag `DEBUG`

This statement switches on extra "debugging" output from `DYNK`. This can be useful if debugging the code or if debugging the input.

6.5.4 Output File `dynksets.dat`

When the `DYNKSETS` flag is present in the `DYNK` block, a file `dynksets.dat` is created and in the current working directory. This file contains first a header

```
# turn element attribute SETidx funname value
```

followed by rows of data in the format specified in the header. This data is written for all element/attribute combinations and in all turns, wether a `SET` is active for this element/attribute in this turn or not. If no `SET` is active when the line is written out, the `SETidx` is written as -1, and the `funname` is "N/A". If a `SET` is active when the line is written out, the `SETidx` is the index of the currently active `SET` statement, where the first statement occurring in `fort.3` has index 1, etc. Similarly, the `funname` is the name referencing the currently active `FUN` statement.

6.5.5 Examples

Replacement of `RIPP` Block

One use of the `DYNK` block is to replace the functionality of the `RIPP` block (Section 6.4). The `FUN` type `COSF_RIPP` is provided for exactly this purpose, and provides an exact replacement. As an example, the `RIPP` block in the SixTest test-case prob1 looks like (slightly reduced in size):

```
RIPPLE OF POWER SUPPLIES
  dmqx1f5015+2      3.2315D-10    224.9
  dmqx2af5015+2    -3.2315D-10    224.9
  dmqx1f10mel5+2    2.5246D-16    0.0011245
NEXT
```

This can be replaced by the following:

```
DYNK
  FUN RIPP-dmqx1f5015+2 COSF_RIPP 3.2315D-10 224.9 0.0
  SET dmqx1f5015+2 average_ms RIPP-dmqx1f5015+2 1 -1 0
  FUN RIPP-dmqx2af5015+2 COSF_RIPP -3.2315D-10 224.9 0.0
  SET dmqx2af5015+2 average_ms RIPP-dmqx2af5015+2 1 -1 0
  FUN RIPP-dmqx1f20kl5+2 COSF_RIPP 2.5246D-12 0.56225 0.0
  SET dmqx1f20kl5+2 average_ms RIPP-dmqx1f20kl5+2 1 -1 0
NEXT
```

Here, each `RIPP` data line is replaced with two lines, one FUN statement for generating the function, and one `SET` statement for applying the value. Note that the `SET` statements have an end time `-1`, meaning it is used until the end of the simulation.

Starting tracking inside a bump

This example was taken from the paper [38], and demonstrates how a bump can be temporarily disabled if the starting point of the tracking is inside of it. The reason for doing this is removing the necessity of generating a starting distribution with the bump already applied. Here, the HL-LHC v1.1 lattice is used, with vertical crab cavities around the first interaction point (IP1, ATLAS), which is also the point where the tracking is started. The crab cavities opening the bump are called `CRAB_IP1_L1···4`, while the closing cavities are `CRAB_IP1_R1···4`. The `DYNK` block for this looks like:

```
DYNK
  FUN zero CONST 0.0
  FUN CV_1R1 Get CRAB_IP1_R1 voltage
  FUN CV_1R2 GET CRAB_IP1_R2 voltage
  FUN CV_1R3 GET CRAB_IP1_R3 voltage
  FUN CV_1R4 GET CRAB_IP1_R4 voltage
  SET CRAB_IP1_R1 voltage zero 1 1 0
  SET CRAB_IP1_R2 voltage zero 1 1 0
  SET CRAB_IP1_R3 voltage zero 1 1 0
  SET CRAB_IP1_R4 voltage zero 1 1 0
  SET CRAB_IP1_R1 voltage CV_1R1 2 2 0
  SET CRAB_IP1_R2 voltage CV_1R2 2 2 0
  SET CRAB_IP1_R3 voltage CV_1R3 2 2 0
  SET CRAB_IP1_R4 voltage CV_1R4 2 2 0
NEXT
```

Here, the function `zero` is defined such that it always returns 0.0, and is used to switch off the closing cavities in the first turn, i.e. when the beam exits the bump. Further, the functions `CV_1R1···1R4` and `CV_1L` are used to store the original value of the voltages, without having to explicitly enter them into the DYNK block.

The `SET` statements then first sets the voltage of all the cavities to zero in turn 1, and then in turn 2 sets it to their respective "switched on" voltages. The `SET` statements end after turn 2, but the last values are retained.

This means that when the simulation starts with the bunch in `IP1`, it exits the bump without any kicks from the closing crab cavities. It then comes around (still in turn 1), and encountered the switched-on opening cavities `CRAB_IP1_L1···4`, which crabs the beam. After passing through IP1, the turn counter is increased from 1 to 2, triggering the SET statements to switch on the closing cavities `CRAB_IP1_R1···4` as well.

Ramp and exponential decay of crab voltage, combined with a linear drift of crab phase

This slightly more complicated example builds on the example given above. It shows how to change two parameters (voltage and phase) of several objects. It also demonstrates how functions can

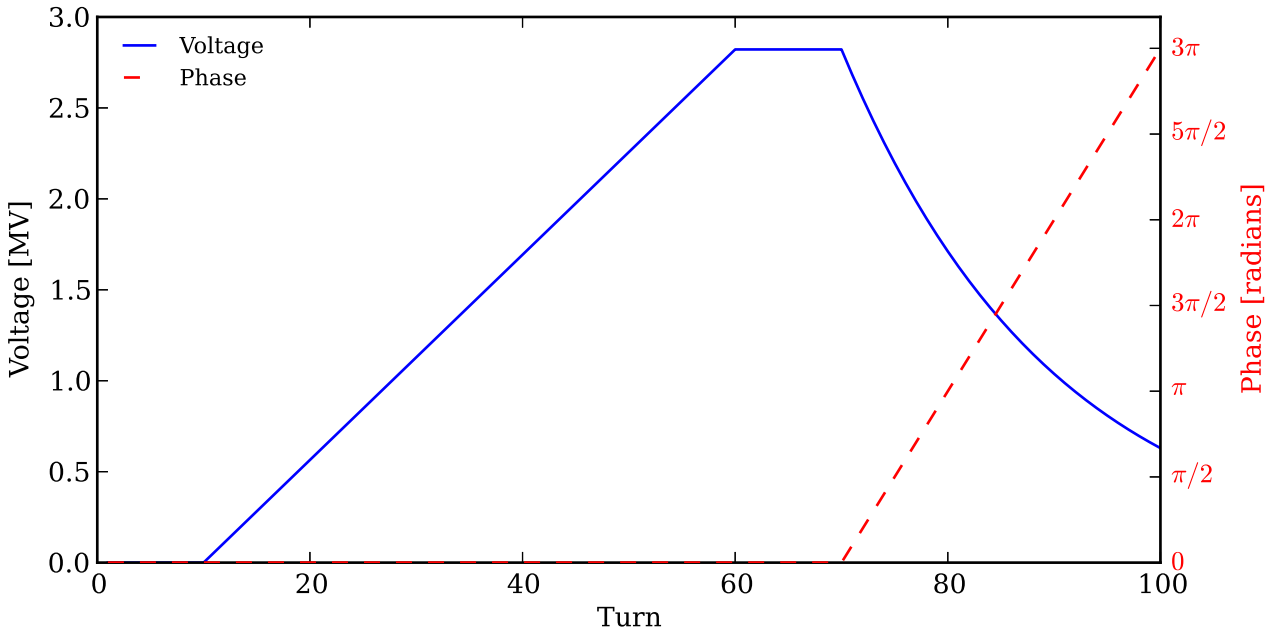

Figure 6.3: Singals generate by `DYNK` example for ramp + exponential decay of crab voltage, and also linear drift of crab phase. Only the signals for CRAB_IP1_L1 are shown. The plot is made from the data in `dynksets.dat`.

be chained together, making more complicated functions. Some of the resulting functions are shown in Figure 6.3, and the `DYNK` block here looks like:

```
DYNK
  DEBUG

  FUN zero CONST 0.0
  FUN CV_R1 GET CRAB_IP1_R1 voltage
  FUN CV_R2 GET CRAB_IP1_R2 voltage
  FUN CV_R3 GET CRAB_IP1_R3 voltage
  FUN CV_R4 GET CRAB_IP1_R4 voltage
  FUN CV_L  GET CRAB_IP1_L1 voltage
  FUN ramp LIN 0.02 0
  FUN ramp_R1 MUL CV_R1 ramp
  FUN ramp_R2 MUL CV_R2 ramp
  FUN ramp_R3 MUL CV_R3 ramp
  FUN ramp_R4 MUL CV_R4 ramp
  FUN ramp_L  MUL CV_L  ramp
  SET CRAB_IP1_R1 voltage zero 1 10 0
  SET CRAB_IP1_R2 voltage zero 1 10 0
  SET CRAB_IP1_R3 voltage zero 1 10 0
  SET CRAB_IP1_R4 voltage zero 1 10 0
  SET CRAB_IP1_L1 voltage zero 1 9 0
  SET CRAB_IP1_L2 voltage zero 1 9 0
  SET CRAB_IP1_L3 voltage zero 1 9 0
  SET CRAB_IP1_L4 voltage zero 1 9 0

  SET CRAB_IP1_R1 voltage ramp_R1 11 61 -11
  SET CRAB_IP1_R2 voltage ramp_R2 11 61 -11
  SET CRAB_IP1_R3 voltage ramp_R3 11 61 -11
  SET CRAB_IP1_R4 voltage ramp_R4 11 61 -11
  SET CRAB_IP1_L1 voltage ramp_L 10 60 -10
  SET CRAB_IP1_L2 voltage ramp_L 10 60 -10
  SET CRAB_IP1_L3 voltage ramp_L 10 60 -10
  SET CRAB_IP1_L4 voltage ramp_L 10 60 -10

  ! Voltage decay and detuning
  FUN expCore LIN -0.05 0.0
  FUN decay EXP expCore
  FUN decayScaled MUL decay CV_L
  SET CRAB_IP1_L1 voltage decayScaled 70 100 -70
  SET CRAB_IP1_L2 voltage decayScaled 70 100 -70
  SET CRAB_IP1_L3 voltage decayScaled 70 100 -70
  SET CRAB_IP1_L4 voltage decayScaled 70 100 -70
  FUN phasedrift LIN 0.3141592654 0.0
  SET CRAB_IP1_L1 phase phasedrift 70 100 -70
  SET CRAB_IP1_L2 phase phasedrift 70 100 -70
  SET CRAB_IP1_L3 phase phasedrift 70 100 -70
  SET CRAB_IP1_L4 phase phasedrift 70 100 -70
NEXT
```

The first functions defined here are the same as above, storing the default values (as defined in the single element list) for the relevant elements and also zero. Then follows a normalized linear ramp function `ramp`, with gradient $0.02 = 1/50$. This is then used by the "specialized" ramp functions `ramp_R1`$\cdots$`R4`, which scales `ramp` so that the end point is the standard voltages for $t \in 0 \ldots 50$.

These functions are used to first set the crabs to 0.0 for the first 9 revolutions, and in the 10th revolution the ramp starts. As the `ramp` function is defined starting at turn 0, a shift -10 or -11 is applied to the ramps. The ramp is switched off after turn 60/61, leaving the crabs to be operating at the last `SET` value.

Further, we want to demonstrate a failure in the crab voltage. This is done using an exponential decaying function $V(t) = V_0 \exp(-0.05t)$, which is implemented as three chained functions:

expCore	$f(t) = -0.05t + 0.0$
decay	$g(t) = \exp(f(t)) = \exp(-0.05t + 0.0)$
decayScaled	$h(t) = V_0 \cdot g(f(t)) = V_0 \cdot \exp(f(t)) = \exp(-0.05t + 0.0)$

For the `SET`, the time t is then shifted by -70 turns, so that the functions are evaluated starting at $t = 0$.

Using the PIPE function

To use the PIPE functionality, add a FUN and SET to the DYNK block such as:

```
FUN pipe1 PIPE /tmp/pip1 /tmp/pip2 myID1 4242
SET  ACFCA.AR1.B1 voltage pipe1 10 -1 -9
```

Then create the two pipes using the mkfifo UNIX command, e.g. mkfifo pip1 and mkfifo pip2 in the chosen directory. When starting SixTrack, it will first open the input pipe (while reading the DYNK block), and wait for the external program to do the same. This can be simulated by running cat > pip1; it is also possible to open the input pipe before starting SixTrack. After opening the input pipe, SixTrack will open the output pipe, again this can be simulated by running cat pip2, and again this pipe may be opened before starting SixTrack. Note that when SixTrack ends, the output pipe will be closed, so the receiving cat process is terminated.

After opening the output pipe, SixTrack writes the line DYNKPIPE !******************! to this file. It then writes a line similar to INIT ID=myID1 for FUN=pipe1 for each FUN using this output pipe.

During tracking, when one of the PIPE FUNs are called SixTrack writes a line similar to GET ID=myID1 TURN= 1 to the output pipe. Note that the turn number is the one passed to the FUN from SET, i.e. including any turn-shift. It then waits for a single floating point number to be written (in ascii) to the input pipe, which is then read and returned from the FUN.

6.6 Beam–Beam Element

The beam–beam kick, including a separation of the beams, is treated à la Basetti and Erskine [19] and implemented as in MAD-X [23]. However, a much faster but nevertheless precise calculation using interpolation can be used [25]. Since SixTrack version 3, the beam–beam is also available in the 6D form à la Hirata [20]. Lastly, the linear coupling has been considered in 4 and 6 dimensional phase space [21].

Keyword	BEAM
Data lines	> 1
Format	Two different input formats are available, "traditional" and "EXPERT". If "EXPERT" mode is wanted, this is triggered by adding the flag EXPERT on the first line of the block.

Traditional format

First line: partnum emitnx emitny sigz sige ibeco ibtyp lhc ibbc
Further lines: name ibsix xang xplane xstr

partnum	float	Number of particles in bunch
emitnx,emitny	floats	Horizontal and vertical normalised emittance respectively [μm·rad]
sigz,sige	floats	r.m.s. bunch length [m] and r.m.s. energy spread
ibeco	integer	Switch (0 = off; 1 = on) to subtract the closed orbit introduced by the separation of the beams. It is recommended to always subtract it as it is not yet calculated in a selfconsistent manner. The ibeco switch also acts on the "wire" elements 6.7 in the same way as on the beam-beam elements. It subtracts the closed orbit introduced by the wire if ibeco=1 and applies it if ibeco=0.

`ibtyp`	integer	Switch (0 = off; 1 = on) to use the fast beam–beam algorithms developed in collaboration with G.A. Erskine and E. McIntosh. The linear optics are calculated with "exact" beam–beam kicks.
`lhc`	integer	For the LHC with its anti-symmetric IR the separation of the beams in one plane can be calculated by the β-function of the other plane. For flat beams (not anti-symmetric optics) the separation can be loaded from the `fort.2` file. (0 = off; 1 = anti-symmetric; 2 = load from file).
`ibbc`	integer	Linear coupling considered in 4D and 6D (0 = off; 1 = on).
`name`		Name of 6D beam–beam element. Beam–beam elements that do not appear will be treated as 4D kicks.
`ibsix`	integer	Number of slices of the 6D beam–beam element. If `ibsix` is set to 0 this element is treated as a 4D element.
`xang`	float	Half crossing angle (angle the between the trajectories of the two beams) at this particular element [rad].
`xplane`	float	Crossing plane angle [rad].
`xstr`	float	Angle of the position of the slices in the boosted frame [rad] (i.e. $X = Z\sin(xstr)\cos(xplane)$, $Y = Z\sin(xstr)\cos(xplane)$). In absence of crabbing user should make sure `xstr=xang`; in case the `xstr` flag is not set then `xstr=xang` is assumed and a warning is printed (since version 4.5.45).

EXPERT format

First line: `EXPERT`

Second line: `partnum emitnx emitny sigz sige ibeco ibtyp lhc ibbc`

Further lines 4D BB lens (1 line per element):

`name ibsix` $\Sigma_{x,x}$ $\Sigma_{y,y}$ `h-sep v-sep strength-ratio` $\Sigma_{x,y}$

6D BB lens (3 lines per element):

`name ibsix xang xplane h-sep v-sep`

$\Sigma_{x,x}$ $\Sigma_{x,xp}$ $\Sigma_{xp,xp}$ $\Sigma_{y,y}$ $\Sigma_{y,yp}$

$\Sigma_{yp,yp}$ $\Sigma_{x,y}$ $\Sigma_{x,py}$ $\Sigma_{xp,y}$ $\Sigma_{xp,yp}$ `strength-ratio`

Some parameters are new or defined in a different way:

`lhc`	integer	This parameter is kept for now only for RHIC studies when equal to 9.
`name`		Name of the beam–beam element.
`ibsix`	integer	Number of slices of the 6D beam–beam element. If `ibsix` is set to 0, this element is treated as a 4D element. If `ibsix` is larger or equal 1, this element is treated as a 6D element.
Σ_{xx}	float	Horizontal σ for the strong beam [mm^2].
Σ_{yy}	float	Vorizontal σ for the strong beam [mm^2].

Σ_{xy}	float	Coupled σ for the strong beam [mm^2]. Optional, used only if `ibbc=1`.
`h-sep`	float	Horizontal beam–beam separation [mm]
`v-sep`	float	Vertical beam–beam separation [mm]
`strength-ratio`	float	Strength ratio with respect to the nominal beam–beam kick strength. This is useful to allow for splitting one beam–beam kick into several (longitudinally close by) kicks.
$\Sigma_{i,j}$	float	Second order momenta matrix for the strong beam, in units of mm and mrad. For example Σ_{xxp} in [mm mrad]

Conversion from traditional to EXPERT format

An automatic converter from the "traditional" input block to the new "expert" format is built into SixTrack; every time a non-`EXPERT` input block is encountered, a conversion is printed to the standard output. Therefore, all the user needs to do is to run SixTrack (number of turns does not matter) on an input file that should be converted, and follow the instructions which are printed at the beginning of the program output.

Remarks

These beam–beam elements have to appear in the single element list (5.1.2) (type 20). If the "traditional" option is used in the `BEAM` block, the listing in the single element list must contain their horizontal and vertical beam–beam separations (see 5.1.7).

Sign Convention

Some clarifications regarding the sign convention used for the separation and crossing angle variables.

Separations:

1. The separation is added to the transverse coordinates of each particles just before the beam-beam subroutines (see Fig. 6.4).

$$\tilde{x}_i = x_i + \mathrm{sep}_x - \mathrm{CO}_x$$
$$\tilde{y}_i = y_i + \mathrm{sep}_y - \mathrm{CO}_y$$

2. Lorentz boost applied to the updated coordinates.

3. The separation used for the actual beam-beam kick ($\mathrm{sep}_{x,y,kick}$) is the difference between the centroid of the strong slice ($\mathrm{X}^\dagger$,$\mathrm{Y}^\dagger$) and the each particle (x_i,y_i).

4. Antiboost to return to accelerator frame.

5. The separation is removed and the closed orbit is added back. Tracking continues.

$$\tilde{x}_i = x_i - \mathrm{sep}_x + \mathrm{CO}_x$$
$$\tilde{y}_i = y_i - \mathrm{sep}_y + \mathrm{CO}_y$$

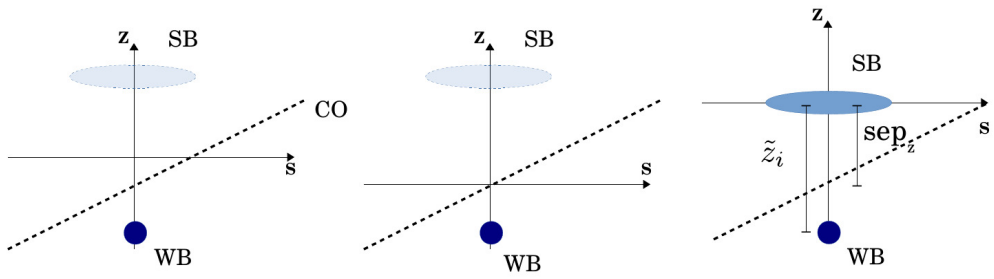

Figure 6.4: Coordinate manipulation taking into consideration the beam-beam lens separation as stated in point 1 of the separation sign convention.

Crossing angles:

1. The closed orbit is removed just before the beam-beam subroutines.

$$\tilde{x}'_i = x'_i - \mathrm{CO}_{x'}$$
$$\tilde{y}'_i = y'_i - \mathrm{CO}_{y'}$$

2. Lorentz boost applied to the updated coordinates.
3. Apply Synchro-Betatron Mapping.
4. Antiboost to return to accelerator frame.
5. The closed orbit is added back. Tracking continues.

$$\tilde{x}'_i = x'_i + \mathrm{CO}_{x'}$$
$$\tilde{y}'_i = y'_i + \mathrm{CO}_{y'}$$

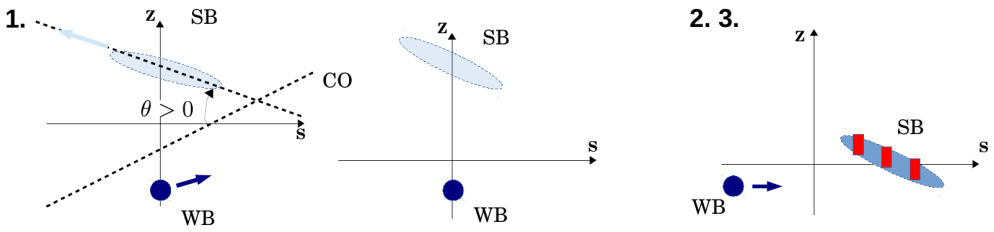

Figure 6.5: Coordinate manipulation to move from the accelerator frame to a head-on collision frame (Figures left and center). A positive crossing angle is considered as shown in the left figure. Then Lorentz boost and Synchro-Betatron Mapping are applied (right).

6.7 Wire

The wire block serves for reading in the input parameters for the wire. Each wire also needs to be added as single element in the list of single elements.

Keyword WIRE
Data lines Variable
Format `name flag_co current int_length phys_length disp_x disp_y tilt_x tilt_y`
A description of the input parameters for the wire is given in Table 6.11.

Table 6.11: Input parameters for the WIRE block.

Arguments	Unit	Description
name	-	Name of wire. Must be the same as in list of single elements.
flag_co	-	flag to define the displacement of the wire in respect to the closed orbit or x=y=0. For flag_co=+1 disp_* is the distance between x=y=0 and the wire. For flag_co=-1 disp_* is the distance between the closed orbit and the wire.
current	A	wire current
int_length	m	integrated length of the wire
phys_length	m	physical length of the wire
disp_x	mm	hor. displacement of the wire
disp_y	mm	vert. displacement of the wire
tilt_x	degrees	hor. tilt of the wire $-90 < tilt\_x < 90$ (uses same defintion as DISP block)
tilt_y	degrees	vert. tilt of the wire $-90 < tilt\_y < 90$ (uses same defintion as DISP block)

Remarks

The user has to check that the wires defined in the WIRE block are also defined in the list of single elements and vice versa. All parameters, except for the type (type 15), are ignored in the single element definition and the execution is aborted if the parameters are non-zero. In addition to the parameters defined in the WIRE block, the ibeco parameter in the BEAM block (see Section 6.6) imposes the same behavior on the wire as for beam-beam. Explicitly, the closed orbit introduced by the wire is subtracted if ibeco=1 and not subtracted if ibeco=0.

Example:

In the following we give some examples for wire definitions. This example defines two wires wire_1 and wire_2.
The input block in fort.3 is given by:

```
WIRE
wire_1   -1  +98.9   2.0  1.0   10.0   10.0     1.1      1.1
wire_2   -1  +98.9   2.0  1.0   10.0   10.0     0.0      0.0
NEXT
```

The single and structure element definition in fort.2 is given by:

```
SINGLE ELEMENTS------------------------------------------------------------
...
wire_1   15 0.000000000e+00 0.000000000e+00  0.000000000e+00  0.000000000e+00  0.000000000e+00  0.000000000e+00
wire_2   15 0.000000000e+00 0.000000000e+00  0.000000000e+00  0.000000000e+00  0.000000000e+00  0.000000000e+00
...
STRUCTURE INPUT------------------------------------------------------------
...
BLOC56           wire_1             wire_2
...
```

Note that all parameters except for the type have to be set to 0 in the single element definition.

6.8 "Phase Trombone" Element

The linear "phase trombone" allows for the introduction of an arbitrary transfer matrix. It can be used to introduce a change in the transverse phases without spoiling the linear optics of the rest of the machine, i.e. the Twiss parameters are the same at entrance and exit of the element. Note that it is up to the user to construct the matrix. The coordinates used as inputs are: x, p_x, y ,p_y, σ, p_σ.

Keyword	`TROM`
Data lines	1 line with name and then in blocks of 14 lines with 3 entries each.
Format	First line: `name` Second line: x, p_x, y Third line: p_y, σ, p_σ Fourth util 15^{th}: `M(6 × 6)` matrix

`name`	char	May contain up to 48 characters.
`cx, cx', cy, cy', cz, cz'`	floats	6D closed orbit to be added to the coordinates.
`M(6 × 6)`	floats	6×6 matrix elements.

Remarks

The user has to make sure that the above stated conditions are fulfilled. When using the *mad_6t* [15] converter from MAD-X to SixTrack, this is guaranteed to be the case. Note also that the crossterms between the transverse plains are not considered for the time being.

6.9 Beam Distribution EXchange (BDEX)

The Beam Distribution EXchange allows an external program to read and modify the beam distribution in SixTrack. This can be used for tracking part of the machine in an external program, for example for including physics processes that are normally not available in SixTrack. Another possible use is for multi-bunch tracking, i.e. with an external program "swapping" the bunch at a some point in the ring. This would be useful for studying (for example) beam loading, where the external program would read the position of a bunch in the cavity, use that to compute an update of the cavity voltage (which can be sent to SixTrack using DYNK FUN PIPE), swap the bunch with another one and track that to the cavity (still at "physics turn" 1, but "SixTrack turn" 2) etc.

Please note that `BDEX` is currently not supported in the checkpoint/restart version or in the collimation version. Including `BDEX` in one of these versions results in a run-time error.

Keyword	`BDEX`
Data lines	Variable
Format	There are three types of statements possible in a `BDEX` block, listed below.

ELEM `ELEM chanName elemName action`

This associates a given element with an already existing channel and an action. The element must appear in the SINGLE ELEMENT block, and be of type 0 (marker). The action indicates what should be done with the particle distribution when it reaches this element. Currently, the only allowed action is "1", which means "particle exchange", i.e. output the beam distribution and read back another one at the same point.

CHAN `CHAN chanName chanType ...`

This creates a new channel through which the `BDEX` can communicate. Currently, the only implemented `chanType` is `PIPE`, however `TCPIP` is also foreseen.

For the `PIPE` type, the statement including arguments is `CHAN PIPE inPipeName outPipeName format fileUnit`. This uses a pair of UNIX FIFOs, through which SixTrack can communicate with an external program. When the channel is used, it sends a message on the outpipe, then waits for a reply with the new distribution over the inPipe. The `format` is an integer used to indicate the output/input format, and is currently unused. The `fileUnit` is the Fortran unit number that should be used to open the inPipe. The `outPipe` is opened on the next unit, so both units fileUnit and fileUnit+1 must be free.

DEBU

This statement switches on extra "debugging" output from `BDEX`. This can be useful if debugging the code or if debugging the input.

6.9.1 Communication protocols

The communication protocols used by the different channel types are listed below:

PIPE communication protocol

When a pair of pipes are first initialized, a header "`BDEX-PIPE !******************!`" is written to the output pipe. Then, when the tracking reaches an element which is marked as active for this channel, it writes another header like "`BDEX TURN= 1 BEZ=ip1 I= 1 NAPX= 64`", where `TURN` is the number of the current SixTrack turn, `BEZ` the name of the SINGLE ELEMENT, `I` the index of the STRUCTURE ELEMENT, and `NAPX` the number of particles to be written out. After this follows `NAPX` lines with the particle information (one per particle), each line of the format `xv(1,j) yv(1,j) xv(2,j) yv(2,j) sigmv(j) ejv(j) ejfv(j) rvv(j) dpsv(j) oidpsv(j) dpsv1(j),nlostp(j)` where all but the last floating point numbers, the last being an integer. Finally, it writes "`BDEX WAITING...`" to the output pipe, and waits for data on the input pipe.

The first line expected on the input pipe should be an integer containing the number of particles to write back. If this integer is -1, the current particle distribution is kept. Otherwise, a number of lines of the same format as with the output is expected. After reading in the expected number of particles, the string "`BDEX TRACKING...`" is written to the output pipe and tracking is resumed.

TCPIP communication protocol

TCPIP is not yet implemented, as it would require an external library. The FLUKA version implements this, we should make sure that we are compatible with their requirements and ideally their protocol.

6.10 Electron Lens

The electron lens module serves for reading in the input parameters of electron lenses. Each e-lens also needs to be added as single element in the list of single elements. Currently, the ideal electron lens is implemented, i.e.

- no errors in the e–beam distribution;
- the kick is only due to the transverse components of the electric and magnetic fields generated by the electron beam;
- electrons travel all parallel to the e-lens axis;
- no longitudinal component of the kick is considered.

Keyword `ELEN`
Data lines Variable
Format `name type theta_r2 r2 r1 offset_x offset_y (L I E_k)`
The last three parameters are optional, but if specified, all of them must be present; they allow to re-compute the angle at `r2` `theta_r2`, taking into account also the beam energy.
Additional parameters specific to the electron distributions (see later) must be specified before the optional ones.

Currently, three types of electron beam profiles on the transverse plane (i.e. in the xy-plane) are supported:

`UNIFORM` e–beam with constant density of electrons.
`GAUSSIAN` e–beam with a radial Gaussian profile.
`RADIAL` e–beam follows a radial distribution as described in a plain ASCII file.

Table 6.12: Input parameters for `ELEN` block.

Type Name	Arguments	Unit	Description
Valid for all types			
	`name`	–	Name of e-lens. Must be the same as that in the list of single elements.
	`type`	–	Type of e-lens - either `UNIFORM` or `GAUSSIAN`.
	`theta_r2`	mrad	Kick[1] received at $r = r_2$ where r_2 is the outer radius of the e-lens.
	`r2`	mm	Outer radius of the e-lens.
	`r1`	mm	Inner radius of the e-lens.
	`offset_x`	mm	Horizontal offset of the e-lens.
	`offset_y`	mm	Vertical offset of the e-lens.
Specific to `GAUSSIAN` type			
	`sig_el`	mm	σ of the electron beam.

(The table continues on the next page)

[1] Total, *not* per charge.

Type Name	Arguments	Unit	Description
Specific to `RADIAL` type			
	`filename`		filename of the radial profile. Please see Sec. 6.10.1 for proper formatting of the file. The profile is read from the file, radially integrated and normalised to the total current. Hence, the kick at `r2` is given by the value of `theta_r2` (or by the optional parameters) and not by integration of the radial profile. The domain of the profile must be larger than the domain identified by R_1 and R_2.
Optional arguments (all types)			
	`L`	m	Length of the e-lens. Must be positive.
	`I`	A	Electron current of the e-lens. Must be non–zero. If negative, the electron beam is considered travelling in the direction opposite to that of the main beam.
	`E_k`	keV	kinetic energy of electrons in the e-lens. Must be positive.

The spatial charge density of all profiles is defined between `r1` and `r2`:

$$\rho(r) = \begin{cases} 0 & \text{if } r \leq r_1 \\ f(r) & \text{if } r_1 < r < r_2 \\ 0 & \text{if } r_2 \leq r \end{cases} \tag{6.1}$$

For the time being, `r_1` cannot be `0`; hence, the electron lens can only be hollow.

Remarks

1. The user has to check that the e-lens defined in the `ELEN` block is also defined in the list of single elements and vice versa. All parameters in the single element definition except the type (type `29`) are ignored.

2. The current implementation (ideal e-lenses) has no explicit energy–dependency, except for the user defined parameter `theta_r2` (see [16]), in case the user specifies `L`, `E_k` and `I`. In fact, in this case, `theta_r2` is re-calculated after input parsing, taking into account the beam energy. This parameter is re-computed also during energy ramping by the `DYNK` module.

3. The implementation is fully chromatic and hence compatible with ion tracking and tracking of species different from the synchronous one only if the three main optional parameters (i.e. length, kinetic energy of electrons and current) are specified by the user. In fact, only in this case, the code knows the value of the normalised relativistic speed β_e;

4. In case of energy ramping, if not `DYNK`-ed, the kick stays constant, as if the e-lens was ramped in beam current in synch with the rest of the machine. On the contrary, if the user specifies the three main optional parameters (i.e. length, kinetic energy of electrons and current), if not `DYNK`-ed, the kick is recomputed at any update of the reference energy, as if the e-lens was not ramped. Hence, it is responsibility of the user to define the way the e-lens is ramped with beam energy.

5. Note that if the user specifies `theta_r2` with DYNK, then, starting with the first turn where it is set via `DYNK`, `theta_R2` is no longer ramped with the energy, as if the elens was specified using only this parameter in `fort.3`. However note that the chromatic behaviour, which requires `E_k`, is still handled in a physically correct way.

Examples

In the following we give an example of e-lens definition. The example defines two electron lenses `hel1` and `hel2` for cleaning purposes; the former has a uniform electron density whereas the latter has an electron beam with a Gaussian profile with σ=1.1547 mm. While the former has an explicit definition of the kick at `r2`, the latter triggers the re-computation of the kick, given the length of the electron lens of 4 m, the electron current of 2 A (opposite wrt the main beam) and the electron kinetic energy of 1 keV. The input block in `fort.3` is then given by:

```
ELEN
hel1 UNIFORM  4.920e-03 6.928 4.619 1.1547 2.3093
hel2 GAUSSIAN 4.920e-03 6.928 4.619 1.1547 2.3093 1.1547 4 -2 1
NEXT
```

The single and structure element definition in `fort.2` are given by:

```
SINGLE ELEMENTS------------------------------------------------------------
...
hel1           29  0.000000000e+00  0.000000000e+00  0.000000000e+00  0.000000000e+00  0.000000000e+00  0.000000000e+00
hel2           29  0.000000000e+00  0.000000000e+00  0.000000000e+00  0.000000000e+00  0.000000000e+00  0.000000000e+00
...
STRUCTURE INPUT------------------------------------------------------------
...
BLOC56           hel1             hel2
...
```

Note: All parameters except for the type are set to `0` in the single element definition.

6.10.1 Format of Radial Profile

The file format of the radial profile is

Data lines	Variable
Comment character	`#` (only as first char of the line)
Format	`R` (mm) `J` (A cm^{-2})

Formatting rules:

- every non-commented line is taken as a point in the profile;
- values of the radius must be in *increasing* order. Otherwise, the program stops.

An example of radial profile is contained in each test of e-lenses.

6.11 Scattering

Note: The PYTHIA implementation in this module should be considered experimental! It is not yet fully tested for correct physics.

The `SCATTER` module is a framework for scattering particles through Monte Carlo processes at various points in the machine.

The Scatter Module currently supports interfacing with PYTHIA8 to generate scattering events. In order to use PYTHIA as an event generator, SixTrack must be built with the `PYTHIA` flag. If this is enabled, the `PYTHIA` block becomes available as an input block in `fort.3`. The `PYTHIA` block provides

access to the SoftQCD module of PYTHIA8 through a minimal interface. See Section 7.1 for how to use this block.

In addition, the Scatter Module has an internal generator for elastic scattering events that does not rely on any external tools.

The Scatter Module uses the internal random number generator in SixTrack, both with and without PYTHIA integration. It therefore requires the `RANDOM NUMBERS` block to be present. See Section 4.6.

Keyword	`SCAT`
Data lines	Variable
Format	Name/value sets, see below.

6.11.1 Module Flags

These are the main settings and flags to control the general flow of the Scatter Module, and allows for additional debugging options to ensure the simulation is set up correctly. Especially when scattering against a realistic Beam 2, it can be tricky to ensure that the beams actually interact. The additional output files below are intended to help verifying that they do.

Module Debugging `DEBUG`

This statement switches on extra "debugging" output from `SCATTER`. This can be useful if debugging the code or if debugging the input.

Particle Losses `LOSSES`

This statement switches on particle losses in the scatter module. This flag is propagated to the `PYTHIA` module (see Section 7.1), allowing for selecting double diffractive and non-diffractive processes. In addition, this will allow losses with single diffractive when the tracked prticle is destroyed and the target particle survives. If losses are disabled, the event generator will only accept events where the tracked particle survives.

Particle losses are off by default.

Beam 2 Sample Log `WRITE_PLOG`

If the `PYTHIA` block has the `REALBEAM` flag enabled (see Section 7.1), the Scatter Module will generate a random three-momentum vector of Beam 2 particles that is then fed to the PYTHIA event generator. This flag enables the logging of these generated events to the `scatter_momentum.dat` file.

Density Dump `DENSITY_DUMP element_name turn`

This setting allows for a dump of the target PDF in a 500 by 500 matrix to the file `scatter_pdf.dat`, and a dump of the target density seen by each particle to the file `scatter_density.dat`. This is intended as a debug tool to check that Beam 1 and Beam 2 are actually interacting. The dump can only be used once, on a given element, on a given turn.

Beam Dump `BEAM_DUMP element_name turn`

This setting dumps a particle state file (see Section 3.4) immediately before and after the scatter point. This information is written to the files `scatter_beam_before.dat` and `scatter_beam_after.dat`. This is intended as a debug tool to check the effect of the scatter element on the beam. The dump can only be used once, on a given element, on a given turn.

6.11.2 Element, Target and Process Definitions

This section describes the keywords needed to set up the actual scattering points. The target density profile for scattering, and the scattering event generators, are defined separately. The profiles and the generators are then applied to the scattering elements (of type 40) in the lattice. This ensures that the input block is reasonably human readable, and avoids redundant configurations.

Note:

If using very low scattering probabilities, ensure that the probability is within the scope of the resolution of the random number generator. To select whether a scattering event has occured, the probability is evaluated against a uniformly distributed random number between 0 and 1. However, the default random number generator has a resolution of just below 2^{31}, meaning the smallest number that can be sampled is 4.66×10^{-10}. Scattering probabilities below $\approx 10^{-8}$ should be avoided.

The Scatter Module allows for statistical scaling of scattering probabilities. In the `scatter_log.dat` file, there is a column keeping track of the statistical weight through multiple scattering of the same particle.

Beam 2 Emittance `BEAM2_EMIT emitX emitY`

If the target profile is a model of Beam 2, defined by Twiss parameters, this keyword is required to set the x and y normalised emittance. The emittance is in units of μm.

Beam 2 Length `BEAM2_LEN sigmaZ`

If the target profile is a model of Beam 2, defined by Twiss parameters, this keyword is required to set the beam length, assuming a Gaussian longitudinal profile. The length is in units of mm.

Density Profiles `PRO name type (arguments)`

This keyword defines a profile, that is a density profile and its general properties. This is the target the tracked particles are colliding with.

Several different types are available:

`PRO name FIXED density[targets/cm`^{2}`]`

A uniform density profile with infinte extent.

`PRO name GAUSS1 beamtot[particles] sigmaX[mm] sigmaY[mm] offsetX[mm] offsetY[mm]`

The simple, round, Gaussian target profile with a given number of particles. The `GAUSS1` profile parameters are given by

$$\rho(x,y) = \frac{N_{\text{tot}}}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{(x-\mu_x)^2}{2\sigma_x^2}\right) \exp\left(-\frac{(y-\mu_y)^2}{2\sigma_y^2}\right). \tag{6.2}$$

`REFBEAM density [crossX crossY]|[MIRROR]`

Each particle is collided against the reference particle. The `density` determins the scattering probability, and is a fixed number. Optionally, the x and y crossing angles can be added, or the `MIRROR` flag used to take the same crossing angles as for Beam 1. If neither is specified, the crossing angle is taken to be 0. The crossing angle is only relevant when the `PYTHIA` option `REALBEAM` is enabled.

`UNCORRBEAM nbeam betaX betaY alphaX alphaY crossX crossY [offsetX offsetY] | MIRROR`

The target is a Beam 2 described by a set of uncorrelated Twiss parameters, and a total beam particle count. The target beam is effectively a "thin" beam, in that it appears as an infinitely thin foil at the centre of the interaction point. The crossing angle is taken into account using an effective sigma, as is commonly done for luminosity calculations. The probability is then corrected using the particle time offset and assuming a longitudinal Gaussian distribution. If the keyword `MIRROR` is specified, the Twiss, offset and crossing angle is taken from Beam 1.

Event Generator `GEN name type [arguments]`

The generator block takes a name and a generator type input, followed by the parameters for the generator type.

`GEN name PPBEAMELASTIC a b1 b2 phi tmin crossSection`

Takes five or six input arguments, and generates the probability distribution given by

$$g(t) = \frac{1}{a_1^2}\frac{d\sigma}{dt} = e^{-b_1 t} + 2ae^{-(b_1+b_2)t/2}\cos\phi + a^2 e^{-b_2 t}, \tag{6.3}$$

where the first expression is a soft scatter data fit, the third expression a hard scatter fit, and the second expression is the interference. $a = a_2/a_1$ is the amplitudes of the expressions. These are combined into the first four input arguments a, b_1, b_2, and ϕ, as well as t_{min} which provides a cut-off limit. The sixth argument defines a fixed cross section for the scattering probability in units of millibarn. If all elements have `ratio` set to a fixed value, the cross section set here is ignored.

Input example with values for a fit to 13 TeV LHC.

```
GEN  sc_thin     PPBEAMELASTIC 0.046 18.52 4.601 2.647 0.0 30.0
```

`GEN name PYTHIA crossSection`

If SixTrack is built with PYTHIA support, it is available as a generator, taking only the total cross section as input parameter. The generator itself is configured through the `PYTHIA` block. See Section 7.1.

Scatter Elements `ELEM elemname profile ratio scaling gen1 ... genN`

This statements associates a `PRO`file and one or more `GEN`erators with a SINGLE ELEMENT. The element must be of type 40.

The `ratio` argument (string/float) can be used to set a fixed scattering probability at the element. Alternatively, the field can be set to "auto", in which case the scattering probability is calculated from the cross section and density profile.

The `scaling` argument (float) is used to scale the probability of an interaction when using cross section to claculate the scattering probability. This field can be controlled through DYNK (see Section 6.5), for example in order to scale only at one specified turn, or to switch off and on scattering at the element by scaling it by 0 and 1 respectively.

The profile, generator(s), and single elements are referenced through their names. The generators and profile must be defined above the `ELEM` defintiion where they are used. When using multiple generators, a branching ratio between them is calculated, and sampled by the scatter routine.

6.12 Collimation

Collimation and beam cleaning studies are carried out with the well established SixTrack code, extended for tracking large numbers of halo particles, and to take into account halo interaction with arbitrarily placed collimators. Particles are transported through the lattice element by element and their phase space coordinates are transformed according to the type of element. When a particle hits a collimator jaw, it is randomly scattered through matter. The effect of collimator scattering is modeled using COLLTRACK/K2 [42, 43] routines.

The main characteristics of the SixTrack used for collimations studies are:

- Proton scattering in various collimator materials, including:
  - Multiple Coulomb scattering,
  - Ionization of the collimator material,
  - Elastic proton-proton (pp) scattering, and inelastic diffractive pp scattering (single diffractive scattering),
  - Inelastic proton-nucleon scattering,
  - Elastic and inelastic proton-nucleus scattering,
  - Rutherford scattering.
- Various types of halo and possibility of including diffusion.
- Tracking of large particle ensembles (106 protons) over hundreds of turns.
- Multiple imperfections on the beam and the collimator properties (setting errors, tilts, orbit, beta beat,)

Input parameters are divided in 3 files:

- `fort.2` (generated by MAD-X), defining the lattice of the machine without magnetic field errors.
- Collimator database file, containing the details of collimators geometry, material, settings (opening)
- `fort.3` (modified from the one used for SixTrack without collimation), including tracking parameters (number of particles and turns), type of beam, type of halo.

MAD-X is used to generate the LHC lattice, the optics and eventual orbit and focusing errors. BeamLossPattern: Implementation of the LHC aperture model with analysis of loss locations for all tracked protons.

SixTrack for collimation studies tracks particles populating an halo (with typically $\sigma \geq 6$) throughout the LHC lattice (as defined by the MAD-X output file `fc.2`). The halo is represented in the phase diagram Y (offset from beam orbit axes), Y' (angle with respect to the beam orbit axes) in Figure 6.6.

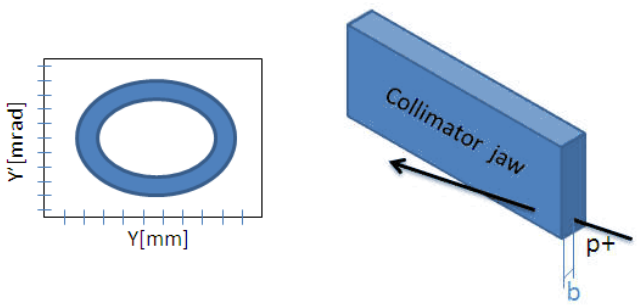

Figure 6.6: **Left:** Halo particles in the phase diagram. **Right:** Impact Parameter.

One defines the Impact Parameter b (see Figure 6.6) as the transverse offset between the jaw surface and the impact point. b typically equals $1\mu m$ at the first impact.

These changes implied to modify/add some input files:

- the generic SixTrack parameter file `fort.3` now has a new block for collimation parameters,
- a separate collimator database file is now mandatory if collimation studies are to be done.

This database including the name, length, orientation and material of every single collimator of the ring (1 file per Collimation System Phase though). Apart from these two, the other required SixTrack files are produced via MAD-X and its conversion module.

6.12.1 Collimation Input Block

The collimation module is controlled by the `COLL` block in `fort.3`.

Keyword	`COLL`
Data lines	Either 17 lines (old format), or free (new format)
Format	See below.

Format Description:

From SixTrack version 5.2.5 and on, the main format of the (COLL) block is a set of keyword/value lines.

For backwards compatibility, the old format is still supported, and it is described in Section 6.12.4. When encountering the `COLL` block, the parser will check if the first line of the block is a single value, and if it is, it will conclude that the block is in the old format. The old block format consists of 17 lines of varying number of values, and the full description is laid out in Table 6.14.

Note that the old and new format cannot be inter-mixed. The old format is parsed based on the line number, and must follow exactly the format described in Table 6.14.

The keyword/value formated block is described below. The keywords mostly correspond to the keywords listed in Table 6.14.

Do Collimation (Required) `DO_COLL true|false`

The collimation module can be switched on and off in its entirety with this flag, without having to comment in and out the entire block. When the flag is off, the collimation routines are skipped in tracking, and the additional memory needed for collimation is not allocated. This entry is required for collimation to be enabled, but if ommitted, defaults to `false`.

Collimator Database (Required) `COLLDB filename`

The filename of the database of collimators. This entry is required. The format of the database is described in Section 6.12.2. It is also possible to load the old style database format. If so, a converted database is written to the simulation folder with an added file extension `.new`.

Beam Energy (Required) `ENERGY energy[MeV]`

The energy of the beam to be tracked. This is independent of the energy specified in other blocks, and is a required value.

Emittance (Required) `EMIT eX_d eY_d eX_g eY_g`

Specifies the normalised emittance for the distribution [eX_d, eY_d], and for the collimator gaps [eX_g, eY_g]. These are required.

Random Seed `SEED seed`

Set the seed for the random number generator. If set to 0 (default) a new seed will be selected for each run.

Distribution Generator Type `DIST_TYPE 0-6`

The type of the distribution generator to use. Setting this to larger than 0 will overwrite the distribution generated by the initialisation routines in SixTrack, including the distribution read from the `DIST` block (see Section 4.7). This feature will be deprecated in the future, and the functionality taken over by an extended `DIST` module. The different options are listed below. The default value is 0.

0. The internal collimation distribution generator is disabled and the general SixTrack distribution is used instead (default).
1. Distribution in the plane for which the parameters are specified ONLY: flat distribution in the selected plane between $A_x \pm \delta A_x$ (horizontal) or $A_y \pm \delta A_y$ (vertical). The amplitude in the other plane is zero. The parameters must be provided with the `DIST_PARAM` keyword. Values 1 to 4 are used.
2. Distribution in the plane for which the parameters are specified + a Gaussian distribution cut at 3σ in the other plane. The parameters must be provided with the `DIST_PARAM` keyword. Values 1 to 4 are used.
3. Distribution in the plane for which the parameters are specified + a Gaussian distribution cut at 3σ in the other plane. The parameters must be provided with the `DIST_PARAM` keyword. Values 1 to 4 are used. In addition an energy spread is given as value 5, and a longitudinal component given by value 6.
4. Reads an external file that contains the beam distribution to be tracked. The file is specified by `DIST_FILE`.
5. Radial transverse distribution of radius A_r. This corresponds to a flat distribution both in the horizontal and vertical planes between $A_x \pm \delta A_x$ and $A_y \pm \delta A_y$, with $A_x = A_y = A_r/\ddot{O}2$.
6. Reads a normalised distribution from an external file. The file is specified by `DIST_FILE`.

Distribution Generator Parameters `DIST_PARAM` A_x δA_x A_y δA_y `[enerror bunchlen]`

The parameters for the distribution generator `DIST_TYPE`. The required values are determined by the type selected. All values default to 0.

Recomended values are 3.06×10^{-4} at 450 GeV and 1.129×10^{-4} at 7 TeV for energy error (`enerror`), and 11.24 cm at 450 GeV and 7.55 cm at 7TeV for bunch length (`bunchlen`).

Distribution Generator Input File `DIST_FILE filename`

Name of the distribution file to be read in for distribution type 4 or 6.

Special Radial Distribution `DO_RADIAL true|false amp smear`

Alternative radial beam distribution where `amp` is the amplitude of the beam in numbers of radial sigmas, and `smear` is the smear of the beam in radial sigma. Setting this to true will override `DIST_TYPE`.

This distribution is equivalent to `DIST_TYPE = 1` with the `DIST_PARAM` values:

$$A_x = A_y = \texttt{amp}/\sqrt{2}$$
$$\delta A_x = \delta A_y = \texttt{smear}/\sqrt{2}$$

.

Special Radial Distribution `DO_MINGAP true|false`

If `true`, the particle distribution is generated at the collimator with the smallest gap (to be used with sheet/pencil beam). Default value is `false`.

Do Change Collimator Gaps `DO_NSIG true|false`

If `true`, the collimator gaps specified in the imput block by keyword `NSIG_FAM` are used instead of those in the collimator database file. Default value is `false`.

Set Collimator Gaps `NSIG_FAM name collgap`

Family name and collimator gap in units of sigma for each collimator family specified below. This is only applied to the collimators themselves if the `DO_NSIG` is true.

The collimator family name is extracted from the collimator name in the collimator database. The default collimator families are listed below. A new collimation database format will be added in the near future where the families can be specified independently in the database, and not locked to the naming convention for the LHC.

The default value is the sigma gap specified in the collimator database.

`tcp3`	The primary collimator in IR3.
`tcsg3`	The secondary graphite collimator in IR3.
`tcsm3`	The secondary metallic collimator in IR3.
`tcla3`	The active absorbers in IR3.
`tcp7`	The primary collimators in IR7.
`tcsg7`	The secondary graphite collimator in IR7.
`tcsm7`	The secondary metallic collimator in IR7.
`tcla7`	The active absorbers collimator in IR7.
`tclp`	The physics debris collimator.
`tcli`	The absorbers for injection protection.
`tcdq`	The beam dump protection collimator[2].
`tcstcdq`	The secondary collimator dedicated to beam dump.
`tdi`	The injection protection collimator.
`tcth1`	The horizontal tertiary collimator in IR1.
`tcth2`	The horizontal tertiary collimator in IR2.
`tcth5`	The horizontal tertiary collimator in IR5.

tcth8	The horizontal tertiary collimator in IR8.
tctv1	The vertical tertiary collimator in IR1.
tctv2	The vertical tertiary collimator in IR2.
tctv5	The vertical tertiary collimator in IR5.
tctv8	The vertical tertiary collimator in IR8.
tcxrp	The Roman Pots[3].
tcryo	The collimators in the DS regions.

Jaw Slicing

This feature has been removed from the main collimation block. The collimators or collimator families that are to be sliced are now set in the collimator database file. See Section 6.12.2.

One-Sided Collimators

This feature has been removed from the main collimation block. The collimators or collimator families that are to be treated as one-sided are now set in the collimator database file. See Section 6.12.2.

Beta-Beating `BETA_BEAT xbeat xbeat_phase ybeat ybeat_phase`

In case of beta-beating:

xbeat	float	Offset in x for the computation of collimator.
xbeat_phase	float	Phase offset in x for the computation of collimator.
ybeat	float	Offset in y for the computation of collimator.
ybeat_phase	float	Phase offset in y for the computation of collimator.

These values default to 0.

Pencil Beam `PENCIL ipencil offset rmsx rmsy distr`

Resets original distribution to pencil beam distribution on the collimator database ID specified by `ipencil`. The formats are described below. Setting `ipencil` to 0, disables this feature. These values default to 0.

`distr = 0`
Pencil Beam Distribution
	`offset = 0`
	`rmsx = 0`
	`rmsy = 0`

`distr = 0` Rectangular Distribution	`offset` = center of rectangle `rmsx` = spread of impact parameter (uniform) `rmsy` = spread parallel to jaw surface
`distr = 1` Gaussian Distribution	`offset` = mean of Gaussian distribution `rmsx` = spread of impact parameter (Gaussian) `rmsy` = spread parallel to jaw surface (Gaussian)
`distr = 2` Half Gaussian Distribution	`offset` = mean of Gaussian distribution `rmsx` = spread of impact parameter (uniform) `rmsy` = spread parallel to jaw surface (Gaussian)

Note: The distribution with `distr = 2` is used when wanting to simulate the loss of a magnet.

[2]One-sided collimators (only positive x)
[3]One-sided collimators (only positive x)

Dedicated Collimator Study `DO_SELECT true|false name`

Do a dedicated study of selected collimator. This value defaults to `false`.

Nominal Beta `DO_NOMINAL true|false`

Switches on or off the use of design β values of collimators. This value defaults to `false`.

Emittance Drift `EMIT_DRIFT driftx drifty`

Apply an emittance drift in the x and y direction, respectively. These values default to 0.

Alignment Errors `ALIGNERR_[PRIM|SEC] rms_tilt sys_tilt rms_offset sys_offset`

Apply errors to primary or secondary collimator tilt and offset. These values default to 0.

`rms_tilt`	float	RMS value of tilt to apply.
`sys_tilt`	float	Systematic value of tilt to apply.
`rms_offset`	float	RMS value of offset to apply.
`sys_offset`	float	Systematic value of offset to apply.

Alignment Errors to Gaps `ALIGNERR_GAP rmserror_gap`

The RMS error of collimator gap. These values default to 0.

Alignment Errors Random Seed `ALIGNERR_SEED seed`

The random number seed to use for alignemnt errors. This value defaults to 0.

Systilt `SYSTILT_ANTI true|false`

Deduce `SYSTILT` to `RMSTILT` instead of adding. This value defaults to `false`.

Cut Sigmas `SIGSECUT sigma_xy sigma_r`

Sigma cuts for `tracks2.dat` controlled by the `WRITE_TRACKS` flag. Cut in square sigmas x/y for saving particles (e.g. 64 for a cut at 8 σ_x/σ_y). Cut in square sigmas radial for saving particles (e.g. 90.25 for a cut at 9.5 σ_r). These values default to 1.

File Write Flags `WRITE_* true|false`

Switches on or off the writing of various output files, listed below. All flags default to `false`.

`WRITE_DIST`	Writes the beam distribution to file before and after tracking.
`WRITE_IMPACT`	Writes the impact parameters for each collimator.
`WRITE_SECOND`	Writes a secondary halo file based on normalised amplitude.
`WRITE_AMPL`	Writes checking files for amplitude, closed orbit.
`WRITE_EFFIC`	Writes the efficiency files.
`WRITE_TRACKS`	Writes secondary/tertiary halo files.
`WRITE_CRYCOORD`	Writes crystal coordinate file(s).

6.12.2 The Collimator Database

Collimators require specific settings, which are provided in a dedicated collimator database. The file is specified with the `COLLDB` keyword in the `COLL` block in `fort.3`. See Section 6.12.1.

Only the new database format is described in this section, but it is possible to load a file with the old format. The old format has been deprecated due to having a single column of parameters that are not straight-forward to read (by humans) and difficult to extend with new features. If an old database file is specified, a converted database is written with the same file name plus an added extension `.new`. New features added to the collimation module will not be supported by the old format.

Database Structure

The database is split into two sections. The first (main) section lists all the collimators and their default parameters. The second section, separated from the first by the keyword `SETTINGS`, contains additional settings for specific collimators or families of collimators. The second section is parsed following similar logic to the `fort.3` input block, but has a different set of keywords. The second section is optional.

6.12.2.1 Main Database Section

The main section of the database consists of a list of collimator family declarations, followed by a list of collimators. The family declaration is optional, but it makes it easier to assign parameters to a group of collimators instead of specifying each one individually. In the main section, this is used to provide the default collimator opening.

The collimator family settings are povided similarly to the `NISG_FAM` keyword in the `COLL` block in `fort.3`. The keyword takes a family name, a collimator opening setting, and a collimator stage as input parameters. Note that the sigma settings in `fort.3` overrides the settings of the database.

The collimator stage is not required, and defaults to `UNKNOWN`. Each collimator stage has a number associated with it that is an exponent of 2 (se table below). In files like `coll_ellipse.dat` and `tracks2.dat`, the sum of these are printed, meaning the information of what collimator stages a particle (that has not been absorbed) has hit can be extracted. Only the first three characters of the collimator stage is required, the rest of the word is optional for clarity.

`PRIMARY`	1	Primary collimators
`SECONDARY`	2	Secondary collimators
`TERTIARY`	4	Tertiary collimators
`OTHER`	8	Any other collimator
`CRYSTAL`	16	Crystal collimators
`UNKNOWN`	32	The collimator stage is unknown (default)

The collimator key settings are provided in a table of either 6 or 8 columns. The latter two columns are optional. The formats are as follows:

`elemname`	char	The single element name of the collimator.
`opening`	char/float	The family name as defined above, or if no family, the collimator opening in units of sigma.
`material`	char(4)	The material of the collimator.
`length`	float	The length of the collimator in metres.
`angle`	float	The skew angle (i.e. about the longitudinal axis) of the collimator in degrees.
`offset`	float	The offset of the collimator on the cleaning plane in metres.
`beta_x`	float	The Twiss β_x of the collimator in metres (optional).
`beta_y`	float	The Twiss β_y of the collimator in metres (optional).

Example

```
# Families
NSIG_FAM tcp3      12.000000 PRIMARY
NSIG_FAM tcsg3     15.600000 SECODNARY
NSIG_FAM tcsm3    999.000000 SECODNARY
NSIG_FAM tcla3     17.600000 TERTIARY
# Collimators
# name          opening mat.   length[m]  angle[deg]  offset[m]    beta_x[m]    beta_y[m]
tclx.4r1.b1        tclp Iner    1.000000    0.000000   0.000000  6812.605863  4597.092632
tcl.5r1.b1         tclp CU      1.000000    0.000000   0.000000   902.616764  1935.588447
tcl.6r1.b1         tclp Iner    1.000000    0.000000   0.000000   142.965006  1166.200842
tctph.4l2.b1      tcth2 Iner    1.000000    0.000000   0.000000    84.519178   104.259232
tctpv.4l2.b1      tctv2 Iner    1.000000   90.000000   0.000000    86.619339   103.111681
tdi.4l2.b1          tdi CU      4.000000   90.000000   0.000000   144.175250    96.678174
tclia.4r2          tcli C       1.000000   90.000000   0.000000    98.565386   160.118523
tclib.6r2.b1       tcli C       1.000000   90.000000   0.000000   149.324152    63.396485
tcp.6l3.b1         tcp3 C       0.600000    0.000000   0.000000   131.520626   144.694013
tcsg.5l3.b1       tcsg3 C       1.000000    0.000000   0.000000    54.604612   298.623904
tcsg.4r3.b1       tcsg3 C       1.000000    0.000000   0.000000    26.209714   395.196297
tcsg.a5r3.b1      tcsg3 C       1.000000  170.799865   0.000000    35.867692   344.111324
tcsg.b5r3.b1      tcsg3 C       1.000000   11.400000   0.000000    45.537821   312.677533
tcla.a5r3.b1      tcla3 Iner    1.000000   90.000000   0.000000   142.529342   176.013077
tcla.b5r3.b1      tcla3 Iner    1.000000    0.000000   0.000000   151.614782   168.687046
tcla.6r3.b1       tcla3 Iner    1.000000    0.000000   0.000000   129.435749   168.706418
tcla.7r3.b1       tcla3 Iner    1.000000    0.000000   0.000000    66.935067    92.241046
tcth.6l5.b1       tcth5 CuCD    1.000000    0.000000   0.000000  1262.953872   303.824116
tctv.6l5.b1       tctv5 CuCD    1.000000   90.000000   0.000000  1322.390819   361.810773
tctxh.4l5.b1      tcth5 CuCD    1.000000    0.000000   0.000000  4673.450003  7076.790857
```

6.12.2.2 Additional Collimator Settings

Further collimator settings can be provided at the bottom of the database file separated by the `SETTINGS` keyword. Currently, only one-sided collimators can be specified in this section.

One-Sided Collimators `ONESIDED name 1|2`

One-sided treatment of collimators can be achieved by setting this flag. The `name` can either be a collimator name or a family name. The last parameter specifies either jaw 1 (i.e. at x¿0 in the collimator reference system) or jaw 2 (i.e. at x¡0 in the collimator reference system). If jaw 2 is selected, the collimator is rotated by 180 degrees, as only the positive x side is treated when one-sided is enabled.

Example

```
SETTINGS
ONESIDED tcdqa.a4r6.b1    1
ONESIDED tcdqa.c4r6.b1    1
ONESIDED tcdqa.b4r6.b1    1
```

Jaw Fit Profile `JAW_PROFILE name fac0 [... fac5]`

Adds a named fit profile with factors s_i up to fifth order on the form:

$$P_i = A\left[s_0 + s_1 x_i + \frac{s_2 x_i^2}{L} + s_3 x_i^3 + s_4 x_i^4 + s_5 x_i^5\right] \tag{6.4}$$

where P_i is the profile of the i-th slice, A is a scaling factor, L is the length of the collimator, and

$$x_i = \frac{L(i-1)}{N} \tag{6.5}$$

where N is the number of slices. The scaling factors default to zero, and only the 0-th order factor is required.

Jaw Fit `JAW_FIT target nslices fit1 fit2 [scale1 scale2 [recentre1 recentre2]]`

Apply a fit defined with `JAW_PROFILE` to the two jaws of a `target` collimator or family of collimators. The number of slices is required, and so are the prfoile names `fit1` and `fit2` for collimator jaw 1 and 2, respectively. The scaling factors `scale1` and `scale2` are optional, and default to one. It is also possible to move the collimator such that the opening corresponds to the smallest gap after the jaw fit profile has been applied. This can be enabled by setting the logical flags `recentre1` and `recentre2` for the two jaws. These flags default to `off`.

Example

```
SETTINGS
JAW_PROFILE FIT_A -1.0e-5 -2.4e-3 4.3e-3 -4.1e-3 1.8e-3 3.0e-4
JAW_PROFILE FIT_B -1.0e-5 -2.4e-3 4.3e-3 -4.1e-3 1.8e-3 3.0e-4
JAW_FIT tcsg7 15 FIT_A FIT_B 1.0 1.0 on on
```

Crystal Collimators `CRYSTAL name Bend XDim YDim Thick Tilt MisCut Orient`

This flag enables the collimator or collimator family specified in `name` to be treated as a crystal collimator. Crystal collimators require an additional set of parameters described in the following:

`Bend`	float	Bending radius of the crystal collimator in [m].
`XDim`	float	Transverse dimension along the X axis of the crystal collimator in [m].
`YDim`	float	Transverse dimension along the Y axis of the crystal collimator in [m].
`Thick`	float	Thickness of the amorphous layer of the crystal collimator in [m].
`Tilt`	float	Tilt of the crystal collimator in [rad] with respect to its default orientation (calculated as the divergence of the beam at that location, i.e. optimal orientation for channeling).
`MisCut`	float	Miscut angle of the crystal collimator in [rad].
`Orient`	integer	Used only for Si crystal collimators. Its value is either 1 (strip crystal, 110 planes) or 2 (quasi-mosaic crystal, 111 planes).

All the parameters default to zero. Additional details on the physics principles of crystal collimation and benchmark against experimental data can be found in [55, 56, 57, 58, 59, 60].

The main output file is the `cry_interaction.dat` file. This file is enabled with the `WRITE_CRYCOORD` flag in `fort.3`. In addition, if the `DEBUG` flag is set in the `SETTINGS` block (see Section 3.4), a `cry_entrance.dat` and `cry_exit.dat` file is written as well.

Example

```
SETTINGS
CRYSTAL  cry.h.b1 61.54 2.0e-3 50.0e-3 0.0 0.0 0.0 1
```

6.12.3 Collimation with Geant4

It is possible to use the particle-matter scattering physics of Geant4 within SixTrack. To do this, the G4COLLIMATION flag must be added at build time, with an appropriate Geant4 library existing in your path. Many standard collimation features are currently not implemented. Particle longitudinal dynamics are also not currently implemented. The `GNT4` block specified below must be used in addition to the standard collimation block in the standard `fort.3` file.

Keyword `GNT4`
Data lines Variable, see below.
Format This module uses a keyword, value format. See below.

Select Physics List `PHYSICS phys`

The `PHYSICS` flag selects which physics list to use in Geant4. If no flag is specified, the simulation will default to FTFP_BERT. Any Geant4 physics list is allowed. Please see http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/index.html for more details.

`phys` char The name of the physics list to use.

Select Particles to Return to SixTrack `RETURN type`

The `RETURN` flag selects which types of particles to return back to SixTrack after performing collimation. Only charged particles are returned; neutral particles are always killed. Possible options are `STABLE`, which will return all stable particles, `IONS`, which takes back all ions (not including protons), and to specify by individual PDG id numbers. Both particles and anti-particles should be specified separately. Multiple copies of the `RETURN` flag can be used to precisely select which particles are of interest.
If no flag is specified, all particles are returned, including ones that should decay.

`type` char/int The particles to send back to SixTrack after tracking through Geant4.

Select Relative Energy Cut `RELENERGYCUT cut`

The `RELENERGYCUT` flag selects an energy cut as a fraction of the energy of the reference particle. For example, 0.5 will cut at half the energy of the reference particle.

`cut` float The cut value to use.

Select Absolute Energy Cut `ABSENERGYCUT cut`

The `ABSENERGYCUT` flag selects an absolute energy cut, and if a particle's energy goes below this value, it will be killed. This value is specified in GeV.

`cut` float The cut value to use in GeV.

Select Range Cut `RANGECUT_MM cut`

The `RANGECUT_MM` flag selects a range cut for particle production in Geant4. This value is specified in mm. The unit of this cut may change in the future.

`cut` float The cut value to use in mm.

Enable Debugging Output DEBUG

The DEBUG flag enables some additional debugging output in Geant4. Warning: the level of output can be very verbose with a large number of particles.

6.12.4 Old Input Format

The old input block format is illustrated in the following example, and the associated variable which each value is described in Table 6.14.

```
COLLIMATION
   .true.
   50   7000000
   3  5  0.958  0.0015  0.0  0.0  "nothing"  1.0 129e-4  75.5
   .true.  15.0  18.0  18.0  20.0  6.0  7.0  7.0  10.0 1 0.0  999.0  8.0   7.5   999.0
   8.3  8.3  8.3  8.3  8.3  8.3  8.3  8.3  5.0 15.0
   0 19789.0  20150.0  1  1
  -1.3899e-6  -9.345e-5  5.05324e-3  -1.6595e-2  2.15955e-2  -9.96261e-3  1.0
  -1.3899e-6  -9.345e-5  5.05324e-3  -1.6595e-2  2.15955e-2  -9.96261e-3  1.0
   0.503E-09  0.503E-09
  .false. .false. 0 .true. TCP.C6L7.B1 .false. .true. .true. .true. .true.
   0.0 0.0 0.0 0.0
   0   0   0   0   0   0   0   0   0   0   .false.
  .false.  6.003  0.0015
   0.0 0.0 .false. .false.
   0   0.0025  0.0   0.0   0
   "CollDB_V6.500_lowb_st.b1.data"  1
   .true. .false. WAbsVertLowbcoll  101  1  1.
NEXT
```

Old Collimator Database Format

The collimator database is stored in files named CollDB_V6.500_[type]_st.[beam].data, with [type] being either inj for injection case or low b for collision case, and [beam] being $b1$ or $b2$. These files contain mechanical and optical data related to the collimators planned for LHC. (Note that at present only Phase I collimators have a length different from zero)

A sample block of either one of these input files follows:

```
TCP.D6L7.B1            <-- collimator name in capital letters
tcp.d6l7.b1            <-- collimator name in minimal letters
5.7                    <-- collimator nominal opening (in sigma units)
C                      <-- collimator material (C = graphite, CU = copper, W=tungsten)
  0.2000000000000000   <-- collimator length [m]
  1.5710000000000000   <-- collimator angle [rad]
  0.0000000000000000   <-- collimator offset [m]
 90.4467000000000070   <-- design Beta x [m]
156.4360000000000070   <-- design Beta y [m]
#                      <-- line jump to next block
```

The introduction of the optic parameter β allows studies of error scenarios (orbit distortion, beta-beating) and/or to see the effect of misaligned collimators. This structure is then repeated within the file for each of the collimators to be included in the study.

Table 6.14: Old Collimation Input Format

Ln	Keyword	Type	Description
1	DO_COLL	logical	Switches on/off the collimation studies.
(The table continues on the next page)			

Ln	Keyword	Type	Description
2	NLOOP	integer	Number of samples. No longer in use. Fixed to 1.
	MYENOM	float	Energy of the beam to be tracked.
3	DO_THISDIS	integer	Selects the type of distribution of the particles to be tracked (see below).
	MYNEX	float	A_x normalized amplitude of particles (in sigma units) in the x direction.
	MDEX	float	dA_x smear (in sigma units) of the beam halo around A_x (in x direction).
	MYNEY	float	A_y normalized amplitude of particles (in sigma units) in the y direction.
	MDEY	float	dA_y smear (in sigma units) of the beam halo around A_y (in y direction).
	FILENAME_DIS	char	Name of the distribution file to be read if DO_THISDIS = 4/6.
	ENERROR	float	Energy spread of the tracked beam. Read only if DO_THISDIS = 3.
	BUNCHLENGTH	float	Bunch length of the tracked beam in millimeters. Read only if DO_THISDIS = 3.
4	DO_NSIG	logical	If TRUE use collimators settings from fort.3. If FALSE from CollDB_V6.500_[type]_st.[beam].data.
	NSIG_TCP3	float	Opening of the primary collimator in IR3 in sigma units.
	NSIG_TCSG3	float	Opening of the secondary graphite collimator in IR3 in sigma units.
	NSIG_TCSM3	float	Opening of the secondary metallic collimator in IR3 in sigma units.
	NSIG_TCLA3	float	Opening of the active absorbers in IR3 in sigma units.
	NSIG_TCP7	float	Opening of the primary collimators in IR7 in sigma units.
	NSIG_TCSG7	float	Opening of the secondary graphite collimator in IR7 in sigma units.
	NSIG_TCSM7	float	Opening of the secondary metallic collimator in IR7 in sigma units.
	NSIG_TCLA7	float	Opening of the active absorbers collimator in IR7 in sigma units.
	NSIG_TCLP	float	Opening of the physics debris collimator in sigma units.
	NSIG_TCLI	float	Opening of the absorbers for injection protection in sigma units.
	NSIG_TCDQ	float	Opening of the beam dump protection collimator in sigma units[4].
	NSIG_TCSTCDQ	float	Opening of secondary collimator dedicated to beam dump in sigma units.
	NSIG_TDI	float	Opening of the injection protection collimator in sigma units.
5	NSIG_TCTH1	float	Opening of the horizontal tertiary collimator in IR1 in sigma units.
	NSIG_TCTH2	float	Opening of the horizontal tertiary collimator in IR2 in sigma units.
	NSIG_TCTH5	float	Opening of the horizontal tertiary collimator in IR5 in sigma units.
	NSIG_TCTH8	float	Opening of the horizontal tertiary collimator in IR8 in sigma units.
	NSIG_TCTV1	float	Opening of the vertical tertiary collimator in IR1 in sigma units.
	NSIG_TCTV2	float	Opening of the vertical tertiary collimator in IR2 in sigma units.
	NSIG_TCTV5	float	Opening of the vertical tertiary collimator in IR5 in sigma units.
	NSIG_TCTV8	float	Opening of the vertical tertiary collimator in IR8 in sigma units.
	NSIG_TCXRP	float	Opening of the Roman Pots in sigma units[5].
	NSIG_TCRYO	float	Opening of the collimators in the DS regions in sigma units.

(The table continues on the next page)

[4] One-sided collimators (only positive x)
[5] One-sided collimators (only positive x)

Ln	Keyword	Type	Description
6	N_SLICES	float	Surface model of the jaw: number of slices in which each jaw should be cut.
	SMIN_SLICES	float	Surface model of the jaw: s position for the start of the slicing.
	SMAX_SLICES	float	Surface model of the jaw: s position for the end of the slicing.
	RECENTER1	float	Surface model of the jaw: moving the 1st jaw to the new smallest opening.
	RECENTER2	float	Surface model of the jaw: moving the 2nd jaw to the new smallest opening.
7	FIT1_1	float	Surface model of the jaw: Polynomial fit 0th order for the 1st jaw.
	FIT1_2	float	Surface model of the jaw: Polynomial fit 1st order for the 1st jaw.
	FIT1_3	float	Surface model of the jaw: Polynomial fit 2nd order for the 1st jaw.
	FIT1_4	float	Surface model of the jaw: Polynomial fit 3rd order for the 1st jaw.
	FIT1_5	float	Surface model of the jaw: Polynomial fit 4th order for the 1st jaw.
	FIT1_6	float	Surface model of the jaw: Polynomial fit 5th order for the 1st jaw.
	SSF1	float	Surface model of the jaw: Polynomial fit scaling for the 1st jaw.
8	FIT2_1	float	Surface model of the jaw: Polynomial fit 0th order for the 2nd jaw.
	FIT2_2	float	Surface model of the jaw: Polynomial fit 1st order for the 2nd jaw.
	FIT2_3	float	Surface model of the jaw: Polynomial fit 2nd order for the 2nd jaw.
	FIT2_4	float	Surface model of the jaw: Polynomial fit 3rd order for the 2nd jaw.
	FIT2_5	float	Surface model of the jaw: Polynomial fit 4th order for the 2nd jaw.
	FIT2_6	float	Surface model of the jaw: Polynomial fit 5th order for the 2nd jaw.
	SSF2	float	Surface model of the jaw: Polynomial fit scaling for the 2nd jaw.
9	DISTEMITX0	float	Normalised emittance in the horizontal plane for distribution.
	DISTEMITY0	float	Normalised emittance in the vertical plane for distribution.
	COLGAPEMITX0	float	Normalised emittance in the horizontal plane for collimator gaps.
	COLGAPEMITY0	float	Normalised emittance in the vertical plane for collimator gaps.
10	DO_SELECT	logical	Does dedicated study of selected collimator, see NAME_SEL.
	DO_NOMINAL	logical	Switches on/off the use of design β values of collimators.
	RND_SEED	integer	Seed studied. If set to 0, seed will be selected from system clock for every run.
	DOWRITE_DIST	logical	Saves the initial distribution to be tracked.
	NAME_SEL	char	Name as in the fort.2 file of the collimator one wants a dedicated study.
	DO_ONESIDE	logical	Switches on/off all TCPs being one-sided. Only positive jaw. If the negative jaw is to be used, it is necessary to turn collimator by 180 degrees in the collimator database file CollDB_V6.500_[type]_st.[beam].data.
	DOWRT_IMPACT	logical	Saves the impact parameters for each collimator.
	DOWRT_SECOND	logical	Writes a secondary halo file based on normalised amplitude.
	DOWRT_AMPL	logical	Writes checking files for amplitude, closed orbit.
	DOWRT_EFFIC	logical	Writes the efficiency files.
11	XBEAT	float	Offset in x for the computation of collimator in case of beta-beating.
	XBEATPHASE	float	Phase offset in x for the computation of collimator in case of beta-beating.

(The table continues on the next page)

Ln	Keyword	Type	Description
	YBEAT	float	Offset in y for the computation of collimator in case of beta-beating.
	YBEATPHASE	float	Phase offset in y for the computation of collimator in case of beta-beating.
12	RMSTILT_PRIM	float	RMS value of tilt to apply to primary collimators.
	RMSTILT_SEC	float	RMS value of tilt to apply to secondary collimators.
	SYSTILT_PRIM	float	Systematic value of tilt to apply to primary collimators.
	SYSTILT_SEC	float	Systematic value of tilt to apply to secondary collimators.
	RMSOFFS_PRIM	float	RMS value of offset to apply to primary collimators.
	RMSOFFS_SEC	float	RMS value of offset to apply to secondary collimators.
	SYSOFFS_PRIM	float	Systematic value of offset to apply to primary collimators.
	SYSOFFS_SEC	float	Systematic value of offset to apply to secondary collimators.
	OFFTILT_SEED	float	Random number seed to be used for the simulation.
	RMSERROR_GAP	float	RMS error of collimator gap.
	DO_MINGAP	logical	If TRUE the particle distribution is generated at the collimator with the smallest gap (to be used with sheet/pencil beam).
13	RADIAL	logical	Switches on/off the radial distribution.
	NR	float	Size of the beam to be tracked in number of radial sigmas.
	NDR	float	Smear of the beam to be tracked in number of radial sigmas.
14	DRIFTSX	float	Apply an emittance drift in the x direction.
	DRIFTSY	float	Apply an emittance drift in the y direction.
	CUT_INPUT	logical	This value is no longer used, and is ignored.
	SYSTILT_ANTI	logical	Deduce SYSTILT to RMSTILT instead of adding.
15	IPENCIL	integer	Resets original distribution to pencil beam distribution on selected collimator.
	PENC_OFFSET	float	Size in sigma units of the desired impact parameter.
	PENC_RMSX	float	See table below.
	PENC_RMSY	float	See table below.
	PENC_DISTR	integer	See table below.
16	COLL_DB	char	Name of the collimator database.
	IBEAM	integer	This value is no longer used, and is ignored.
17	DOWRITETRACKS	logical	Writes secondary/tertiary halo files.
	CERN	logical	This value is no longer used, and is ignored.
	CASTORDIR	char	This value is no longer used, and is ignored.
	JOBNUMBER	integer	This value is no longer used, and is ignored.
	SIGSECUT2	float	Cut in square sigmas x/y for saving particles (e.g. 64 for a cut at 8 σ_x/σ_y).
	SIGSECUT3	float	Cut in square sigmas radial for saving particles (e.g. 90.25 for a cut at 9.5 σ_r).

6.13 Fringe Fields

Note: This module is experimental, and not yet ready for production.

This method allows for the usage of a longitudinal description of the quadrupole magnetic field, adapted for each magnet specifically selected for the study, without changing the reference optics of

SixTrack [49, 50, 51]. This Fringe Field module is controlled by the `FFIE` block.

Keyword	`FFIE`
Data lines	Variable, see below.
Format	This module uses a keyword, value format. See below.

Select Quadrupole `FFQN quadname in ex`

The `FFQN` flag selects which quadrupole (name) has a longitudinal description in additional files that will be loaded in the study and link it to the type of Fringe Field that will be used (additional file).

`quadname`	char	Name of quadrupole. Must be the same as in the list of single elements.
`in`	integer	Index of the fringe field to use at the beginning of the quadrupole.
`ex`	integer	Index of the fringe field to use at the end of the quadrupole.

Select Skipped Multipole `FFMS multname`

The `FFMS` flag specify the name of the multipole to skip in the tracking, in order to not increase the integrated non-linearities if multipole kicks are already added for the quadrupole in the lattice.

`multname`	char	Name of multipole that will be skipped in the main lattice. Must be the same as in the list of single elements.

Profile Files `FFFI file length_quad length_tot [aperture]`

The `FFFI` flag is followed by the files where the different Fringe Field vector potential values are saved. The fringe field vector potential profile is saved using the format shown in Table 6.15.

`file`	char	Name of the fringe field profile file. See next table for proper formatting.
`length_quad`	float	Length inside the quadrupole, i.e. integrated length of the equivalent quadrupole for this file.
`length_tot`	float	Total length of integration inside the file.
`aperture`	float	The physical aperture of the quadrupole, or the maximum apeture at which you trust the representation of the Vector Potential. This parameter is optional. By defualt it is fixed as 0.080m.

Table 6.15: Input parameter in the Fringe Field profile files.

Arg.	Unit	Description
`z`	m	Longitudinal position of the kick.
`i`	–	Exponent of x in the Horner polynomial representation of the vector potential.
`j`	–	Exponent of y in the Horner polynomial representation of the vector potential.
`k`	–	Exponent of z in the Horner polynomial representation of the vector potential. Always 0 in case of magnetic quadrupole.
`Ax`	$m^{-(1+i+j)}$	Coefficient of the Horner polynomial of the vector potential for the axis x.

`Ay`	$m^{-(1+i+j)}$	Coefficient of the Horner polynomial of the vector potential for the axis y.
`Az`	$m^{-(1+i+j)}$	Coefficient of the Horner polynomial of the vector potential for the axis z.

Chapter 7

External Tools

SixTrack supports interfacing with external libraries and simulation tools. This chapter describes how to use these tools with SixTrack.

7.1 Pythia Integration

Note: This module should be considered experimental. It is not yet fully tested for correct physics.

Pythia8 is "a standard tool for the generation of high-energy collisions, comprising a coherent set of physics models for the evolution from a few-body hard process to a complex multihadronic final state" [45]. Version 8 is rewritten in c++ as a library, which makes it suitable for direct integration into SixTrack 5 via the update and extended c-interface available in Fortran2008. The integration was made specifically for the Scatter module to function as an event generator for single diffractive scattering.

The Pythia8 library is built using the `buildLibraries.sh` script, and linking with SixTrack is provided with the `PYTHIA` compiler flag. When the `PYTHIA` block is present in `fort.3`, a new generator becomes available to the Scatter module.

Note that the proton mass used by PYTHIA may not be the same as the proton mass in SixTrack. This may cause differences in the energy to momentum conversion. The mass from PYTHIA is printed to stdout during execution, and can also be found in the `scatter_summary.dat` file.

Particle Species `SPECIES beam1 beam2`

Set the particle species for Beam 1 and Beam 2 (or the target, if not a beam). These are the species used for the PYTHIA event generator, and in principle can be anything that PYTHIA supports, even if SixTrack cannot track it.

Supported particle species are: `PROTON`, `ANTIPROTON`, `NEUTRON`, `ANTINEUTRON`, `PION+`, `PION-`, `PION0`, `PHOTON`, `ELECTRON`, `POSITRON`, `MUON-`, and `MUON+`.

Beam Energy `ENERGY beam1 [beam2]`

The energy of the beams in MeV. If only Beam 1 is specified, the value is taken as the centre of mass energy by PYTHIA.

Use Tracked Particles `REALBEAM on|off`

By default, PYTHIA generates events for head on collisions at reference energy. The event is then projected back onto the tracked particle in SixTrack. If, instead, you want to send the actual tracked particle to SixTrack, together with a sample particle generated from a beam distribution, set the `REALBEAM` flag to "on".

Process Selection `PROCESS type [crossSection]`

The scattering process for the event generator can be set with this keyword. One process for each line. If the `MODES` are set to `MANUAL`, the cross sections can be specified as a second argument.

Available process types are:

- Elastic: `EL`, `ELASTIC`
- Single diffractive: `SD`, `SINGLEDIFF`,`SINGLEDIFFRACTIVE`
- Double diffractive: `DD`, `DOUBLEDIFF`,`DOUBLEDIFFRACTIVE`
- Central diffractive: `CD`, `CENTRALDIFF`,`CENTRALDIFFRACTIVE`
- Non-diffractive: `ND`, `NONDIFF`,`NONDIFFRACTIVE`

Coulomb `COULOMB on|off [tmin]`

Enable or disable the Coulomb part of the elastic scattering generator. A `tmin` is required, but if it is not provided here, it defaults to 5×10^{-5}.

Total Cross Section `SIGTOTAL crossSection`

The total cross section can be provided in units of millibarn. This is only relevant when the `MODES` are set to `MANUAL`. The total cross section is then sent to PYTHIA with the cross sections provided with the `PROCESS` settings.

Total Cross Section Mode `SIGTOTALMODE mode`

Set the total cross section mode for PYTHIA. The options are: `MANUAL`, `DL`, `MBR`, `ABMST`, and `RPP2016`. Alternatively, an integer value can be provided. For a description of these options, and their integer equivalents, please consult the PYTHIA manual [45].

Diffractive Cross Section Mode `SIGDIFFMODE mode`

Set the diffractive cross section mode for PYTHIA. The options are: `MANUAL`, `SAS`, `MBR`, and `ABMST`. Alternatively, an integer value can be provided. For a description of these options, and their integer equivalents, please consult the PYTHIA manual [45].

Random Seed `SEED value`

The random number seed to send to the PYTHIA library. This has no relation to the internal SixTrack random seeds.

Settings File `FILE fileName`

If the `PYTHIA` block does not provide sufficient settings, a PYTHIA settings file can be set instead using this command.

Example

```
PYTHIA-----------------------------------------------------------------
  SPECIES PROTON PROTON
  ENERGY 7000000 7000000
  SEED 41
  PROCESS ELASTIC
  PROCESS SINGLEDIFFRACTIVE
NEXT

SCATTER----------------------------------------------------------------
  GEN  sc_elastic  PPBEAMELASTIC 0.046  18.52  4.601  2.647  0.2  8.0
  GEN  sc_pythia   PYTHIA  27.0
  PRO  ipFLAT      FLAT   3e25  938.0  7000000
  ELEM ip1_scatter ipFLAT auto 1.0 sc_elastic
  ELEM ip5_scatter ipFLAT auto 1.0 sc_pythia
NEXT
```

Chapter 8

Organising Tasks

In this chapter, the input data blocks used to organise the input structure are described.

8.1 Random Fluctuation Starting Number

If besides mean values for the multipole errors (Gaussian) random errors should be considered, this input data structure is used to set the start value for the random generator.

Keyword `FLUC`
Data lines 1
Format `izu0 mmac mout mcut` (integers)

Format Description

`izu0` Start value for the random number generator

`mmac` Support for multiple starting seeds has been removed. This value must be 1.

`mout` A binary switch for various purposes, so all options, as described below, can be combined.
`mout` = 0 : multipole errors internally created
`mout` = 1 : multipole errors read-in from external file
External multipole errors are read-in from file 16 into the array of random values. To activate these values one has to set to a value of 1 the relevant r.m.s.-positions of the corresponding multipole blocks (6.1). The systematic components are added as usual and multipoles not found in the `fort.16` are treated as for (`mout` = 0). An error is only detected if there are too few sets of multipoles in `fort.16`.
`mout` = 2: the geometry and strength file is written to file `fort.4` in the same format as the input file `fort.2`; the multipole coefficients are written to file `fort.9`; name, misalignments and tilt is written to file `fort.27` and finally name, random single multipole strength and both random transverse misalignments are written to file `fort.31`.
`mout` = 4: Name, horizontal and vertical misalignment and also the element tilt are read-in from file `fort.8`.
`mout` = 8: Name and 3 Random numbers for single kick strength and both random transverse misalignments and also the value of the tilt are read-in from file `fort.30`.

`mcut` The random distribution can be cut by `mcut` sigma of the distribution. No cuts are applied for `mcut = 0`.

Remarks

1. The `RANECU` random generator [26] is used as it produces machine independent sequences of random numbers.

2. If the starting point has to be changed or another non-linear element is to be inserted, this can be done without changing the once chosen random distribution of errors by using the *Organisation of Random Numbers* input block.

3. The description of an accelerator is fully contained in 4 files: `fort.2` (geometry), `fort.3` (tracking parameters and definition of multipole blocks), `fort.16` (multipole errors) and `fort.30` (random numbers of the single multipole kick, the horizontal and vertical misalignment and the value of the tilt). This block allows to write out the files `fort.4`, `fort.9`, `fort.27`, `fort.31` which may serve as the input files `fort.2`, `fort.16`, `fort.8` and `fort.30` respectively. The file `fort.30` supersedes `fort.8` if both files are read in.

8.2 Organisation of Random Numbers

Working on a lattice for an accelerator often requires to introduce new non-linear elements. In those cases simply introducing this new element means that the previously chosen random distribution of the errors will be changed and with it often the linear parameters. This input data block is mainly used to avoid this problem by reserving extra random numbers for the new elements. It also allows to change the observation point without affecting the machine. The random values of different nonlinear elements including blocks of multipoles can be set to be equal to allow to vary the number of nonlinear kicks in one magnet which clearly should have the same random distribution for each multipolar kick. Finally, multipole sets with different name can be made equal with this input data block.

Keyword `ORGA`
Data lines Variable
Format `ele1 ele2 ele3`
The data lines can be set in three different ways described below.

Method 1 `Ele1` = "name" where name $\neq$ `MULT`
`Ele2` = ignored
`Ele3` = ignored
The nonlinear element or multipole set will have its own set of random numbers.

Method 2 `Ele1` = "name1" where name1 $\neq$ `MULT`
`Ele2` = "name2"
`Ele3` = ignored
The nonlinear element or multipole block `Ele1` has the same random number set as those of `Ele2`, if it follows `Ele2` as the first non-linear element in the structure list (5.2.1).

Method 3 `Ele1` = `MULT`
`Ele2` = "name2"
`Ele3` = "name3"
The multipole set "name3" is set to the values of the set "name2". random errors are not influenced in this case.

Remarks

1. A simple change of the starting point, by placing a `GO` somewhere in structure, used to change the machine optics as the random numbers were shifted, too. Simply calling this block even without a data line, will always fix the sequence of random numbers to start at the first multipole in the structure.

2. This input data block must follow the definition of the multipole block, otherwise multipoles cannot be set equal (option 3).

3. Do not use the keyword `MULT` in the single element list (5.1).

8.3 Combination of Elements

It is often necessary to use several families of magnetic elements with a certain ratio R of magnetic strength to perform corrections like tune adjustment (9.2), chromaticity correction (9.3) or resonance compensation (9.8). The *Combination of Elements* input block allows such a combination of elements. The maximum number of elements is defined by the parameter `NCOM` (see Appendix A.2).

Keyword	`COMB`
Data lines	Variable
Format	`e0 R1 e1 ... Rn en`

Format Description

`e0`	Reference element which appears in the input of the processing procedure
`e1, ..., en`	Elements to be combined with `e0`
`Rj`	Ratio of the magnetic strength of element `ej` to that of element `e0`

Chapter 9

Processing

This chapter comprises all the input blocks that do some kind of pre- or post-processing.

9.1 Linear Optics Calculation

The linear optics calculation input block is used to make a print-out of all linear parameters (magnet lengths, β and α functions, tunes, dispersion and closed orbit) in the horizontal and vertical planes at the end of each element or linear block. The number of elements or blocks can be chosen.

Keyword	`LINE`
Data lines	≥ 1
Format	First line: `mode num_blocks ilin ntco E_I E_II`
	Other lines: `name(1), ..., name(nlin)`

Format Description

`mode`	char	`ELEMENT` for a printout after each single element (5.1).
		`BLOCK` for a printout after each structure block (5.2).
`num_blocks`	integer	The number of the blocks in the structure to which the linear parameter will be printed. If this number is set to zero or is larger than the number of blocks, the complete structure will be calculated.
`ilin`	integer	Logical switch to calculate the traditional linear optics calculation in 4D (`1 = ilin`) and with the DA approach 6D (`2 = ilin`).
`ntco`	integer	A switch to write out linear coupling parameters.
		`ntco = 0`: no write-out.
		`ntco` $\neq$ `0`: write-out of all linear coupled (4D) parameters including the coupling angle. These parameters (name, longitudinal position, the phase advances at that location, 4 β-, α- and γ-functions, 4 angles for coordinates and momenta respectively, plus the coupling angle [rad]) are written in ascii format on file `linopt_coupled.dat`. This write-out happens every `ntco` turns.
`E_I, E_II`	floats	The two eigen-emittances to be chosen to determine the coupling angle. They are typically set to be equal.
`names`	char	For `nlin` $\leq$ `nele` element and block names the linear parameters are printed whenever they appear in the accelerator structure.

Remarks

- To make this block work the Tracking Parameter block (4.2) has to used as well.
- When the `ELEMENT 0` option is used, a file `linopt_dump.dat` is written with the longitudinal position, name, element type, multipole strength, β functions and phase advances in the horizontal and vertical phase space respectively. This file is used as input for the `SODD` program [22] to calculate de-tuning and distortion terms in first and second order. A full program suite can be found at: /afs/cern.ch/group/si/slap/share/sodd
- If the `BLOCK` option has been used, the tunes may be wrong by a multiple of 1/2. This option is not active in the DA part (`2 = ilin`), which also ignores the (`NTCO`) option.

9.2 Tune Variation

This input block initializes a tune adjustment with zero length quadrupoles. This is normally done with two families of focusing and defocusing quadrupoles. It may be necessary, however, to have a fixed phase advance between certain positions in the machine. This can be done with this block by splitting the corresponding family into two sub-families which then are adjusted to give the desired phase advance.

Keyword	`TUNE`
Data lines	2 or 4
Format	Line 1: `name1 Qx iqmod6`
	Line 2: `name2 Qy`
	Line 3 (optional): `name3` ΔQ
	Line 4 (optional): `name4 name5`

Format Description

`name1, name2`	char	Names of focusing and defocusing quadrupole families respectively (in the single element list (5.1.1).
`Qx, Qy`	floats	Horizontal and vertical tune *including* the integer part.
`iqmod6`	integer	Switch to calculate the tunes in the traditional manner (`1 = iqmod6`) and with the DA approach including the beam-beam kick (`2 = iqmod6`). FixMe: This is out of date and does not match the source code.
`name3`	char	Name of the second sub-family, where the first sub-family is one of the above (`name1` or `name2`). This second sub-family replaces the elements of the first sub-family between the positions marked by `name4` and `name5`.
ΔQ	float	Extra phase advance *including* the integer part (horizontal or vertical depending on the first sub-family) between the positions in the machine marked by `name4` and `name5`.
`name4, name5`	char	Two markers in the machine for the phase advance ΔQ with the elements of the second sub-family between them

Remarks

The integer has to be included as the full phase advance around the machine is calculated by the program.

9.3 Chromaticity Correction

The chromaticity can be adjusted to desired values with two sextupole family using this input block.

Keyword	`CHRO`
Data lines	2
Format	Line 1: `name1` Q'_x `ichrom`
	Line 2: `name2` Q'_y

Format Description

`name1, name2`	char	Names (in the single element list (5.1.2) of the two sextupole families.
Q'	float	Desired values of the chromaticity: $Q' = \frac{\delta Q}{\delta(\frac{\Delta p}{p_o})}$.
`ichrom`	integer	Logical switch to calculate the traditional chromaticity calculation (`1`) and with the DA approach including the beam-beam kick (`2`). Setting this flag to `3` switches on both.

Remarks

To make the chromaticity correction work well a small momentum spread is required (`DE0` in table (3.1)). It sometimes is required to optimize this spread.

9.4 Orbit Correction

Due to dipole errors in a real accelerator, a closed orbit different from the beam axis is unavoidable. Even after careful adjustment, one always will be left over with some random deviation of the closed orbit around the zero position. A closed orbit is introduced by non-zero strengths of b_1 and a_1 components of the multipole block (6.1), horizontal and vertical dipole kicks (5.1.2), or displacements of non-linear elements (5.2.2). This input data block allows the correction of a such a random distributed closed orbit using he first two types in a "most effective corrector strategy" [27]. For that purpose, correctors have to be denoted by `HCOR` and `VCOR`, and monitors by `HMON` and `VMON` for the horizontal and vertical plane respectively. After correction, the orbit is scaled to the desired r.m.s. values, unless they are zero.

The horizontal orbit displacement, measured at the horizontal monitors, will be written to `fort.28` – together with the monitor number, in `fort.29`. The same is done for the vertical closed orbit displacement.

Keyword	`ORBI`
Data lines	≥ 1
Format	First line: `sigmax sigmay ncorru ncorrep`
	Other lines: `HCOR namec`, `HMON namem`, `VCOR namec` or `VMON namem`.

Format Description

`sigmax, sigmay`	Desired r.m.s.-values of the randomly distributed closed orbit.
`ncorru`	Number of correctors to be used.
`ncorrep`	Number of corrections.

	If `ncorrep=0`, the correction is iterated until `ITCO` iterations or after the both desired r.m.s.-values have been reached (see table 3.1).
`HCOR=namec`	Horizontal correction element of name `namec`.
`HMON=namem`	Horizontal monitor for the closed orbit of name `namem`.
`VCOR=namec`	Vertical correction element of name `namec`.
`VMON=namem`	Vertical monitor for the closed orbit of name `namem`.

Remarks

- Elements can have only one extra functionality: either horizontal corrector, horizontal monitor, vertical corrector or vertical monitor. If the number of monitors in a plane is smaller than the number of correctors it is likely to encounter numerical problems.
- The `HCOR`, `HMON`, `VCOR`, and `VMON` must be separated from the following name by at least one space.

9.5 Decoupling of Motion in the Transverse Planes

Skew quadrupole components in the lattice create a linear coupling between the transverse planes of motion. A decoupling can be achieved with this block using four independent families of skew-quadrupoles, which cancel the off-diagonal parts of the transfer map. As these skew quadrupoles also influence the tunes an adjustment of the tunes is performed at the same time.

Keyword	`DECO`
Data lines	3
Format	Line 1: `name1,name2,name3,name4`
	Line 2: `name5 Qx`
	Line 3: `name6 Qy`

Format Description

`name1,2,3,4`	char	Names of the four skew quadrupole families.
`name5,6`	char	Names of focusing and defocusing quadrupole families respectively (in the single element list (5.1.1).
`Qx, Qy`	floats	Horizontal and vertical tune *including* the integer part.

Remarks

A decoupling can also be achieved by compensating skew-resonances (9.8). The two approaches, however, are not always equivalent. In the resonance approach the zeroth harmonic is compensated, whilst a decoupling also takes into account the higher order terms.

9.6 Sub-Resonance Calculation

First order resonance widths of multipoles from second to ninth order are calculated following the approach of Guignard [10]. This includes resonances, which are a multiple of two lower than the order of the multipole. The first order detuning including feed-down from closed orbit is calculated from all multipoles up to to tenth order.

Keyword	`SUBR`
Data lines	1
Format	`n1 n2 Qx Qy Ax Ay Ip length`

Format Description

`n1, n2`	integers	Lowest and highest order of the resonance.
`Qx, Qy`	floats	Horizontal and vertical tune including the integer part.
`Ax, Ay`	floats	Horizontal and vertical amplitudes in mm.
`Ip`	integer	Is a switch to change the nearest distance to the resonance $e = nxQx + nyQy$. In cases of structure resonances a change of p by one unit may be useful. `ip = 0`: e is unchanged. `ip = 1`: $(e \pm 1) = nxQx + nyQy - (p \pm 1)$.
`length`	float	Length of the accelerator in meters

9.7 Search for Optimum Places to Compensate Resonances

To be able to compensate a specific resonance, one has to know how a correcting multipole affects the cosine and sine like terms of the resonance width at a given position in the ring. This input data block can be used to find best places for the compensation of up to three different resonances, by calculating the contribution to the resonance width for a variable number of positions. For each position, the effect of a fixed and small change of magnetic strength on those resonance widths is tested.

Keyword	`SEAR`
Data lines	≥ 2
Format	Line 1: `Qx Qy Ax Ay length` Line 2: `npos n ny1 ny2 ny3 ip1 ip2 ip3` Other lines: `name1, ..., namen`

Format Description

`Qx, Qy`	floats	Horizontal and vertical tune including the integer part.
`Ax, Ay`	floats	Horizontal and vertical amplitudes in mm.
`length`	float	Length of the accelerator in m.
`npos`	integer	Number of positions to be checked.
`n`	integer	Order of the resonance.
`ny1,ny2,ny3`	integers	Define three resonances of order n via: $nxQx + nyQy = p$ with $\|nx\| + \|ny\| = n$.
`ip1,ip2,ip3`	integers	The distance to a resonance is changed by an integer ip for each of the three resonances: $e = nxQx + nyQy - (p + ip)$.
`namei`	char	The i-th name of a multipole of order n, which has to appear in the single element list (5.1.2).

9.8 Resonance Compensation

The input block allows the compensation of up to three different resonances of order n simultaneously. The chromaticity and the tunes can be adjusted. For mostly academic interest, there is also the possibility to consider sub-resonances, which come from multipoles, which are a multiple of 2 larger

than the resonance order n. However, it must be stated that the sub-resonances depend differently on the amplitude compared to resonances where the order of the resonances is the same as that of the multipoles.

Keyword `RESO`
Data lines 6
Format
Line 1: `nr n ny1 ny2 ny3 ip1 ip2 ip3`
Line 2: `nrs ns1 ns2 ns3`
Line 3: `length Qx Qy Ax Ay`
Line 4: `name1, ..., name6`
Line 5: `nch name7 name8`
Line 6: `nq name9 name10 Qx0 Qy0`

Format Description

`nr`	integer	Number of resonances (0 to 3).
`n`	integer	Order of the resonance, which is limited to `nrco= 5` (see list of parameters in Appendix A.2). normal: $3 \leq n \leq nrco$; skew: $2 \leq n \leq nrco$.
`ny1,ny2,ny3`	integers	Define three resonances of order n via: $nxQx + nyQy = p$ with $\|nx\| + \|ny\| = n$.
`ip1,ip2,ip3`	integers	The distance to the resonance e can be changed by an integer value: $e = nxQx + nyQy - (p + ip)$.
`nrs`	integer	Number of sub-resonances (0 to 3).
`ns1,ns2,ns3`	integers	Order of the multipole with $ns \leq 9$ and $(ns - n)/2 \in \mathbf{N}$.
`length`	float	Length of the machine in meters.
`Qx, Qy`	floats	Horizontal and vertical tune including the integer part.
`Ax, Ay`	floats	Horizontal and vertical amplitudes in mm.
`name1-6`	char	Names (5.1.2) of the correction multipoles for the first, second and third resonance.
`nch`	integer	Switch for the chromaticity correction (0 = off, 1 = on).
`name7,8`	char	Names (5.1.2) of the families of sextupoles to correct the chromaticity.
`nq`	integer	Switch for the tune adjustment (0 = off, 1 = on).
`name9,10`	char	Names (5.1.1) of the families of quadrupoles to adjust the tune.
`Qx0, Qy0`	floats	Desired tune values including the integer part.

9.9 Differential Algebra

This input block initiates the calculation of a one turn map using the LBL Differential Algebra package [1]. The use of this block inhibits post-processing. The same differential algebra tools allow a subsequent normal form analysis (see [18]). A four-dimensional version integrated in SixTrack is available as described in sections 9.10 and 9.11.

Keyword	`DIFF`
Data lines	1 or 2
Format	Line 1: `nord nvar preda nsix ncor`
	Line 2: `name(1),...,name(ncor)`

Format Description

`nord`	integer	Order of the map.
`nvar`	integer	Number of the variables (2 to 6). `nvar = 2,4,6`: two- and four-dimensional transverse motion and full six-dimensional phase space respectively. `nvar = 5`: four-dimensional transverse motion plus the relative momentum deviation $\frac{\Delta p}{p_o}$ as a parameter.
`preda`	float	Precision needed by the DA package, usually set to `preda= 1e-38`.
`nsix`	integer	Switch to calculate a 5×6 instead of a 6×6 map. This saves computational time and memory space, as the machine can be treated up to the cavity as five-dimensional (constant momentum). `nsix = 0`: 6×6 map. `nsix = 1`: 5×6 map. (*`nvar`* must be set to 6; 6D closed orbit must not be calculated, i.e. `iclo6 = 0` (4.3) and the map calculation is stopped once a cavity has been reached and being evaluated.)
`ncor`	integer	Number of zero-length elements to be additional parameters besides the transverse and/or longitudinal coordinates (i.e. two-, four-, five- or six-dimensional phase space).
`name(i)`	char	`Ncor` names (5.1.2) of zero-length elements (e.g dipole kicks, quadrupole kicks, sextupoles kicks etc.)

Remarks

- For `nsix = 1`, the map can only be calculated till a cavity is reached.
- If the 6D closed orbit is calculated, the 5×6 map cannot be done. `nsix` is therefore forced to 0.
- If `nvar` is set to 5, the momentum dependence is determined without the need for including a fake cavity. With other words: the linear blocks are automatically broken up into single linear elements so that the momentum dependence can be calculated.
- If a DA map is needed at some longitudinal location, one just has to introduce an element denoted `DAMAP` at that place in the structure, `DAMAP` has also to appear as a marker (zero length, element type = 0) in the single element list (5.1.2). This extra map is written to file `fort.17`.

9.10 Normal Forms

All the parameters to compute the Normal Form of a truncated one turn map are given in the *Normal Form* input block. Details on these procedures including the next block 9.11 can be found in reference [28].

Keyword	`NORM`
Data lines	1
Format	`nord nvar`

Format Description

`nord`	integer	Order of the Normal Form.
`nvar`	integer	Number of variables.

Remarks

- The *Normal Form* input block has to be used in conjunction with the *Differential Algebra* input block that computes the one turn map of the accelerator.
- The value of the parameter `nord` should not exceed the order specified for the transfer map plus one.
- The value of the parameter `nvar` should be equal to the number of coordinates used to compute the map plus eventually the number of correctors specified in the *Differential Algebra* input block.
- the value 1 for the off-momentum order is forbidden. This case corresponds to the linear chromaticity correction. It is in fact corrected by default when $par1 = 1$ or $par2 = 2$.

9.11 Corrections

Note: The `CORR` block is deprecated as of SixTrack 5.0-RC3.

9.12 Post-Processing

It has been seen in the past that the tracking data hold a large amount of information which should be extracted for a thorough understanding of the nonlinear motion. It is therefore necessary to store the tracking data turn by turn and post-process it after the tracking has been finished. The following quantities are calculated:

1. **Lyapunov exponent analysis:** This allows to decide if the motion is of regular or chaotic nature, and, in the latter case, that the particle will ultimately be lost. This is done with the following procedure:
   (a) Start the analysis where the distance in phase space of the two particles reaches its minimum.
   (b) Study the increase in a double logarithmic scale so that the slope in a regular case is always one, while a exponential increase stays exponential when we have chaos.
   (c) Average the distance in phase space to reduce local fluctuations, as we are interested in a long range effect.
   (d) Make a weighted linear fit with an increasing number of averaged values of distance in phase space, so that an exponential increase results in a slope that is larger than one and is increasing. (The weighting stresses the importance of values at large turn numbers).
2. **Analysis of the tunes:** This is done either by the averaged phase advance method leading to very precise values of the horizontal and vertical tunes. An FFT analysis is also done. With the second method, one can evaluate the relative strength of resonances rather than achieve a precise tune measurement. In both cases, the nearby resonances are determined.

3. **Smear:** The smear of the horizontal and vertical emittances, and the sum of the emittances, are calculated in case of linearly coupled and un-coupled motion.
4. **Nonlinear Invariants:** A rough estimate of the nonlinear invariants are given.
5. **Plotting:** The processed tracking data can be plotted in different ways:
   (a) The distance of phase space as a function of amplitude.
   (b) Phase space plots.
   (c) Stroboscoped phase space.
   (d) FFT amplitudes.
6. **Summary:** The post-processing results for a complete tracking session with varying initial parameters are summarised in a table at the end of the run.

Keyword	`POST`
Data lines	4
Format	Line 1: `comment title` Line 2: `iav nstart nstop iwg dphix dphiy iskip iconv imad cma1 cma2` (general parameters) Line 3: `Qx0 Qy0 ivox ivoy ires dres ifh dfft` (parameters for the tune calculation) Line 4: `kwtype itf icr idis icow istw iffw nprint ndafi` (integer parameters for the plotting)

Format Description

`iav`	integer	Averaging interval of the values of the distance in phase space. Typically a tenth of the total turn number should be used as this interval.
`nstart,nstop`	integers	Start and stop turn number for the analysis of the post-processing (0 0 = all data used).
`iwg`	integer	Switch for the weighting of the slope calculation of the distance in phase space (0 = off, 1 = on).
`dphix,dphiy`	floats	Horizontal and vertical angle interval in radians that is used to stroboscope phase space. This stroboscoping of one of the two phase space projections is done by restricting the angle in the other phase space respectively to lie inside ± `dphix` or ± `dphiy`.
`iskip`	integer	This parameter allows to reduce the number of data to be processed: only each `iskip` sample of data will be used.
`iconv`	integer	If `iconv` is set to 1, the tracking data are not normalised linearly. Sometimes it is necessary to compare normalised to unnormalised data as the later will be found in the real machine.
`imad`	integer	This parameters is useful when Mad-X data shall be analysed (`imad` set to one).
`cma1,cma2`	floats	To improve the Lyapunov analysis for Mad-X data, and in the case that the motion is 6D but the 6D closed orbit is not calculated the off-momentum and the path-length difference ($\sigma = s - v_o \times t$) can be scaled with `cma1` and `cma2` respectively (see also 4.4). Please set both to 1. when the 6D closed orbit is calculated.

`Qx0, Qy0`	floats	Values of the horizontal and vertical tune respectively (integer part) to be added to the averaged phase advance and to the Q values of the FFT analysis.
`ivox, ivoy`	integers	The tunes from the average phase advance are difficult to be calculated when this phase advance is strongly changing from turn to turn and when the tune is close to 0.5, as then the phase may become negative leading to a deviation of one unit. This problem can partly be overcome by setting these switches in the following way: tune close to an integer: `ivox, ivoy = 1`. tune close to half an integer: `ivox, ivoy = 0`.
`ires, dres`	int,float	For the calculated tune values from the average phase advance method and the FFT-routine the closest resonances are searched up to `ires`'th order and inside a maximum distance to the resonance `dres`, so that $nxQx + nyQy < dres$ and $nx + ny \leq ires$.
`ifh, dfft`	int,float	For the FFT analysis, the tune interval can be chosen with `ifh`. To find resonances with the FFT spectrum, all peaks below a fraction `dfft` of the maximum peak are accepted. `ifh = 0`: $0 \leq Q \leq 1$. `ifh = 1`: $0 \leq Q \leq 0.5$. `ifh = 2`: $0.5 \leq Q \leq 1$.
`kwtype`	integer	Disabled, set to 0. Terminal type, e.g. 7878 for the Pericom graphic terminals. For details, consult the HPLOT manual [8].
`itf`	integer	Switch to get PS file of plots: `itf = 0`: off `itf = 1`: on
`icr`	integer	Disabled, set to 0 Switch to stop after each plot (0 = no stop, 1 = stop after each plot).
`idis, icow` `istw, iffw`	integers	Switches (0 = off) to select the different plots. If all values are set to zero, the HBOOK/HPLOT routine will not be called. `idis = 1`: plot of distance in phase space. `icow = 1`: a set of plots of projections of the six-dimensional phase space and the energy E versus the turn number. `istw = 1`: plot of the stroboscoped phase space projection by restricting the phase in the other phase space projection. `iffw = 1`: plots of the horizontal and vertical FFT spectrum with linear amplitude scale. `iffw = 2`: plots of the horizontal and vertical FFT spectrum with logarithmic amplitude scale.
`nprint`	integer	Switch to stop the printing of the post-processing output to unit 6 (0 = printing off, 1 = printing on).
`ndafi`	integer	Number of particle pairs to be processed, starting from first pair.

Remarks

1. The post-processing can be done in two ways:

(a) directly following a tracking run by adding this input block to the input blocks of the tracking,

(b) as a later run where the tracking parameter file `fort.3` consists of only the *Program Version* input block 3.2 (using the `FREE` option) and of this input block specifying the post-processing parameters followed by `ENDE` as usual.

2. The HBOOK/HPLOT routines are only used at the start of the main program for initialisation and termination. The actual plots are done in the post-processing subroutine. The routines are activated only if at least one of the plotting parameters (`idis, icow, istw, iffw`) is set to one.

Chapter 10

Extra Output Files

For some studies, extra output from the simulation is desired. How to do this is described below.

10.1 Dumping of Beam Population

The DUMP block allows the beam population (i.e. the position in phase-space for all the particles) to be written to file. This can be done in any SINGLE ELEMENTS which are directly mentioned in the STRUCTURE INPUT part of fort.2 (BLOCs cannot be used). The particles are dumped just after the kick is applied, and how often to dump (every turn, every second turn, etc.) is user-selectable. Please note that each single element can only be selected once; however it is possible to overcome this limitation by placing multiple markers with different names in the same position in the sequence (by editing fort.2).

Keyword	DUMP
Data lines	Variable, one for each element for which dump is active.
Format	element_name frequency unit format (filename) (first last) or HIGH or FRONT

Format Description

element_name	char	One of the *single elements*, or ALL to dump at the exit of all single elements, or StartDUMP to dump at the injection point. Note that if ALL or StartDUMP is in use, these cannot be used as single element names.
frequency	integer	How often the beam population should be dumped in number of turns.
unit	integer	This value is ignored. Unit numbers are now assigned automatically.
format	integer	A switch specifying the output format. See table (10.3).
filename	char	The name of the file to write to. The filename may be shared between different DUMP outputs, as long as they have the same format and element_name is not ALL. This argument may be omitted (unless first and last are present, if so, then filename must also be present), and if so the output file is named dump_element_name.
first	integer	The first turn where this dump should be active. This argument may be omitted if last is also omitted, and if so it defaults to turn 1.

`last`	integer	The last turn where this dump should be active, –1 meaning "untill the end of the simulation". This argument may be omitted if `first` is also omitted, and if so it defaults to -1.
`HIGH`	keyword	If present anywhere in the `DUMP` block, this triggers high-precission output, meaning more digits in the output files.
`FRONT`	keyword	If present anywhere in the `DUMP` block, this keyword triggers the DUMPed particles to be dumped in front of the element, i.e. before the kick. This works for all elements, including `BLOC`s, when combined with the `ALL` as `element name`. Note that `FRONT` is not yet supported for thick tracking, and trying to use this combination will produce a run-time error.

Conventions

Table 10.2: The following table shows a summary of the quantities used in describing the output format. The units are declared explicitly for each dump.

Name	Variable	Unit	Symbol	Description
`x`	`xv1(j)`	[mm]	x	horizontal position
`y`	`xv2(j)`	[mm]	y	vertical position
`xp`	`yv1(j)`	[1/1000]	$\frac{P_x}{P} \approx x'$	approximated horizontal angle
`yp`	`yv2(j)`	[1/1000]	$\frac{P_y}{P} \approx y'$	approximated vertical angle
`sigma`	`sigmv(j)`	[mm]	$\sigma = s - \beta_0 ct$	longitudinal offset (sometimes called `z`)
`psigma`	n/a	[1]	$p_\sigma = \frac{E-E_0}{\beta_0 P_0 c}$	canonical conjugate of σ
`delta`	`dpsv(j)`	[1]	$\delta = \frac{P-P_0}{P_0}$	canonical conjugate of σ
`rv`	`rvv(j)`	[1]	$r_v = \frac{\beta_0}{\beta}$	velocity ratio
`rpp`	`oidpsv(j)`	[1]	$r_p = \frac{P_0}{P}$	momentum ratio
`zeta`	n/a	[mm]	$\zeta = \sigma / r_v$	longitudinal offset conjugate with δ
`mass`	`nucm(j)`	[MeV/c^2]	m	mass
`mtc`	`mtc(j)`	[1]	$\frac{q}{q_0}\frac{m_0}{m}$	mass to charge ratio
`P`	`ejfv(j)`	[MeV/c]	P	momentum
`E`	`ejv(j)`	[MeV]	E	energy
`E0`	`e0`	[MeV]	E_0	reference energy
`P0`	`e0f`	[MeV/c]	P_0	reference momentum

Table 10.3: The following formats, set by the `format` option, are accepted:

#/Pos	Description
`0`	General format
Header	No header.
Lines	`turn structure_element_idx single_element_idx single_element_name s x[m] xp[rad] y[m] yp[rad] P[GeV/c] (E-E0)[eV] t[s]`

(The table continues on the next page)

#/Pos	Description
1	Format for aperture check
Header	`# particleID turn s[m] x[mm] xp[mrad] y[mm] yp[mrad] (E-E0)/E0[1] ktrack`
Lines	`particleID turn s[m] x[mm] xp[mrad] y[mm] yp[mrad] (E-E0)/E0 ktrack`
2	Modified format for aperture check
Header #1	(single element) `# DUMP format #2, bez=bez(i), number of particles=napx, dump period=ndumpt(i), first turn=dumpfirst(i), last turn=dumplast(i), HIGH=T/F, FRONT=T/F`
Header #1	(all elements) `# DUMP format #2, ALL ELEMENTS, number of particles=napx, dump period=ndumpt(i), first turn=dumpfirst(i), last turn=dumplast(i), HIGH=T/F, FRONT=T/F` Here bez is the name of the SINGLE ELEMENT, and napx the number of particles being tracked, ndumpt(i) the dump frequency as described above, and dumpfirst(i) and dumplast(i) the first and last turn as descirbed below. `HIGH` and `FRONT` is normally false, unless this (global) option is active, as described below.
Header #2	`# particleID turn s[m] x[mm] xp[mrad] y[mm] yp[mrad] sigma[mm] (E-E0)/E0[1] ktrack` If there are multiple single elements attached to the file, the headers are repeated.
Lines	As described in the header, one per particle and per turn.
3	Modified format for aperture check (Binary)
Header	No header. A number of Fortran records describing which elements are used and the current dump period is added one per relevant line in the `DUMP` block.
Lines	`particleID turn s[m] x[mm] xp[mrad] y[mm] yp[mrad] sigma[mm] (E-E0)/E0 ktrack` The Fortran code `SixTest/readDump3/readDump3.f90` can be used to convert these files into the Format 2 (sans headers).
4	Beam means
Header #1	Same as for Format 2.
Header #2	`# napx turn s[m] <x>[mm] <xp>[mrad] <y>[mm] <yp>[mrad] <sigma>[mm] <(E-E0)/E0>[1]` If there are multiple single elements attached to the file, the headers are repeated.
Lines	As described in the header; one per turn.
5	Beam mean and sigma
Header #1	The same as for format 2.
Header #2	`# napx turn s[m] <x>[mm] <xp>[mrad] <y>[mm] <yp>[mrad] <sigma>[mm] <(E-E0)/E0>[1] <x^2> <x*xp> <x*y> <x*yp> <x*sigma> <x*(E-E0)/E0> <xp^2> <xp*y> <xp*yp> <xp*sigma> <xp*(E-E0)/E0> <y^2> <y*yp> <y*sigma> <y*(E-E0)/E0> <yp^2> <yp*sigma> <yp*(E-E0)/E0> <sigma^2> <sigma*(E-E0)/E0> <((E-E0)/E0)^2>`

(The table continues on the next page)

#/Pos	Description
	If there are multiple single elements attached to the file, the headers are repeated. A number of lines describing which elements are used and the current dump period is added one per relevant line in DUMP block.
Lines	As described in the header; one per turn. For the "product" quantities, the units are the product of the units of the "normal" ones.
6	Beam mean and sigma (canonical)
Header #1	The same as for format 2.
Header #2	`# napx turn s[m] <x>[m] <px>[1] <y>[m] <py>[m] <sigma>[m] <psigma>[1] <x^2> <x*px> <x*y> <x*py> <x*sigma> <x*psigma> <px^2> <px*y> <px*py> <px*sigma> <px*psigma> <y^2> <y*py> <y*sigma> <y*psigma> <py^2> <py*sigma> <py*psigma> <sigma^2> <sigma*psigma> <psigma^2>` If there are multiple single elements attached to the file, the headers are repeated. A number of lines describing which elements are used and the current dump period is added one per relevant line in `DUMP` block.
Lines	As described in the header; one per turn. For the "product" quantities, the units are the product of the units of the "normal" ones. Note that $p_\sigma = \Delta E / (\beta_0 P_0 c)$. For more details, see the physics manual [16].
7	Normalized coordinates

	Dumps the particle trajectories in normalised coordinates. If the coordinates are dumped at the start of the sequence (`StartDUMP`), the normalization matrix as used for the initialization of the particle amplitudes is used. This means, that if 4D optics are chosen, the 4D matrix is used, if 6D optics is chosen, the matrix obtained from the 6D optics calculation is chosen. For every other element except `StartDUMP`, the 6D optics are used independent of the tracking method chosen. In this case the 6D optics needs to be run and the following lines have to be inserted in `fort.3`:
	`DUMP`
	`element_name_1 1 unit_1 7 filename_1 first_turn_1 last_turn_1`
	`...`
	`NEXT`
	`LINE`
	`ELEMENT  0 2 1 emit_1 emit_2`
	`NEXT`
	If there are multiple single elements attached to the file, the headers are repeated.

Header #1	The same as for format 2.
Header #2	Closed orbit x,x_p,y,y_p,σ,δ, units are [mm, mrad, mm, mrad, 1].
Header #3	Matrix of eigenvectors (TA Matrix or `tamatrix`). Eigenvectors are normalized, rotated and ordered as in the Ripken formalism and described in the SixTrack physics manual, Chapter "Optics Calculation". The matrix `tamatrix` is in canonical variables x,p_x,y,p_y,ζ,δ, units are [mm, mrad, mm, mrad, 1].
Header #4	Inverse of ta-matrix `inv(tamatrix)` used for normalization where $z_{\text{norm}} = \texttt{inv(tamatrix)} \cdot z$ (10.1) Matrix `inv(tamatrix)` and z is given in canonical variables x,p_x,y,p_y,ζ,δ, units are [mm, mrad, mm, mrad, 1].
Header #5	Header with units of normalized particle coordinates:

(The table continues on the next page)

#/Pos	Description
	`# particleID turn s[m] nx[1.e-3 sqrt(m)] npx[1.e-3 sqrt(m)] ny[1.e-3 sqrt(m)] npy[1.e-3 sqrt(m)] nsigma[1.e-3 sqrt(m)] ndp/p[1.e-3 sqrt(m)] ktrack`
Lines	As described in the header, one per particle and per turn.
`8`	Normalized coordinate (binary)
Header	No header. A number of Fortran records describing which elements are used and the current dump period is added one per relevant line in DUMP block. Format 8 is format 7 without header and in binary format.
Lines	`# particleID turn s[m] nx[1.e-3 sqrt(m)] npx[1.e-3 sqrt(m)] ny[1.e-3 sqrt(m)] npy[1.e-3 sqrt(m)] nsigma[1.e-3 sqrt(m)] ndp/p[1.e-3 sqrt(m)] ktrack` The Fortran code `SixTest/readDump3/readDump3.f90` can be used to convert these files into the Format 2 (sans headers).
`9`	Beam mean and sigma (normalized coordinates)
Header #1	The same as for format 2.
Header #2	`# napx turn s[m] <nx>[1.e-3 sqrt(m)] <npx>[1.e-3 sqrt(m)] <ny>[1.e-3 sqrt(m)] <npy>[1.e-3 sqrt(m)] <nsigma>[1.e-3 sqrt(m)] <npsigma>[1.e-3 sqrt(m)] <nx^2> <nx*npx> <nx*ny> <nx*npy> <nx*nsigma> <nx*npsigma> <npx^2> <npx*ny> <npx*npy> <npx*nsigma> <npx*npsigma> <ny^2> <ny*npy> <ny*nsigma> <ny*npsigma> <npy^2> <npy*nsigma> <npy*npsigma> <nsigma^2> <nsigma*npsigma> <npsigma^2>` If there are multiple single elements attached to the file, the headers are repeated. A number of lines describing which elements are used and the current dump period is added one per relevant line in `DUMP` block.
`101`	Binary format for debugging (format not stable)
Header	No header. A number of Fortran records describing which elements are used and the current dump period is added one per relevant line in the `DUMP` block.
Lines	`particleID turn s[m] x[mm] xp[mrad] y[mm] yp[mrad] z[mm] (E-E0)/E0 ktrack E[MeV] P[MeV/c] delta(j)[1] rpp[1] rvv[1] nucm[MeV] mtc[1] e0[MeV] e0f[MeV/c]`

Examples

```
DUMP
/ALL 1 663 2
/CRAB5 1 659 0
ip1 1 660 2 IP1_DUMP.dat
ip5 1 662 2
mqml.10l4.b1..1 1 661 2 MQ_DUMP.dat
NEXT
```

10.2 FMA Analysis

The FMA block generates the basic files needed for frequency map analysis (FMA). Explicitly, it returns one output file with calculated tunes and amplitudes for the files specified in the `DUMP` block, see Sec. 10.1. For the calculation of the tunes (Q_1, Q_2 and Q_3) in normalized phase space, the normalization matrix is extracted from the `LINE` block (linear optics calculation in 6D, 9.1). In case the particles are dumped at the beginning of the sequence (`StartDUMP`), the closed orbit and

normalization matrix used also for the initialization of the particles is used. In this case, the LINE block is not needed. The tunes Q_1, Q_2 and Q_3 are then calculated with the routine specified in the FMA block either in physical coordinates (x,x',y,y',z,dE/E) or normalized phase space coordinates and dumped to the file fma_sixtrack together with the minimum, maximum and average normalized particle amplitudes and phases.

To use normalised coordinates for the FMA analysis is always possible in case of 6D tracking (remember to put the LINE block for other elements than the start of the sequence). In case of 4D tracking, the following limitations apply:

- The FMA analysis is only implemented for the start of the sequence (StartDUMP). For other elements the normalization matrix would need to be obtained from the LINE block, which has not been checked in case of 4D optics.
- 4D tracking with scan in energy is disabled as in this case the normalization matrix would need to be saved for each element and particle, which requires a huge amount of memory breaking other parts of the code.

In general it is also recommended to already normalize the coordinates in DUMP as this is faster than in FMA.

Keyword	FMA
Data lines	Variable, one for each file with particle amplitudes and tune calculation method, and one for ea
Format	filename_1 method_1 (fma_flag_norm_1 (fma_first_turn fma_last_turn))
	OR NoNormDUMP

The FMA block has to be proceeded by the LINE block (calculation of the normalization matrix) and the DUMP block (dump particle coordinates).

```
DUMP
element_name_1 1 unit_1 2 filename_1 first_turn_1 last_turn_1
element_name_2 1 unit_2 2 filename_2 first_turn_2 last_turn_2
NEXT
LINE
ELEMENT  0 2 1 emit_1 emit_2
NEXT
FMA
filename_1 method_1 fma_flag_norm_1 fma_first_turn_1 fma_last_turn_1
filename_2 method_2 fma_flag_norm_2 fma_first_turn_2 fma_last_turn_2
NEXT
```

For the DUMP block (Sec. 10.1) the frequency has to be 1 (dump every turn) and the file format has to be 2 or 3. For the linear optics calculation 9.1, the optics needs to be calculated at each element (mode ELEMENT), the number-of-blocks is then 0 and 6D linear optics calculation is required (ilin = 2) in order to decouple the 6D motion.

Format Description

filename	One of the output files specified in the FMA block preceding DUMP block.
method	Method used to calculate the tune. Available methods are: TUNELASK, TUNEFIT, TUNENEWT1, TUNEABT, TUNEABT2, TUNEFFT, TUNEFFTI, TUNENEWT, TUNEAPA, NAFF. A short description of the different methods is given in Table 10.5.
fma_flag_norm	Optional flag for calculating the tunes with physical (x,x_p,y,y_p,ζ,δ) or normalized coordinates in case physical coordinates are used in DUMP. The default is using normalized coordinates (fma_flag_norm = 1). For using physical coordinates explicitly set (fma_flag_norm = 0). See **Description** for the conditions under which normalization is available.

`fma_first_turn`, `fma_last_turn`	Turns used for FMA analysis. As the `DUMP` files are used as input for the FMA analysis `fma_first_turn` must be larger `first_turn` in the `DUMP` block and `fma_last_turn` must be smaller than `last_turn` in the `DUMP` block. If `fma_last_turn = -1` the last turn number in the dump file is taken as the last turn number, including the last turn tracked if the `last` setting of the dump equals -1. By default, `FMA` will use the same turns as for the `DUMP`.
`NoNormDUMP`	A flag for disabling the `NORM_filename*` output files. This saves disk space and speeds up the calculation of the FMA. If used, the flag should be alone on a one line of the FMA input block in `fort.3`. Note that the capitalization must be correct for the flag to be recognized.

Output file format

The `FMA` block returns the output files `NORM_filename*` containing the normalized phase space coordinates, where `filename` are the filenames specified in the `DUMP` block, and the file `fma_sixtrack` containing the initial, average, minimum and maximum amplitudes and the calculated tunes for each specified filename and method. The structure of the `NORM_filename*` is described in Table 10.6 and of the `fma_sixtrack` in Table 10.7.

Table 10.5: Available tune calculation methods in SixTrack.

Library	Method	Description
PLATO [30, 31]	TUNELASK	Compute the tune of a 2d map by means of laskar method. A first indication of the position of the tune is obtained by means of a FFT. Refinement is obtained through a newton procedure.
	TUNEFIT	Computes the tune using a modified apa algorithm. The first step consists of taking the average of the tune computed with the APA method, then a best fit is performed.
	TUNENEWT1	Computes the tune using a discrete version of laskar method. It includes a newton method for the search of the frequency.
	TUNENEWT	Computes the tune using a discrete version of laskar method. It includes a newton method for the search of the frequency.
	TUNEABT	Computes the tune using FFT interpolated method.
	TUNEABT2	Computes the tune using the interpolated FFT method with hanning filter.
	TUNEFFT	Computes the tune as the FFT on a two dimensional plane, given n iterates of a map. The FFT is performed over the maximum mft which satifies $2^{\text{mft}} <= n$, where the maximum number of iterates is fixed in the parameter n.
	TUNEFFTI	Computes the tune as the FFT on a two dimensional plane, given n iterates of a map. The FFT is performed over the maximum mft which satifies $2^{\text{mft}} <= n$. Then, the FFT is interpolated fitting the three points around the maximum using a Gaussian. The tune is computed as the maximum of the Gaussian.
	TUNEAPA	Computes the tune as the average phase advance on a two dimensional plane, given n iterates of a map.
NAFF [32, 33]	NAFF	Computes the tune using the laskar method. The first estimation of the tune is obtained with an FFT and the precise value is determined by maximizing the Fourier integral. A Hann window of first and second order for the transverse and longitudinal motion are used respectively.

Table 10.6: Format of the NORM files

Line Number	Type	Description
1	Header	Closed orbit x,x',y,y',z,dE/E, units are [mm, mrad, mm, mrad, 1].
2–38	Header	Matrix of eigenvectors (tamatrix). Eigenvectors are normalized, rotated and ordered as in the Ripken formalism. The matrix tamatrix is in canonical variables x,p_x,y,p_y,ζ,δ, units are [mm, mrad, mm, mrad, 1].
39–75	Header	Inverse of ta-matrix inv(tamatrix) used for normalization where $z_{\text{norm}} = \text{ta} \cdot \text{z}$. Matrix inv(tamatrix) is given in canonical variables x,p_x,y,p_y,ζ,δ, units are [mm, mrad, mm, mrad, 1].
76	Header	Header with units: # id turn pos[m] nx[1.e-3 sqrt(m)] npx[1.e-3 sqrt(m)] ny[1.e-3 sqrt(m)] npy[1.e-3 sqrt(m)] nsig[1.e-3 sqrt(m)] ndp/p[1.e-3 sqrt(m)] kt
77–EOF	Lines	See header in line 76: particle id, turn number position s[m], normalized coordinates [$10^{-3}\sqrt{\text{m}}$], ktrack (type of element)

Table 10.7: Format of the fma_sixtrack file

Line Number	Type	Description
1–2	Header	Header with units and description: # eps0*,eps2*,eps3* all in 1.e-6*m, phi* [rad] # inputfile method id q1 q2 q3 eps1_min eps2_min eps3_min eps1_max eps2_max eps3_max eps1_avg eps2_avg eps3_avg eps1_0 eps2_0 eps3_0 phi1_0 phi2_0 phi3_0 norm_flag first_turn last_turn
3–EOF	Lines	See header in line 1-2: The lines are ordered as particles 1-npart for (inputfile1,method1), then particles 1-npart for (inputfile2,method2), etc.. The minimum (min), maximum (max) and average (avg) are taken over the number of turns in the inputfile (fiel specified in the FMA and DUMP block). Units are μm for eps* and rad for phi*, where phi* is the angle in the normalized phase space coordinates.

Example

An input block to compare the tunes at element IP3 calculated over the interval $[1, 4096]$ and $[5905, 10000]$, and using the method TUNELASK would look like:

```
DUMP
IP3 1 1030 2 IP3_DUMP_1 1 4096
IP3..1 1 1031 2 IP3_DUMP_2 5905 10000
IP3..2 1 1032 2 IP3_DUMP_3 1 4096
IP3..3 1 1033 2 IP3_DUMP_4 5905 10000
NEXT
LINE
ELEMENT  0 2 1 3.75 3.75
NEXT
FMA
IP3_DUMP_1 TUNELASK
IP3_DUMP_2 TUNELASK 1 512 1024
IP3_DUMP_3 TUNELASK 0
IP3_DUMP_4 TUNELASK 0 512 1024
NEXT
```

where for IP3_DUMP_1 and IP3_DUMP_2 the tunes are calculated using normalized coordinates (default) and for IP3_DUMP_3 and IP3_DUMP_4 the physical coordinates are used (fma_norm_flag = 0). For IP3_DUMP_2 and IP3_DUMP_4 the turns from 512 to 1024 are used for the FMA analysis. This is particularly useful for detecting the maximum diffusion in tunes by taking the maximum over difference over several moving windows.

Note that all element names have to be different due to a limitation in DUMP module. This means practically, that one needs to insert additional markers (here IP3..1 etc.) in the SixDesk [35, 36] mask file prior to the SixTrack run. It is important to install the additional markers after cycling the machine if the machine is cycled at the location of the additional (e.g. IP3), as they are installed in front of the element given in the from statement in the cycle command.

10.3 File Hash

The hash module can optionally compute the MD5 digest of a selected set of output files listed in a HASH block. The hash values are written to the output file hash.md5. The hash values are computed just before the ZIPF file compression routine is called, so the hash file can be included in the final archive.

During initialisation, the module will perform a self test to ensure it generates valid md5 hashes according to the RFC1321 standard. If this is not achieved, the module is disabled, and the hash.md5 file contains an error message instead.

The primary purpose of this module is for validation of results for BOINC and the test suite.

Keyword HASH
Data lines Variable, one for each file to be hashed.

Usage

The module accepts a single keyword, MD5SUM, followed by a file name, followed by either "text" or "binary". This can be repeated multiple times. The "text" flag is used on Windows to strip carriage returns from the file before hashing. If the md5sum tool is installed on the user's computer, the results can be validated with the command:

```
md5sum -c hash.md5
```

Example

```
HASH
  MD5SUM final_state.dat text
  MD5SUM fort.10         text
NEXT
```

10.4 ZIPFile Combined and Compressed Output

In order to retrieve extra simulation output such as DUMP or FMA from BOINC, it is necessary to pack the output files into a single file with a special name that will be retrieved. This can be achieved with the ZIPF block, which packs the listed files into a compressed archive at the end of the simulation. The ZIPF block can always be present in fort.3, but if SixTrack was not built with the ZLIB option, no archive file will be produced.

The ZIPF block can take two optional arguments, otherwise everything else is interpreted as a file name to be packed into the archive file. Multiple file names can be specified on a single line, and are separated by a space character. If the file name contains a space, it must be wrapped in single or double quotes.

Note that if one of the files do not exist at the end of the simulation, it will be silently skipped and not included in the archive.

Keyword `ZIPF`
Data lines Variable, see below.

Archive Name `OUTFILE`

This keyword can be used to set the file name of the archive file. If this keyword is omitted, the archive file will be named "Sixout.zip".

Compression Level `ZIPLEVEL`

This keyword can be used to set the compression level. A value of 0 will just pack the files, and not compress them. A value of 9 is the highest level of compression, and also the slowest, The default value is 3.

Example

```
ZIPF
    OUTFILE Sixout.zip
    ZIPLEVEL 3
    fma_sixtrack IP3_DUMP_1 fort.90
NEXT
```

10.5 HDF5 Output

The `HDF5` block allows for writing certain outputs to a HDF5 file instead of regular text or binary files. HDF5 files can be easily read and manipulated with for instance MATLAB or Python. MATLAB has native support, while Python support is available through h5py.

The SixTrack HDF5 option is enabled through the `HDF5` compiler flag, and controlled via the `HDF5` block.

Note: SixTrack HDF5 support is experimental.

Keyword `HDF5`
Data lines Variable, see below.
Format This module uses a keyword, value format. See below.

Debugging `DEBUG`

This statement switches on extra "debugging" output for the `HDF5` module. This can be useful if debugging the code or if debugging the input.

Precision `SINGLE`, `DOUBLE`

The precision of float numbers for the file. If omitted, the value defaults to `DOUBLE`.

The output precision is independent of the internal precision of SixTrack set at compile time. If necessary, the float values will be converted on the fly. Quad precision is currently not available.

The precision of integers is the same as the internal Fortran precision defined by the compiler. Generally, this is 32 bits.

Output File `FILE filename truncate`

The name of the file to write to. Spaces are allowed as long as quote marks are used. The truncate flag is optional, either `.true.` or `.false.`. If true, any existing file will be truncated. If false, any existing file will throw an error. Default value is `.false.`. If truncation is disabled, and the file exists, the root group must be unique for the current run. This allows the option to write multiple simulation runs to the same file with different root groups.

Root Group `ROOT groupname`

The name of the root group (folder) for where to write the simulation data. The default value is "/", that is, all data is written into block specific groups at the root of the file. Setting root group allows several runs to use the same output file as long as the root group is unique.

For further information on how HDF5 uses groups and datasets, see the HDF5 manual [17].

Chunking `CHUNK chunksize`

HDF5 files written by SixTrack uses data chunking. Chunking allows for writing data into related block. For instance, for `DUMP`, the chunck size is hard coded to the nuber of particles. This can improve read performance as the particle data will then be written in a single chunk per turn. For non-predictable outputs, like log files, a default chunk value can be set. The chunk size should be close to the number of entries expected to be written per turn. If none is specified, the defualt value is 10.

For further information on HDF5 chunking, see the HDF5 manual [17].

Compression `GZIP level`

The level of compression to use for data chunks written to the HDF5 file. Allowed values are -1 to disable gzip compression, and 0 to 9 for none to maximum compression.

`0`	No compression
`1`	Best compression speed; least compression
`2-8`	Compression improves; speed degrades
`9`	Best compression ratio; slowest speed

Note that 0 does not turn off use of the gzip, it just instructs the filter to perform no action. To disable GZIP, either ommit the line, or set the level to -1. For more detail, see the HDF5 manual [17].

Enable HDF5 `ENABLE blockname`

HDF5 output needs to be specifically enabled for the blocks where it is to be used instead of ASCII or binary data dumps. The `blockname` takes the four first characters of the block for which to enable HDF5. An further characters are ignored, but may be used for clarity like for othe rblock declarations. In other words, `ENABLE SCAT` and `ENABLE SCATTER` are equally valid.

HDF5 output is currently available only for `SCATTER`, `DUMP`, `APERTURE` and `COLLIMATION`.

Write Flag `WRITE type`

Certain special outputs are possible through the `WRITE` flag:

`OPTICS` Dumps the linear optics to the root group of the file.

`TRACKS2` Writes the collimation tracks2 output to the root group of the file.

Example:

The following is an example of a valid `HDF5` block:

```
HDF5
    DEBUG
    DOUBLE
    GZIP 1
    CHUNK 50
    FILE data.hdf5 .true.
    ROOT test
    ENABLE SCATTER
    ENABLE DUMP
    WRITE OPTICS
NEXT
```

10.6 ROOT Output

Keyword `ROOT`
Data lines Variable, see below.
Format This module uses a keyword, value format. See below.

Select output to enable `ENABLE type`

The `ENABLE` flag selects which output to enable for root. Since a large run or parameter scan usually uses multiple copies of SixTrack, a lot of information is the same between each run, e.g. the accelerator layout. It is suggested to only enable the full output on a single run, and reduced output on all others.

`type` char The type of output to enable.

`ALL`	Writes all possible output.
`ACCEL`	Writes a description of the accelerator layout
`COLL`	Writes the energy and nucleons lost on each collimator.
`COLDB`	Writes the contents of the collimation database.
`APER`	Writes particles lost on the aperture.
`OPTICS`	Writes the linear optics of the machine.
`FLUKA`	Writes additional information on insertion lengths required for usage with the FLUKA coupling.
`PIPE`	Writes a description of the physical beam pipe from the aperture module.

Enable writing directly to eos `EOS`

The `EOS` flag enables writing directly to the CERN eos filesystem directly via xrootd instead of as normal files.

Selection of the output file path `PATH loc`

The `PATH` flag selects the path (folder) of where the output files should be written to.

`loc` char The folder where output should be written to.

Selection of the output file name `PREFIX pre`

The `PREFIX` flag selects the first part of the output filename. Added to this is the run number (defined below). This is then followed by the file extension ".root".

`pre` char The first part of the output file name.

Selection of the output run number `RUN number`

The `RUN` flag selects the run number to be added to the root output file name. This is usually the random number seed.

`number` integer The output run number.

10.7 Simulation Meta Data and Timing Output

SixTrack produces two text files containing meta data and timing data for the last run simulation. Both these files are in a fixed column width format, making them straightforward to parse with other tools.

META: The file `sim_meta.dat` contains information about the SixTrack executable such as version information, build details and execution time. The file also contains initial values and values calculated during tracking that may be useful for further post-processing.

TIME: The file `sim_time.dat` contains information about the execution time of the last SixTrack run. There are a number of time stamp at key points during execution written to the file. In addition, the total tracking time is extracted, and various averages computed based on number of particles, turns or the size of the lattice.

Appendix

Appendix A

List of Default Values

A.1 Default Tracking Parameters

Some of the parameters for tracking are set to non-zero values. This is done for instance to avoid as much as possible program errors such as division by zero due to an erroneous input. The default values for the *Iteration Errors* are described in Section 3.6, Table 3.1.

Table A.1: Default Tracking Parameters

#	Description	Value	§	Page
1	General Aperture Limitations (horizontal and vertical)	1000 mm	6.3	40
2	Starting in the Accelerator Structure at Element Number	1	5.2.1	36
3	Number of Turns in the forward Direction	1	4.2	13
4	Initial horizontal Amplitude	0.001 mm		
5	Horizontal and vertical Phase Space Coupling Switches on	1		
6	Flat Bottom, Ramping and Flat Top Printout after Turn Number	1		
7	Printout of Coordinates (file 6) after Turn Number	10000		
8	Kinetic Energy [MeV] of the Reference Particle	10^{-6}	4.3	16
9	Harmonic Number	1	4.4	18
10	Momentum Compaction Factor	0.001		
11	Length of the Machine	1 km		
12	Mass of the Particle (Proton)	938.2723128 MeV/c^2		
13	Momentum Correction Factor for Distance in Phase Space	1		
14	Path-length Correction Factor for Distance in Phase Space	1		
15	Averaging Turn Interval for Post-processing	1	9.12	98

A.2 Default Size Parameters

SixTrack 5 automatically scales the arrays related to the machine description and number of tracked particles, while earlier versions used fixed size arrays. However, many other parameters are still fixed and are defined as parameters in the module `parpro` in the source code. They are generally large enough for their intended purpose, but if these are found to be too small, they can be adjusted in the source at the top of the file `source/common_modules.f90`.

Table A.2: Default Size Parameters

#	Description	Value	Name	§	Page
1	Maximum Number of Coordinates used in the Correction Routines	6	`MPA`		
2	Number of Single Elements	auto	`NELE`	5.1	27
3	Number of Blocks of Linear Elements	auto	`NBLO`	5.2	35
4	Number of Linear Elements per Block	280	`NELB`		
5	Total Number of Elements in the Structure	auto	`NBLZ`	5.2.1	36
6	Number of Accelerator Super-periods	16	`NPER`		
7	Total Number of Random Values	auto	`NZFZ`	8.1	87
8	Number of Random Values for the basic Set of Nonlinear Elements	2000000	`NRAN`		
9	Number of Random Values for inserted Nonlinear Elements	20000		8.2	88
10	Number of Random Values for each Inserted Nonlinear Element Number of Nonlinear Elements that can be inserted	500 20	`MRAN`		
11	Limit Number of Particles for Vectorisation	auto	`NPART`		
12	Maximum Number of Elements for Combined Tasks	100	`NCOM`	8.3	89
13	Maximum Resonance Compensation Order	5	`NRCO`	8.3	89
14	Total Number of Data for Processing	20000	`NPOS`	9.12	98
15	Number of Intervals for Calculation of Lyapunov Exponents	10000	`NLYA`		
16	Number of Intervals for Calculation of Invariants	1000	`NINV`		
17	Number of Data for Plotting	20000	`NPLO`		
18	Maximum Pole Order of Multipole Block	20	`MMUL`	6.1	39
19	Maximum Number of extra Parameters of the DA Map	10	`MCOR`	9.9	96
20	Maximum Order of DA Calculation	15	`NEMA`	9.9	96
21	Maximum Number of Monitors for Micado Closed Orbit Correction	600	`NMON1`	9.4	93
22	Maximum Number of Correctors for Micado Closed Orbit Correction	600	`NCOR1`	9.4	93
23	Maximum Number of Beam–Beam Elements	500	`NBB`	6.6	54
24	Maximum Number of Slices for 6D Beam–Beam Kick	99	`MBEA`	6.6	54
25	Maximum Number of “Phase Trombone” Elements	auto	`NTR`	5.1.9	31

Appendix B

Input and Output Files

The program uses a couple of files for its input and output procedures.

Table B.1: List of Input and Output Files.

File Unit	Input	Output	File Type	Contents
2	✓		Ascii	Geometry and Strength Parameters
3	✓		Ascii	Tracking Parameters
4		✓	Ascii	Geometry and strength Parameters (format as file `fort.2`)
6		✓	Ascii	Standard Output
8	✓		Ascii	Name, hor., ver. Misalignment and Tilt
9		✓	Ascii	Internally used multipoles Format: $a16,\ 2 \times \{6 \times (1p, 3d23.15), (1p, 2d23.15)\}$
10	✓	✓	Ascii	Summary of Post-processing (auxiliary)
12		✓	Ascii	End coordinates of all particles. Format: $(15 \times F10.6)$
13	✓		Ascii	Start coordinates for a prolongation
14		✓	Ascii	Horizontal FFT spectrum for detailed analysis. Format: $(2 \times F10.6)$
15		✓	Ascii	Vertical FFT spectrum for detailed analysis. Format: $(2 \times F10.6)$
16	✓		Ascii	External multipole errors. Format: $a16,\ 2 \times \{6 \times (1p, 3d23.15), (1p, 2d23.15)\}$
17		✓	Ascii	Additional Map at location of interest
18		✓	Ascii	One turn map with differential algebra
19	✓	✓	Ascii	Internal use for Differential Algebra
20		✓	Meta-file	PS-file of selected Plots
21		✓	Ascii	Factorisation of the one turn map

File Unit	Input	Output	File Type	Contents
22		✓	Ascii	Transformation in the Normal Form coordinate
23		✓	Ascii	Hamiltonian in action variables
24		✓	Ascii	Tune-shift in action coordinates
25		✓	Ascii	Tune-shift in Cartesian coordinates
26		✓	Binary	Binary version of unit 18
27		✓	Ascii	Name, hor., ver. misalignment and tilt
28		✓	Ascii	Horizontal closed orbit displacement, measured at monitors
29		✓	Ascii	Vertical closed orbit displacement, measured at monitors
30	✓		Ascii	Name, random strength, misalignments and tilt
31		✓	Ascii	Name, random strength, misalignments and tilt
33	✓		Ascii	Guess values for 6D closed orbit search
90		✓	Binary	Tracking Data `singletrackfile.dat`

In addition to those files listed in the table, DUMP uses arbitrary file unit numbers as determined by the input file. The collimation module also uses many input/output files at various units, which are not listed here.

Appendix C

Data Structure of the Data Files

A common data structure for the programs MAD-X and SixTrack is agreed on[1]. Besides some minor differences this allows a straightforward post-processing of data from either program. Each binary data file has a header which holds a description of the run with comments, tracking parameters and 50 additional parameters for future purposes, six of which are already specified in SixTrack.

[1] Historically, SixTrack wrote one such file per particle pair (files `fort.90`, `fort.89`, ...), however recent versions of SixTrack always produce a single `singletrackfile.dat` containing all the data, by interleaving the contents of the old files in one large file as discussed below. Files of the old (MAD-X and Sussix compatible) format can be generated from this new format by running the tool `splitSingletrack` in the folder containing the `singletrackfile.dat`. This splits the combined `singletrackfile.dat` into one file for each pair.

This chapter describes the old format used for single particle pairs, followed by a description of the built-in postprocessing tool output.

Table C.1: Header of the Binary Data Files

Data Type	Bytes	Description
Character	80	General title of the run
Character	80	Additional title
Character	8	Date
Character	8	Time
Character	8	Program name
Integer	4	First particle in the file
Integer	4	Last particle in the file
Integer	4	Total number of particles
Integer	4	Code for dimensionality of phase space 1,2,4 are hor., vert. and longitudinal respectively
Integer	4	Projected number of turns
Float	8	Horizontal Tune
Float	8	Vertical Tune
Float	8	Longitudinal Tune
Float	6 * 8	Closed Orbit vector
Float	6 * 8	Dispersion vector
Float	36 * 8	Six-dimensional transfer map
50 additional parameters		
Float	8	Maximum number of different seeds
Float	8	Actual seed number
Float	8	Starting value of the seed
Float	8	Number of turns in the reverse direction (IBM only)
Float	8	Correction factor for the Lyapunov ($\sigma = s - v_0 \times t$)
Float	8	Correction factor for the Lyapunov ($\Delta p/p_0$)
Float	8	Start turn number for ripple prolongation
Float	43 * 8	Dummies

Following this header the tracking data are written in n samples of mine numbers preceded by the turn number. In the MAD-X format, the number of samples in is not restricted, whilst SixTrack writes only up to two samples for the two particles for the Lyapunov exponent method. Up to 64 particles (two per file) can be treated in the vectorised version of SixTrack.

Table C.2: Format of the Binary Data

Data Type	Bytes	Description
Integer	4	Turn number
One or two samples of 9 values are following		
Integer	4	Particle number
Float	8	Angular distance in phase space (<= 1)
Float	8	x (mm)
Float	8	x' (mrad)
Float	8	y (mm)
Float	8	y' (mrad)
Float	8	Path-length ($\sigma = s - v_0 \times t$) (mm)
Float	8	Relative momentum deviation $\Delta p/p_0$
Float	8	Energy (MeV)

Note that in case the "Single Track File" option is enabled at compile time, multiple of these files (normally one per particle pair) are interleaved in a single file. This is done by writing first all headers in order (i.e. first the header for initial particle/final particle 1/2, then 3/4, 5/6 etc.) and then the same for the tracking data. The "total number of particles" field can always be read from the first header record, which gives the number of header records present in the file. The two file formats are equivalent, i.e. they contain exactly the same data, and it is thus possible to convert losslessly between them.

Some of the post processing data is written in Ascii format to file `fort.10`. This can be used for instance for plotting purposes. Each time the post processing routine is called 60 double precision numbers (some of them still dummy) are added to the file.

The file with the errors (in: `fort.16`, out: `fort.9`) has the following format:

first line name of element;
line 2–7 normal multipoles order 1–18;
line 8 normal multipoles of order 19 and 20;
line 9–14 skew multipoles order 1–18;
line 15 skew multipoles of order 19 and 20.

The strength definition is according to block 6.1 and to be effective in `fort.3`. The random values of the corresponding multipole block have to be set to 1.0. A word of caution: when writing on file `fort.9` the *total* multipole strength is used, i.e. systematic and random part combined. File `fort.16` and `fort.9` might therefore be different. When using `fort.9` as input (`fort.16`), the systematic part in `fort.3` has to be set to 0.0.

Misalignment and tilt are in file `fort.8` and `fort.27` as input and output respectively. The format is (`a16,2x,1p,2d14.6,d17.9`), i.e. name, horizontal misalignment, vertical misalignment and tilt. The misalignment is in units of [mm] the tilt in units of [mrad]. The files `fort.30` (in) and `fort.31` (out) have the random single non-linear element kick, misalignments and tilt in the format: (`a8,1p,d19.11,2d14.6,d17.9`). Misalignment and tilt in file `fort.8` or `fort.30` is automatically activated, while the random strength (strength definition same as in block 5.1) needs an entry in the fourth column in the geometry file `fort.2`. Files `fort.28` and `fort.29` hold integer counter and closed orbit displacement at a horizontal or vertical monitor respectively.

Table C.3: Post Processing Data

Column	Description
1	Maximum turn number
2	Stability Flag (0=stable, 1=lost)
3	Horizontal Tune
4	Vertical Tune
5	Horizontal β-function
6	Vertical β-function
7	Horizontal amplitude 1^{st} particle
8	Vertical amplitude 1^{st} particle
9	Relative momentum deviation $\Delta p/p_0$
10	Final distance in phase space
11	Maximum slope of distance in phase space
12	Horizontal detuning
13	Spread of horizontal detuning
14	Vertical detuning
15	Spread of vertical detuning
16	Horizontal factor to nearest resonance
17	Vertical factor to nearest resonance
18	Order of nearest resonance
19	Horizontal smear
20	Vertical smear
21	Transverse smear
22	Survived turns 1^{st} particle
23	Survived turns 2^{nd} particle
24	Starting seed for random generator
25	Synchrotron tune
26	Horizontal amplitude 2^{nd} particle
27	Vertical amplitude 2^{nd} particle
28	Minimum horizontal amplitude
29	Mean horizontal amplitude
30	Maximum horizontal amplitude
31	Minimum vertical amplitude
32	Mean vertical amplitude
33	Maximum vertical amplitude
34	Minimum horizontal amplitude (linear decoupled)
35	Mean horizontal amplitude (linear decoupled)
36	Maximum horizontal amplitude (linear decoupled)
37	Minimum vertical amplitude (linear decoupled)
38	Mean vertical amplitude (linear decoupled)
39	Maximum vertical amplitude (linear decoupled)
40	Minimum horizontal amplitude (nonlinear decoupled)
41	Mean horizontal amplitude (nonlinear decoupled)
42	Maximum horizontal amplitude (nonlinear decoupled)
43	Minimum vertical amplitude (nonlinear decoupled)

Column	Description
44	Mean vertical amplitude (nonlinear decoupled)
45	Maximum vertical amplitude (nonlinear decoupled)
46	Emittance Mode I
47	Emittance Mode II
48	Secondary horizontal β-function
49	Secondary vertical β-function
50	Q'_x
51	Q'_y
52–58	Dummy
59–60	Internal use

As an option the 4D linear parameters can be dumped to file `fort.11` when the linear optics block 9.1 is activated. This can be used for instance for a post-processing of linear coupling. 25 values are written in a binary format.

Table C.4: 4D Linear Parameters

Column	Description
1	Name of the element
2	Longitudinal Position [m]
3	Horizontal phase advance
4	Vertical phase advance
5	Primary horizontal β-function [m]
6	Secondary horizontal β-function [m]
7	Secondary vertical β-function [m]
8	Primary vertical β-function [m]
9	Primary horizontal α-function [rad]
10	Secondary horizontal α-function [rad]
11	Secondary vertical α-function [rad]
12	Primary vertical α-function [rad]
13	Primary horizontal γ-function [m]
14	Secondary horizontal γ-function [m]
15	Secondary vertical γ-function [m]
16	Primary vertical γ-function [m]
17	Primary horizontal phase of x-coordinate [pi]
18	Secondary horizontal phase of x-coordinate [pi]
19	Secondary vertical phase of y-coordinate [pi]
20	Primary vertical phase of y-coordinate [pi]
21	Primary horizontal phase of x'-coordinate [pi]
22	Secondary horizontal phase of x'-coordinate [pi]
23	Secondary vertical phase of y'-coordinate [pi]
24	Primary vertical phase of y'-coordinate [pi]
25	Coupling angle [pi]

Column	Description
26	D_x [mm]
27	D'_x [mrad]
28	D_y [mm]
29	D'_y [mrad]

When external multipole errors are read in (see section 8.1), the program expects a complete list of magnet errors to file `fort.16`. The format of each set of multipole errors is given in table C.5. The definition of the multipole coefficients should be as described in section 6.1.

Table C.5: Format of file with external errors, `fort.16`, and internal errors written to `fort.9`

Row	Description
1	Name of multipole set
2	B_1 B_2 B_3
3	B_4 B_5 B_6
4	B_7 B_8 B_9
5	B_{10} B_{11} B_{12}
6	B_{13} B_{14} B_{15}
7	B_{16} B_{17} B_{18}
8	B_{19} B_{20}
9	A_1 A_2 A_3
10	A_4 A_5 A_6
11	A_7 A_8 A_9
12	A_{10} A_{11} A_{12}
13	A_{13} A_{14} A_{15}
14	A_{16} A_{17} A_{18}
15	A_{19} A_{20}

With the parameter `mout` set to 2 or 3 in the "Random Fluctuation" block (8.1), the internally used multipoles are written to file `fort.9` in the same format as above. This file can therefore be used as an input `fort.16` file for a subsequent run.

The file `fort.34` is written when the "Linear Optic Block" (see section 9.1) is invoked with the `ELEMENT 0` option.

Table C.6: Format of file `fort.34` for detuning and distortion calculation with external program "SODD" [22]

Column	Description
1	Longitudinal position [m]
2	Type `n` of Multipole ($n > 0$ => erect, $n < 0$ => skew)
3	Multipole strength [mrad $\cdot$ mm$^{(1-\lvert n\rvert)}$]
4	Horizontal β-function [m]

Column	Description
5	Vertical β-function [m]
6	Horizontal phase advance
7	Vertical phase advance

The last line serves as the end of the structure: Length of the accelerator, fake name `END`, fake type `100`, β functions and phase advances at the end of the accelerator for the horizontal and vertical plane respectively.

Bibliography

[1] LBL diffential algebra package and LieLib routines courtesy of É. Forest.

[2] G. Ripken and F. Schmidt, "A symplectic six-dimensional thin-lens formalism for tracking", CERN SL 95–12 (AP)(1995), DESY 95–063 (1995);
G. Ripken and F. Schmidt, "Construction of Nonlinear Symplectic Six-Dimensional Thin-Lens Maps by Exponentiation", DESY 95–189 (1995), http://cern.ch/Frank.Schmidt/report/ripken2.pdf;
D.P. Barber, K. Heinemann, G. Ripken and F. Schmidt, "Symplectic Thin - Lens Transfer Maps for SixTrack: Treatment of Bending Magnets in Terms of the Exact Hamiltonian", DESY 96–156 (1995), http://cern.ch/Frank.Schmidt/report/ripken3.pdf.

[3] A. Wrulich, "RACETRACK, A computer code for the simulation of nonlinear motion in accelerators", DESY 84–026 (1984).

[4] B. Leemann and É. Forest, "Brief description of the tracking codes FASTRAC and THINTRAC", SSC Note SSC–133.

[5] G. Ripken, "Nonlinear canonical equations of coupled synchro-betatron motion and their solution within the framework of a nonlinear 6-dimensional (symplectic) tracking program for ultra-relativistic protons", DESY 85–084 (1985).

[6] D.P. Barber, G. Ripken and F. Schmidt, "A nonlinear canonical formalism for the coupled synchro-betatron motion of protons with arbitrary energy", DESY 87–036 (1987);
G. Ripken and F. Schmidt, "A symplectic six-dimensional thin-lens formalism for tracking", CERN/SL/95–12 (AP), DESY 95–063 (1995), http://cern.ch/Frank.Schmidt/report/ripken.pdf;

[7] R. Brun and D. Lienart, "HBOOK User Guide", CERN Program Library Y250 (1987).

[8] R. Brun and N.C. Somon, "HPLOT User Guide", CERN Program Library Y251 (1988).

[9] R. Bock, R. Brun, O. Couet, N.C. Somon, C.E. Vandoni and P. Zanarini, "HIGZ User Guide", CERN Program Library Q120.

[10] G. Guignard, "A general treatment of resonances in accelerators", CERN 78–11 (1978).

[11] M. Berz, "Differential algebra description of beam dynamics to very high orders", Particle Accelerators, 1989, Vol. 24, pp. 109–124.

[12] M. Berz, "DAFOR – Differential Algebra Precompiler Version 3, Reference Manual", MSUCL–755 (1991).

[13] F. Schmidt and M. Vaenttinen, "Vectorisation of the single particle tracking program SixTrack", CERN SL Note 90–20 (1990) (AP).

[14] F. Schmidt, "Untersuchungen zur dynamischen Akzeptanz von Protonenbeschleunigern und ihre Begrenzung durch chaotische Bewegung", DESY HERA 88–02, (1988).

[15] H. Grote, "A MAD–SixTrack interface", SL Note 97–02 (AP).

[16] SixTrack Physics Manual, http://sixtrack.web.cern.ch/SixTrack/

[17] HDF5 Software Documentation, https://support.hdfgroup.org/HDF5/doc/H

[18] M. Berz, É. Forest and J. Irwin, "Normal form methods for complicated periodic systems: a complete solution using differential algebra and lie operators", Particle Accelerators, 1989, Vol. 24, pp. 91–107.

[19] M. Bassetti and G.A. Erskine, "Closed expression for the electrical field of a two-dimensional Gaussian charge", CERN–ISR–TH/80–06.

[20] K. Hirata, H. Moshammer, F. Ruggiero and M. Bassetti, "Synchro-Beam interaction", CERN SL-AP/90-02 (1990) and Proc. Workshop on Beam Dynamics Issues of High-Luminosity Asymmetric Collider Rings, Berkeley, 1990, ed. A.M. Sessler (AIP Conf. Proc. 214, New York, 1990), pp. 389-404;
K. Hirata, H. Moshammer and F. Ruggiero, "A symplectic beam-beam interaction with energy change", KEK preprint 92-117 A (1992) and Part. Accel. 40, 205-228 (1993);
K. Hirata, "BBC User's Guide; A Computer Code for Beam-Beam Interaction with a Crossing Angle, version 3.4", SL-Note 97-57 AP.

[21] L.H.A. Leunissen, F. Schmidt and G. Ripken, "6D Beam–Beam Kick including Coupled Motion", LHC Project Report 369, http://cern.ch/Frank.Schmidt/report/ripken_new.pdf.

[22] F. Schmidt, "SODD: A Computer Code to calculate Detuning and Distortion Function Terms in First and Second Order", CERN SL/Note 99–009 (AP), http://cern.ch/Frank.Schmidt/report/sodd_manual.pdf

[23] H. Grote and F.C. Iselin, "The MAD program (Methodical Accelerator Design), Version 8.10, User's Reference Manual", CERN SL 90–13 (AP) (Rev. 4), http://cern.ch/Hans.Grote/mad/mad8/doc/mad8_user.ps.gz.

[24] R. Molloy and S. Blitz, "Fringe Field Effects on Bending Magnets, Derived for, TRANSPORT/TURTLE", FERMILAB-TM-2564-AD-APC-PPD, http://lss.fnal.gov/archive/test-tm/2000/fermilab-tm-2564-ad-apc-ppd.pdf

[25] private communication.

[26] F. James, "A Review of Pseudorandom Number Generators", CERN DD/ 88/22, 1988.

[27] B. Autin and Y. Marti, "Closed Orbit Correction of A.G. Machines Using a Small Number of Magnets", CERN–ISR–MA/73–17.

[28] M. Giovannozzi, "Description of software tools to perform tune-shift correction using normal forms", CERN SL Note 93–111 (AP).

[29] F. Schmidt, F. Willeke and F. Zimmermann, "Comparison of methods to determine long-term stability in proton storage rings", 1991, Particle Accelerators, Vol. 35, pp. 249–256.

[30] R. Bartolini, A. Bazzani, M. Giovannozzi, W. Scandale, E. Todesco, "Tune evaluation in simulations and experiments", Part. Accel. 52 147

[31] M. Giovannozzi, E. Todesco, A. Bazzani and R. Bartolini (1997), "PLATO: a program library for the analysis of nonlinear betatronic motion", Nucl. Instrum. and Methods A 388 1

[32] J. Laskar, C. Froeschle and C. Celletti, "The measure of chaos by the numerical analysis of the fundamental frequencies. Application to the standard mapping", Physica D, vol. 56, pp 253-269, 1992.

[33] S. Kostoglou, N. Karastathis, Y. Papaphilippou, D. Pellegrini and P. Zisopoulos, "Development of computational tools for noise studies in the LHC", 2017, Proceedings of IPAC'17, Copenhagen, Denmark, 2017.

[34] SixTrack build manual, see SixTrack website, http://sixtrack.web.cern.ch/SixTrack/

[35] SixDesk manual, see SixTrack website, http://sixtrack.web.cern.ch/SixTrack/

[36] SixDesk manual, https://www.overleaf.com/1345694dwypbp#/3325092/

[37] J. B. Garcia et al., "Long term dynamics of the high luminosity Large Hadron Collider with crab cavities", 2016, PHYSICAL REVIEW ACCELERATORS AND BEAMS 19, 101003 (2016).

[38] K. Sjobak, H. Burkhardt, R.D. Maria, A. Mereghetti and A. Santamaria, "General functionality for turn-dependent element properties in SixTrack", 2015, Procedings of IPAC'13, Richmond, VA, USA, May 2015.

[39] K. Sjobak, V.K. Berglyd Olsen, R. De Maria, M. Fitterer, A. Santamara Garca, H. Garcia-Morales, A. Mereghetti, J.F. Wagner, S.J. Wretborn, "Dynamic simulations in SixTrack", CERN

[40] S. Russenschuck, "Field computation for Accelerator Magnets", Wiley-VCH, 2010

[41] P. Burla, Q. King and J.G. Pett, "Optimisation of the current ramp for the LHC", Proceedings of the 1999 Particle Accelerator Conference, New York, 1999.

[42] T. Trenkler, J.B. Jeanneret, "K2, A software package evaluating collimation systems in circular colliders (manual)", CERN SL/94105 (AP), 1994.

[43] G. Robert-Demolaize, R. Assmann, S. Redaelli, F. Schmidt, "A new version of SixtTrack with collimation and aperture interface", CERN, Geneva, Switzerland (PAC 2005).

[44] R. Assmann, J.B. Jeanneret, D. Kaltchev, "Status of Robustness Studies for the LHC Collimation", APAC 2001.

[45] T. Sjstrand, S. Mrenna, P. Skands, "A Brief Introduction to PYTHIA 8.1 ", Comput. Phys. Comm. 178 (2008) 852 [arXiv:0710.3820].

[46] T. Sjstrand, S. Mrenna, P. Skands, "PYTHIA 6.4 Physics and Manual", JHEP05 (2006) 026.

[47] V. Vlachoudis *et al.*, "Status of Fluka coupling to Sixtrack", Proceedings of Tracking for Collimation Workshop (WP5), CERN, Geneva, Switzerland (in publication).

[48] A. Mereghetti, Performance Evaluation of the SPS Scraping System in View of the High Luminosity LHC, Ph. D. thesis, UniMAN, Manchester, UK (2015).

[49] B. Dalena *et al.*, "Fringe Field Modeling for the High Luminosity LHC Large Aperture quadrupole", Proceedings of IPAC14, Dresden, Germany, June 2014, paper TUPRO002, pp. 993–996.

[50] T. Pugnat *et al.*, "Accurate and Efficient Tracking in Electromagnetic Quadrupoles", Proceedings of IPAC18, Vancouver, Canada, June 2014, paper THPAK004, pp. 3207.

[51] A. Simona *et al.*, "High order time integrators for the simulation of charged particle motion in magnetic quadrupoles", Elsevier, February 2019.

[52] F. James, "RANLUX: A FORTRAN Implementation of the High Quality Pseudorandom Number Generator of Luscher". Comput.Phys.Commun. 79 (1994): 11114.DOI:10.1016/0010-4655(94)90233-X.

[53] F. James, "A Review of Pseudorandom Number Generators", Computer Physics Communications 60, no. 3 (1 October 1990): 32944. DOI:10.1016/0010-4655(90)90032-V.

[54] P. LEcuyer, "Efficient and Portable Combined Random Number Generators", Commun. ACM 31, no. 6 (June 1988): 742751. DOI:10.1145/62959.62969.

[55] V. Previtali, "Performance Evaluation of a Crystal-enhanced Collimation System for the LHC", Ph. D. thesis, EPFL, Lausanne, Switzerland (2010).

[56] D. Mirarchi, "Crystal collimation for LHC", Ph. D. thesis, Imperial College, London, UK (2015).

[57] D. Mirarchi *et al.*, "A crystal routine for collimation studies in circular proton accelerators", Proceedings of the 6^{th} International Conference Channeling: "Charged & Neutral Particles Channeling Phenomena", Capri, Italy (2014).

[58] D. Mirarchi *et al.*, "Crystal implementation in SixTrack for proton beams", ICFA Mini-Workshop on Tracking for Collimation in Particle Accelerators, CERN, Geneva, Switzerland (2015).

[59] F. Forcher, "An improved simulation routine for modelling coherent high-energy proton interactions with bent crystals", Bachelor thesis, UniPD, Padova, Italy (2017).

[60] R. Rossi, "Experimental Assessment of Crystal Collimation at the Large Hadron Collider", Ph. D. thesis, Università La Sapienza', Roma, Italy (2018).

List of Tables