European Organization for Nuclear Research
CERN BE/ABP

CERN/xx/xx
Updated May 2019

SixDesk

Version 2.1.0

the Simulation Environment for SixTrack

User's Reference Manual

- sixtrack -

R. De Maria, M. Giovannozzi, E. McIntosh, A. Mereghetti, F. Schmidt, I. Zacharov

Updated by: P. D. Hermes, D. Pellegrini, S. Kostoglou

Abstract

SixTrack [1, 2, 3] is a single particle tracking code widely used at CERN. One of its most important applications is the estimation of the dynamic aperture available in large storage rings like the Large Hadron Collider (LHC) or the Future Circular Collider (FCC). These studies require massive computing resources, since they consist of scans over large parameter spaces probing non-linear beam dynamics over long times. The SixDesk [4, 5] environment is the simulation framework used to manage and control the large amount of information necessary for and produced by the studies.
This document updates the previous documentation, and summarises how massive tracking campaigns can be performed with SixTrack starting from a MadX "mask" file. The SixDesk environment is an ensemble of shell scripts and configuration files, aimed at easing the everyday life of the user interested in performing large parameter scans with SixTrack.

Geneva, Switzerland
May 20, 2019

Acknowledgement

Some acknowledgements.

Contents

Chapter 1

Introduction

SixTrack [1, 2, 3] is a tracking code for simulating transverse and longitudinal single particle beam dynamics. Tracking is treated in a full six–dimensional way, including synchrotron motion, in a symplectic manner. SixTrack is widely used at CERN for predicting dynamic aperture in large storage rings [6] like the Large Hadron Collider (LHC) [7] or its upgrade as foreseen by the High Luminosity LHC Project (HL-LHC) [8, 9].

The code was extended [10] to predict the performance of a collimation system in terms of loss pattern and cleaning inefficiency. Hence, SixTrack is routinely used nowadays also for addressing the performance of existing cleaning systems, like those of the LHC [11] or of the Relativistic Heavy Ion Collider (RHIC) at BNL [12], or new ones.

The code is in continuous development [13, 14], not only to improve the accuracy of the tracking models, but also including the dynamics introduced by novel accelerator technologies, like electron lenses or powered wires for the compensation of beam–beam long range effects or christal collimation.

The accelerator dynamic aperture is studied scanning the beam phase space in presence of non-linear forces, like the kicks introduced by long range beam–beam interactions or multipolar components of magnetic fields. Moreover, the scan could be also performed varying the machine configurations. The SixDesk [4, 5] environment gives the users of SixTrack a mean to handle the large amount of files to be treated.

1.1 Assumed Environment

Table 1.1: Environment Variables.

Variable Name	Value
`appNameDef`	`sixtrack`
`newBuildPathDef`	`/afs/cern.ch/project/sixtrack/build`
`SixDeskTools`	`/afs/cern.ch/project/sixtrack/SixDesk_utilities/dev`

Throughout the manual, the environment in `lxplus.cern.ch`, native to SixDesk, will be assumed and the environment variables listed in Tab. 1.1 will be considered; these are automatically set by SixDesk. Since SixDesk is native to , the user has nothing to set up their environment. **TODO**: What to do in case of local installations or installations on other clusters?

1.2 Overview

TODO: Logics behind DA scans:

1. prepare the *input files*, i.e. `sixdeskenv`, `sysenv` and `fort.3.local`

Table 1.2: Pre-Requisites

Component	reason
kerberos	to renew/check credentials via `klist`, `kinit` and `aklog`
AFS (local mount)	retrieval of optics files submission to BOINC via spooldir
HTCondor (local installation)	submission of jobs to local batch system
`python2.7`	`SixDB` computation of floating point scan parameters

Table 1.3: Essential technical characteristics of the scans native to SixDesk.

Category	Variable	Comment
beam phase space	amplitude	main loop in SixDesk, sub-loop in SixTrack
	angle	loop in SixDesk, set point in SixTrack
machine phase space	magnetic errors (seed)	loop in SixDesk, a MadX job each
	tune	loop in SixDesk, each SixTrack job matches the tune

2. generate files describing the *accelerator geometry* with MadX (`fort.2`, `fort.8`, `fort.16`); then, run SixTrack; then, collect results (`fort.10`) and analyse them via SixDB;

3. inner loops (i.e. controlled by `sixdeskenv`) and outer loops (i.e. controlled by `scan_definitions`);

1.3 Work Flow

Show workflow of production of results, both for BOINC (including "processed" folder) and HTCondor.

Retrieval of results depends on the submission platform:

- `run_results`: BOINC
- `run_status`: HTCondor, HTBoinc

1.4 Scans

The scans performed by SixDesk (so-called "native") allow to estimate the dynamic aperture for a given machine configuration, mainly probing the beam phase space via a linear scan in particle amplitude parametric in angle. These scans cover different error configurations of the magnetic fields, and optionally, the user can also request to replicate the study varying the machine tune. Scans are handled by SixDesk with the input coded in the `sixdeskenv` file. Table 1.3 summarises essential technical characteristics of the SixDesk "native" scans.

A SixDesk study is exactly made of a complete "native" scan, with all the SixTrack input files describing the machine (see Sec. 1.2) generated by a single `*.mask` file. The beam phase space is scanned based on the settings in `sixdeskenv` file, and machine parameters like the multipolar errors and the tune are treated as "close" variations of the original study case.

"Non-native" scans are available as well, to extend the set of scans that can be performed (please see Sec. 2.6).

Table 1.4: Regular SixTrack input files for DA scans.

File	Description
fort.2	machine lattice and the nominal powering of magnets
fort.3	global simulation parameters and definition of special elements
fort.8	misalignments and tilt angles to be assigned to machine elements
fort.16	multipole errors to be assigned to magnetic elements

1.5 Input Files

In order to perform a Dynamic Aperture (DA) study, SixTrack must be provided with three geometry files [1], i.e. files that describe the geometry and settings of the machine lattice. A further input file, i.e. `fort.3` [1], is needed to set global simulation parameters of the SixTrack job and control special elements (e.g. beam-beam elements, wires, e-lenses, etc...). A summary table is found in Tab. 1.4.

Table 1.5: Input files for DA (inner) scans.

File	Description
sixdeskenv	main simulation parameters, scan ranges and some environment parameters
sysenv	additional environment variables
fort.3.local	additional parameters for each SixTrack job

Internal scans of DA (see Sec. 1.2) are controlled by means of two input files, i.e. `sideskenv` and `sysenv`. A third input file, `fort.3.local`, can be used to add simulation parameters to the `fort.3` file. This file is optional. A summary table is found in Tab. 1.5. Outer scans of DA (see Sec. 1.2) are controlled by further input files (see Sec. 2.6).

1.5.1 sixdeskenv

This file contains main simulation parameters, scan ranges and some environment parameters. They are summarised in Tab. 1.6.

Table 1.6: User-defined parameters of the `sixdeskenv` file.

Name	Description
additionalFilesOutMAD	list of MadX output files that should be fed into SixTrack jobs in addition to the geometry ones (see Sec. 1.5). Please leave blank if not needed; otherwise, please list all filenames (no paths) separate by whitespaces; as it happens for the geometry ones, one file per seed will be generated and gzipped. See also Sec. 2.4. e.g. `export additionalFilesOutMAD="fc.3.aper additional.txt"`
additionalFilesInp6T	list of additional SixTrack input files. Please leave blank if not needed; otherwise, please list all filenames (no paths) separate by whitespaces; every file will be used as it is by all SixTrack jobs. The files must be present either in the `sixjobs` directory or in the respective study subfolder of the `studies` directory. See also Sec. 2.4. e.g. `export additionalFilesInp6T="elens1.dat elens2.dat"`

1.5.2 sysenv

This file contains additional environment variables. They are summarised in Tab. 1.7.

Table 1.7: User-defined parameters of the `sysenv` file.

Name	Description
`appName`	name of the executable. If left blank, it defaults to `50205` (see Sec. 1.5.2.1). It is mandatory if the user wants to specify `appVer`. e.g. `export appName=sixtrack`
`appVer`	version of the executable. If left blank, it defaults to latest version (see Sec. 1.5.2.1). If not left blank, `appName` must be set either. In case of submissions to BOINC, it is advisable to set the variable. e.g. `export appVer=50205`
`SIXTRACKEXE`	full path to sixtrack executable. Used only by HTCondor and single turn jobs, ignored by BOINC (see Sec. 1.5.2.1). e.g. `export SIXTRACKEXE=$sixdeskpath/exes/$appName`

1.5.2.1 SixTrack Executable and its Full Path

In case of a job submitted to BOINC, a volunteer receives a copy of the executable together with the input files. In order to be trustable by volunteers, the executable must be signed. Hence, no custom-made executable can be sent to volunteers, and only signed executables (prepared on purpose by the admins) are made available to users. Therefore, in case of running jobs on BOINC, it is important that the user specifies which application to use and which version, via the `appName` and `appVer` variable. If left blank, SixDesk will set automatically the two in order to have the latest version of sixtrack. In case of BOINC, it is advisabel to set this variable.

The `appName` and `appVer` variables can be used also for single turn jobs (run on the login node on lxplus) and for jobs run on HTCondor; in this way, no matter the platform, the user deals with the same interface. Nevertheless, the user is given the possibility to specify the full path to the requested SixTrack executable, but, as already mentioned, the path will be used only for single turn jobs and HTCondor jobs and *ignored* in case of BOINC.

Therefore, it is recommended to *not define at all* the variable `SIXTRACKEXE` and to *leave blank* the variables `appName` and `appVer` and let SixDesk define them. In this case, SixDesk will set

```
SIXTRACKEXE = ${newBuildPathDef}/${appNameDef}
```

If the user defines `appName`, then SixDesk will set

```
SIXTRACKEXE = ${newBuildPathDef}/${appName}
```

If the user also defines `appVer`, then SixDesk will set

```
SIXTRACKEXE = ${newBuildPathDef}/${appVer}/${appName}
```

The existence of `$SIXTRACKEXE` is anyway checked. For `newBuildPathDef` and `appNameDef`, please see Sec. 1.1.

1.6 The BOINC Platform for Volunteering Computing

BOINC vs local batch system (e.g. HTCondor)

Chapter 2

New Features

This chapter illustrates the new features implemented in SixDesk from the user point of view. In general, all the new features have an introduction, where the rationale and the working principle of the new feature are briefly presented; afterwards, an essential look at user input and implementation is given; each section is then closed by a step by step guide, with examples. In the following, the environment variable `SixDeskTools` is assumed and defined as in Sec. 1.1.

2.1 Initialisation of Workspace and Study

Original work by: A. Mereghetti

It is useful to have a standard way of setting up workspace and study from within SixDesk, so that the user does not have to worry about proper template files and their synchronisation with the version of the scripts they uses.

2.1.1 Step-by-Step Guide

The main steps to properly set up a workspace and a study are:

1. set up the workspace, e.g.

```
> $SixDeskTools/utilities/bash/set_env.sh -N scratch2/wMySpace
```

   This action will set up the workspace, taking care of generating the correct hierarchy between the `sixjobs` and the `scratch*` directories. The action will create also the following tree structure:

```
> cd wMySpace/sixjobs
> tree -h
.
|__ [4.0K] control_files
| |__ [1013] fort.3.mother1_col
| |__ [ 942] fort.3.mother1_inj
| |__ [2.0K] fort.3.mother2_col
| |__ [2.0K] fort.3.mother2_col_b2
| |__ [2.0K] fort.3.mother2_inj
| |__ [2.0K] fort.3.mother2_inj_b2
|__ [ 475] fort.3.local
|__ [4.0K] mask
| |__ [ 39K] hl10BaseB1.mask
| |__ [ 35K] hl13B1.mask
|__ [ 996] scan_definitions
|__ [8.2K] sixdeskenv
|__ [ 115] sixdesklock
|__ [4.0K] sixdeskTaskIds
```

```
|__ [4.0K] studies
| |__ [ 0] sixdesklock
|__ [4.3K] sysenv

4 directories, 14 files
```

As it can be noted, this action takes care also of making available to the user *all* template input files (see following item for details);

2. (optional) go into the `sixjobs` dir and download templates, e.g.

```
> cd wMySpace/sixjobs
> $SixDeskTools/utilities/bash/set_env.sh -n -l -c
```

This action will make available to the user the template input files, i.e. the `sixdeskenv`, `sysenv`, `fort.3.local` (see Sec. 2.2) and `scan_definitions` (see Sec. 2.6) files. This action will also update the `workspace`, `basedir` and `scratchdir` variables in the `sixdeskenv` file with the correct values for the workspace just set up. Please be aware that this operation will overwrite any pre-existing file in the `sixjobs` dir. The templates will be downloaded from

```
${SixDeskTools}/utilities/templates/input
```

in this way, templates and scripts are synchronised. The `-l` option triggers the download of the `fort.3.local` (see Sec. 2.2) file, whereas the `-c` option triggers the download of the `scan_definitions` (see Sec. 2.6) file. This action is optional, as it is already performed by the `-N` action; nevertheless, it can be performed on its own and its usage has been shown.

When using either of the `-n` or `-N` actions, if the user requests the `-g` option, then all the files and directories will be downloaded with `git`. Hence, the user can profit from the diffing tools available with `git`; on the other hand, the disk usage grows (currently ~50 MB globally after either actions).

2.2 fort.3.local

Original work by: A. Mereghetti

2.3 Enforcing the Crossing Angle

Original work by: D. Pellegrini
Updated by: A. Mereghetti

2.4 Additional Files

Original work by: A. Mereghetti

In addition to the usual input SixTrack files, the user has the possibility to specify additional input files for the SixTrack jobs (see Sec. 1.5.1). Two classes of input files are available:

- MadX output files, to be re-used as input to SixTrack. These files are generated by each MadX job, and hence will be stored in the `$sixtrack_input` folder, renamed adding the seed and gzipped, as it happens for the regular geometry files (see Sec. 1.2);
- user-supplied files. These files will be the same for all simulations (i.e. they are not automatically updated based on the seed), and must be present in the `sixjobs` directory (when the study is created) or in the `studies/<myStudy>` subfolder (once the study has been created).

For each set of files, the user has to define a variable where the file names are listed; no paths can be taken into account or specified. The user can specify as many file names as desired, but must be listed separated by whitespaces.

Examples:

```
export additionalFilesOutMAD="fc.3.aper additional.txt"
export additionalFilesInp6T="elens1.dat elens2.dat"
```

As it can be seen, the former array of files store additional information calculated by MadX (e.g. the machine aperture); the latter is suitable for constant input data (e.g. a measured profile of radial electron current in electron lenses).

2.5 Variable Number of Angles with Amplitude

Original work by: D. Pellegrini
Updated by: S. Kostoglou, A. Mereghetti

Angular steps are forced to be consecutive, i.e. `kstep` is ignored and `1` is hardcoded in the logics.

2.6 Non-Native Scans

Original work by: P. D. Hermes, D. Pellegrini
Updated by: A. Mereghetti

SixDesk can perform "non-native" scans, i.e. scans aimed not at exploring further the beam phase space but machine configurations of possible interest – something that could be loosely called machine "phase space". Any point in a "non-native" scan is an independent SixDesk study, and it can be handled with the standard tools, since it has its own folders and files. On the other hand, all the studies have something in common; hence, it can be suitable to have a set of tools for treating all the studies in a "non-native" scan the same way.

"Non-native" scans can be useful to explore the dependence of the dynamic aperture on parameters like chromaticity, octupole current, and crossing angles, for the same collision optics. Therefore, these scans are based on a 1:1 relation between MadX and SixTrack, i.e. the knobs defined in MadX are exported as they are in SixTrack by means of the geometry files (see Sec. 1.2). Hence, the user is responsible for assuring that the desired parameters can be represented by MadX and all the necessary settings are propagated to SixTrack via the geometry input files, including magnet kicks as computed by the MadX matching. It should be noted that no parameter defining the "native" scan coded in the `sixdeskenv` input file is modified.

Two types of "non-native" scans are available to the user:

1. a scan over a *Cartesian grid* of an arbitrary number of variables with given steps for each variable. All the studies will be created and named after a reference machine configuration; each study will inherit a unique set of values of the scanned variables, which will appear explicitly in the study name together with the values actually used;

2. a scan over a *preset list of studies* which must exist. This option is extremely useful when punctual operations are required on a sub-set of studies composing the original scan.

2.6.1 Input Files

The file describing the external scan is the `scan_definitions`. It is a new file to SixDesk, where the user fully describes the Cartesian grid of interest or the pre-set list of studies. As for the `sixdeskenv` and `sysenv` files, it must be coded following the syntax of `bash`. Table 2.1 lists the variables that the file should contain. With the `scan_masks` logical variable, the user instructs SixDesk about the type of external scan to be performed:

Table 2.1: Parameters controlling external scans, to be defined by the user in the scan_definitions file. The central block of variables is used for scans on a *Cartesian grid*, whereas the last block is used for scans on a *preset list* of studies.

Parameter Name	Comment	Example
scan_masks	trigger to use preset list of studies	scan_masks=false
scan_variables	variable names (used in study name)	scan_variables="B QP"
scan_vals_<vNam>	values to be explored for variable <vNam>	scan_vals_B="1 4"
		scan_vals_QP="0 2 4"
scan_placeholders	placeholders in *.mask file	scan_placeholders="%BV %QPV"
scan_prefix	common part of study name	scan_prefix="HLLHC_inj"
scan_studies	explicit list of studies in the scan	scan_studies="HLLHC_inj_B_1_QP_4 HLLHC_inj_B_4_QP_0"

scan_masks=false the scan is performed on the *Cartesian grid*; in this type of scan, the central block of variables shown in Tab. 2.1 are used;

scan_masks=true the scan is performed on the *pre-set list* of studies; in this type of scan, the last block of variables shown in Tab. 2.1 are used.

It should be kept in mind that, in the case of the *Cartesian grid*, the user must set up a *.mask file, to be used as template for the studies in the scan. All the other regular input files (see Sec. 1.2) determine the internal scan performed in each study, and are essentially cloned, so that the dynamic aperture is probed in the same way in all points of the external scan. On the contrary, in the case of the preset list of studies, all the concerned studies must be already existing, and no other input file is required.

2.6.1.1 Scan on a Cartesian Grid

In the scan on a *Cartesian grid*, all the concerned studies are generated out of a set of template files, based on a sixdeskenv, sysenv, *.mask and scan_definitions files (and fort.3.local, optionally). All the optics configurations are variations of the same one coded in the template *.mask file.

The user defines the parameter space in the scan_definitions file at their will, with no restrictions due to interfaces. The user must make sure that the desired parameters can be represented by MadX and all the necessary settings are propagated to SixTrack via the geometry input files (see Sec. 1.2). In fact, contrary to what done normally in SixDesk, the user defines suitable *placeholders* that will be used by SixDesk for query/replace in the *.mask file and for disentangling the various studies. Hence, it is responsibility of the user not only to define the variables and the concerned range of values, but also to set up the necessary *placeholders* in the template *.mask file.

For starting an external scan, the user should prepare:

- a regular sixdeskenv, to be used as template. The file is automatically replicated by SixDesk in all the studies involved in the scan as is, with the exception of the actual study name (i.e. the LHCDescrip field), which is automatically updated at the generation of the study. Hence, it is user's convenience to freeze the paramters for the internal scan before starting the external one, such that all the studies will inherit immediately the correct parameters and range of values;

- a regular sysenv, to be cloned as is, with no further modifications by SixDesk. As for the sixdeskenv file, it is user's convenience to set this file up correctly and completely before starting the external scan;

Table 2.2: Input files for external scans.

File	Comments	Location
`sixdeskenv`	– a template file for automatic query/replace – it must define correct settings for the internal scan	`sixjobs`
`sysenv`	cloned as it is	`sixjobs`
`*.mask`	– a template for automatic query/replace – it must contain place holders of scanned parameters	`mask`
`scan_definitions`	unique it describes the scans (`bash` syntax)	`sixjobs`

- an optional file `fort.3.local`, to be cloned as is, with no further modifications by SixDesk. As for the `sixdeskenv` and `sysenv` files, it is user's convenience to set this file up correctly and completely before starting the external scan;
- a template `*.mask` file, to be used to generate all studies in the scan. SixDesk will take care of cloning it to the involved studies, automatically performing the query/replace of the placeholders necessary to correctly set up the study. The query/replace patterns (and hence the placeholders) are uniquely defined by the user, and no spefic syntax is hard-coded in SixDesk;
- the `scan_definitions` files, which contains the full description of the scans. More than a parameter can be scanned at the same time, and the actual studies handled will follow the cartesian product of all the parameter values.

Table 2.2 summarises the key facts about the input files.

The user requests SixDesk to perform a scan on the *Cartesian grid* setting the `scan_masks` flag in the `scan_definitions` file to `false`. The same file (see Tab. 2.1) contains all the information necessary to define the scan:

- the variable names to be looped on are specified by the user via the `scan_variables` variable;
- the respective placeholders in the `*.mask` file are specified via the `scan_placeholders`;
- the range of values to be scanned are specified via variables like `scan_vals_<vNam>`, one per scanned parameter `<vNam>`.

When generating the `*.mask` specific to each study, SixDesk will automatically copy the template `*.mask` file and query/replace the placeholders with the actual values to be used. Hence, the parameter names must match actual *placeholders* in the template `*.mask` file, and it is the responsibility of the user to match the *placeholders* listed in the `scan_definitions` with those in the template `*.mask` file.

The naming convention of the study (and hence of the `*.mask` file) combines a commond name (which can identify e.g. the specific optics explored in the scan) and the name of each scanned variable with the explicit value used in each study.

Table 2.1 reports an example of variables in the `scan_definitions`, coding an external scan for studying the dynamic aperture of the HL-LHC machine at injection; the scan is performed on both beams (variable `B`, `%BV` as placeholder in `*.mask`, and values 1 and 4) with three values of chromaticity (0, 2 and 4, variable `QP` and `%QPV` as placeholder in). As it can be seen, names of variables and placeholders are fully decided by the user, with no rules enforced by SixDesk. Anyway, at set-up time, SixDesk will check that placeholders exist in the template `*.mask` file.

The template `*.mask` file must be existing in the `mask` directory, and it must have the name specified in the `scan_prefix` field in the `scan_definitions` file. In the example, the template `*.mask` file would be named `HLLHC_inj.mask`. The actual scan is made of 6 studies, named:

```
HLLHC_inj_B_1_QP_0
HLLHC_inj_B_1_QP_2
HLLHC_inj_B_1_QP_4
HLLHC_inj_B_4_QP_0
HLLHC_inj_B_4_QP_2
HLLHC_inj_B_4_QP_4
```

2.6.1.2 Scan on a Preset List of Studies

If the user has already produced the required `*.mask` files and want to scan over a specific (sub)set of studies, they can specify the study names explicitly. This can be useful if they want to run a command for only a subset of a larger set of studies of the Cartesian scan. To use this option, the variables used to set up the *Cartesian product*, listed in the middle block of Tab. 2.1 are not suitable, and those described in last block of the same table should be used.

The user requests SixDesk to perform a scan on the *preset list* of studies setting the `scan_masks` flag in the `scan_definitions` file to `true`. The same file (see Tab. 2.1) specifies also the list of the studies to be treated via their full name. As already mentioned, the concerned studies with all their input files and folders must already exist.

In the above example, the only studies which will be considered in the scan are (once the `scan_masks` flag is set to `true` by the user):

```
HLLHC_inj_B_1_QP_4
HLLHC_inj_B_4_QP_0
```

2.6.2 Implementation

The scans are handled via the `scans.sh` user script; it is simply a bash wrapper which loops the action requested by the user over the desired studies. The actual functions are coded in `dot_scan` (`bash`) library. Hence, the user will have to deal with only the `scans.sh` script.

To perform a desired action on all the studies in the scan, the user just need to issue the `scans.sh` script using the `-x` *action* with the detailed command to be performed enclosed within double quotes. There is no need to specify the `-d` *option* for the called script, since `scans.sh` will automatically trigger the requeted command on each study in the scan separately. The script will take care of looping over all the studies and issue the requested command on each study. The only exceptional actions that have dedicated terminal line arguments are the generation of the actual `*.mask` files, achieved via the `-m` *action*, and the set up of the directories of each study, achieved via the `-s` *action*.

When generating the `*.mask` files, the script checks beforehand that all the placeholders that the user is going to use are found in the `*.mask` template file. To disable this option, please use the `-m` *option*.

The use of the `fort.3.local` file can be triggered via the `-l` *option*, with no need to replicate it also in the string passed through the `-x` *action*.

A very basic parallelisation of the scan is available. The user can split the final scan into smaller ones. Each of them must have its own `scan_definitions` files, with a unique name. Then, the respective number of instances of the `scans.sh` can be issued, each with the `-d` *option* with the specific name of the `scan_definitions` instance to be used.

2.6.3 Step-by-Step Guide

This guide is given for an external scan on a *Cartesian grid* started from scratch:

1. set up your workspace and download template files (see Sec. 2.1);

2. edit all the necessary files, e.g.

(a) sixdeskenv and sysenv, properly setting up the internal scans, versions of codes, etc... Please, make sure that the xing variable in sixdeskenv is not active (see Sec. 2.3);

(b) template *.mask file in the mask directory, and scan_definitions. Please make sure that:
- scan_prefix matches the name of the template *.mask file;
- the lists contained in scan_variables and scan_placeholders match;
- for every variable scanned (e.g. QP), you have the corresponding list of values defined in the scan_vals_* (e.g. scan_vals_QP);
- all the placeholders defined in scan_placeholders are actually in the *.mask template file, and in the correct positions. Please keep in mind that the query/replace will be performed via a sed command;

3. generate all the necessary *.mask file and the studies, e.g.

```
> $SixDeskTools/utilities/bash/scans.sh -m -s -l
```

The -l *option* is illustrated in the example to show the command in case the fort.3.local file is required. The -m *action* (i.e. generation of *.mask files) and the -s *action* (i.e. set up of studies) can also be performed separately;

4. run MadX and generate the geometry files for the SixTrack jobs, e.g.

```
> $SixDeskTools/utilities/bash/scans.sh -x "mad6t.sh -s"
```

Once the jobs are over, it is good practice to check them before running SixTrack, to avoid mis-submissions in case something went wrong with the MadX jobs. Checking can be performed e.g.

```
> $SixDeskTools/utilities/bash/scans.sh -x "mad6t.sh -c"
```

5. submit the actual SixTrack jobs, e.g.

```
> $SixDeskTools/utilities/bash/scans.sh -x "run_six.sh -a -p BOINC"
```

Submission is explicitely done to the BOINC platform for all the studies. The usual list of platforms supported by run_six.sh is available;

6. download results and update the job database

```
> $SixDeskTools/utilities/bash/scans.sh -x "run_results"
```

The same command can be issued with run_status;

7. scans.sh can be used for calling any script in SixDesk, e.g.

```
> $SixDeskTools/utilities/bash/scans.sh -x "correct_cases"
```

Chapter 3

Giudelines and Common Pitfalls

3.1 Naming Convention

3.1.1 Study and Workspace Names

- avoid the use of "__" (i.e. a string of two consecutive underscore characters), as this is used by the BOINC assimilator to properly disentangle study name and name of job in the study;
- avoid the use of the platform in lower case explicitly in the name;

3.2 Choice of Platform

HTCondor is convenient when:

1. results should be collected quickly. This can be the case when the user has short time to collect data or the simulation set-up is being defined. In the second case, indeed, one does not want to wait too long for proceeding;
2. short or few jobs per study. This can be the case when re-submission of selected cases is necessary, e.g. to complete a study when few points in the scan are missing;

The BOINC platform for volunteer computing is convenient in case of large simulation campaigns, i.e. when simulations are long or they are in high number (e.g. hundreds of thousands of jobs).

Not more than 5 scripts per user running at the same time, for ease of functionality of afs.

Bibliography

[1] F. Schmidt, “SixTrack: Version 4.7.16, Single Particle Tracking Code Treating Transverse Motion with Synchrotron Oscillations in a Symplectic Manner, User’s Reference Manua”, CERN/SL/94–56 (AP), CERN, Geneva, Switzerland (2017), http://sixtrack.web.cern.ch/SixTrack/docs/user_full/manual.php

[2] G. Ripken and F. Schmidt, “A symplectic six-dimensional thin-lens formalism for tracking”, CERN, Geneva, Switzerland, Rep. CERN/SL/95–12(AP), 1995.

[3] SixTrack, http://sixtrack.web.cern.ch/SixTrack

[4] M. Hayes and F. Schmidt, “Run Environment for SixTrack”, LHC Project Note 300, CERN, Geneva, Switzerland (2002), https://cds.cern.ch/record/691785/files/project-note-300.pdf

[5] E. McIntosh and R. De Maria, “The SixDesk Run Environment for SixTrack”, CERN-ATS-TE-2012-089 TECH, CERN, Geneva, Switzerland (2002), https://github.com/SixTrack/SixDesk/blob/master/sixjobs/doc/sixdesk_env.pdf

[6] M. Giovannozzi *et al.*, “Dynamic Aperture Studies for the LHC Hight Luminosity Lattice”, in *Proc. 6th Int. Particle Accelerator Conf. (IPAC’15)*, Richmond, VA, USA, May 2015, paper MOPMN003, pp. 705–709.

[7] O. Brüning *et al.* (eds), “LHC design report”, Vol. I, CERN, Geneva, Switzerland, Rep. CERN-2004-003-V-1, 2004.

[8] O. Brüning, L. Rossi (eds.), “The High Luminosity Large Hadron Collider”, World Scientific Press, 2015, ISBN 978-981-4675-46-8.

[9] G. Apollinari, I. Bejar Alonso, O. Brüning, M. Lamont, L. Rossi (eds.), “High Luminosity Large Hadron Collider (HL-LHC) Technical Design Report V.01”, CERN, Geneva, Switzerland, EDMS n. 1723851 v.0.71, https://edms.cern.ch/ui/file/1723851/0.71/HL_TDR_V07.0.2016.10.05.Version15.2h.pdf

[10] G. Robert-Demolaize, R. Assmann, S. Redaelli, F. Schmidt, “A New Version of SixTrack with Collimation and Aperture Interface”, in *Proc. Particle Accelerator Conf. (PAC’05)*, Knoxville, TN, USA, 2005, paper FPAT081, pp. 4084–4087.

[11] R.W. Assmann *et al.*, “The Final Collimation System for the LHC”, in *Proc. 10th European Particle Accelerator Conf. (EPAC’06)*, in *Proc. EPAC’06*, Edinburgh, Scotland, UK, Jun 2006, paper TUODFI01, pp. 986–988.

[12] G. Robert-Demolaize and A. Drees, “Simulations of collimation losses at RHIC”, in *Proc. Tracking for Collimation Workshop*, CERN, Geneva, Switzerland, Oct 2015, unpublished.

[13] S. Redaelli (ed.), “Proceedings of the tracking for collimation workshop”, CERN, Geneva, Switzerland, Oct 2015, unpublished.

[14] A. Mereghetti *et al.*, "SixTrack for Cleaning Studies: 2017 Updates", in *Proc. 8th Int. Particle Accelerator Conf. (IPAC'17)*, Copenhagen, DK, May 2017, paper THPAB046, pp. 3811–3814.

[15] "Berkeley Open Infrastructure for Network Computing", http://boinc.berkeley.edu

List of Tables