European Organization for Nuclear Research
CERN BE/ABP

CERN/xx/xx
Updated April 2018

SixDesk

Version 1.0

the Simulation Environment for SixTrack

User's Reference Manual

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 describes 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
April 25, 2018

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 Overview

1. SixTrack input generated by MadX (`fort.2`,`fort.8`, `fort.16`); then, run SixTrack; then, collect results (`fort.10`) and analyse them via SixDB;

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

1.2 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

1.3 The BOINC Platform for Volunteering Computing

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

1.4 Pre-requisites

SixDesk is native to `lxplus.cern.ch`. Hence, for running in such an environment, the user does not need to set up anything. On the contrary, in case of a local machine or other distributed resources,

Table 1.1: Pre-Requisites

Component	**reason**
kerberos	to renew/check credentials via `klist` and `kinit`
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

Chapter 2

New Features

This chapter illustrates the new features implemented in SixDesk from the user point of view.

2.1 External Scans

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

"Internal scans" are the scans handled by SixDesk coded in the `sixdeskenv` file. These are the fundamental scans used 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. The internal scan can also cover different error configurations of the magnetic fields; optionally, the user can also request to replicate the study varying the machine tune. Table 2.1 summarises essential technical characteristics of the internal scans.

Table 2.1: Essential technical characteristics of the internal scans.

Category	Variable	Comment
beam phase	amplitude	loop both in SixTrack and SixDesk
space	angle	loop in SixDesk, set point in SixTrack
machine	magnetic errors (seed)	loop in SixDesk, involving also MadX
phase space	tune	loop in SixDesk, involving re-matching in SixTrack

The internal scans make actually one study, as all the SixTrack input files describing the machine (i.e. `fort.2`, `fort.8` and `fort.16`) are generated by a single `.mask` file. The beam phase space is scanned, and machine parameters like the multipolar errors and the tune are treated as "close" variations of the original study case.

"External" scans identify a set of additional scan parameters, not aimed at exploring further the beam phase space but machine configurations of possible interest – hence the machine "phase space" is explored. Any point in an external scan is an independent SixDesk study, and it can be handled with the standard tools, since it has its own folder trees. But since all the studies are variations of the same machine configuration, it is logical to keep them boundled together in the same workspace, with a convenient tool which loops through them.

External scans be useful to explore the dependence of the dynamic aperture on parameters like chromaticity, octupole current, and crossing angles, for the same linear optics. Therefore, these scans are based on a 1:1 relation between MadX SixTrack, i.e. the knobs defined in MadX are exported as they are in SixTrack by means of the geometry files (i.e. `fort.2`, `fort.8` and `fort.16`); this includes magnet kicks as computed by the MadX matching.

It should be noted that the present machinery set up for external scans does not modify any other parameter in the `sixdeskenv` and `sysenv` input files.

The user can define two types of external scans:

1. a scan over a *Cartesian grid* of an arbitrary number of variables with given steps for each variable;
2. a scan over a *preset list* of studies which must exist beforehand, including the `sysenv`, `sixdeskenv` and `.mask` files.

In the case of the Cartesian grid, all the studies will be named after a reference machine configuration, and the names of the scanned variables will appear explicitly in the study name, together with the values actually used for a given study.

2.1.1 Input Files

The essential information is contained in the `.mask` file, used as template for the studies in the scan, and in the `scan_definitions` file, describing the parameters and range of values of the scan. These two files determine the set of studies building up the external scan, whereas all the other regular input files (i.e. `sixdeskenv`, `sysenv` and `fort.3.local`) determine the internal scan performed in each study, and are essentially cloned, so that all points in the external scans are probed the same way.

The user defines the parameter space at their will, with no restriction 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 (i.e. `fort.2`, `fort.8` and `fort.16`). In particular, it is responsibility of the user not only to define the variables and range of values, but also to prepare in the template `.mask` file suitable *placeholders* that will be recognised by SixDesk and used for differentiating the various studies. Hence, contrary to what done normally in SixDesk, the user is responsible for the proper definition of the parameter space, i.e. not only that the range of explored values is sensible, but also all the handles in the `.mask` file generate adequate input to SixTrack jobs.

More in details:

`sixdeskenv` a regular `sixdeskenv` is 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. `LHCDescrip` field), which is automatically generated as well. 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 ones;

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

`fort.3.local` the optional file `fort.3.local` can be used in the external scan with no specific limitations; it will be cloned as it is by SixDesk to all bundled studies. As for the `sixdeskenv` and `sysenv` files, it is user's convenience to set this file up before starting the external scan;

`.mask` a `.mask` file is used as template to all studies in the scan. SixDesk will take care of cloning it to the involved studies, automatically performing the query-replace necessary to correctly set up the study. The query-replace patterns are uniquely defined by the user, and no spefic syntax is hard-coded in SixDesk;

`scan_definitions` this is a new file to SixDesk. It contains the full description of the scans, using a `bash` syntax. More than a parameter can be scanned at the same time; if it is the case, then the actual studies submitted will follow the cartesian product of all the parameter values.

Table 2.2: Input files for external scans.

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

Table 2.2 summarises the key facts about the input files.

The user requests SixDesk to perform a scan on the Cartesian grid or on the preset list of studies via the `scan_masks` flag in the `scan_definitions` file (see later); it must be set to `false` in case the Cartesian grid is the desired method of scanning, of to `true` for the preset list of studies.

2.1.1.1 Scan on a Cartesian Grid

For the Cartesian grid, the variable names to be looped on in the `.mask` files are specified by the user, together with the the esplored range of values. The variables are automatically replaced by SixDesk and the new `.mask` files created. The user defines the parameters of the Cartesian grid and the range of values that should be covered by each. In each new study, the same template `.mask` file is copied and modified according to the requirements of the user. The parameter names must match actual *placeholders* in the `.mask` file. The variable names to use in the `.mask` file and the values to be spanned are set in the `scan_definitions` file.

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

Table 2.3: Parameters controlling external scans on a Cartesian grid, to be defined by the user in the `scan_definitions` file.

Parameter Name	Comment	Example
`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_masks`	trigger to use preset list of studies	`scan_masks=false`

The user defines in the `scan_definitions` files the variable names, the placeholders used in the `.mask` file and the actual values to be scanned. Table 2.3 summarises the necessary fields, with examples.

The examples report the parameters scans for studying the dynamic aperture of the HL-LHC machine at injection; the scans are done on both beams (1 and 4, variable `B` and `%BV` as placeholder in `.mask`) with three values of chromaticity (0, 2 and 4, variable `QP` and `%QPV` as placeholder in `.mask`). 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 above 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
```

2.1.1.2 Scan on a Preset List

If the user has already produced the required `.mask` files and wants to scan over a specific (sub)set of studies, he can also specify the study names explicitly. This can be useful if we want to run a command for only a subset of a larger set of studies of the Cartesian scan. To use this option, the variable listed in Tab. 2.3 are not suitable, and those described in Tab. **??** should be used. In the

Table 2.4: Parameters controlling external scans for a preset list of studie, to be defined by the user in the `scan_definitions` file.

Parameter Name	Comment	Example
`scan_masks`	trigger to use preset list of studies	`scan_masks=true`
`scan_studies`	explicit list of studies in the scan	`scan_studies="HLLHC_inj_B_1_QP_4 HLLHC_inj_B_4_QP_0"`

above example, the only studies which will be considered in the scan are:

```
HLLHC_inj_B_1_QP_4
HLLHC_inj_B_4_QP_0
```

As already mentioned, all the necessary input files (e.g. `.mask`, `sixdeskenv`, `sysenv`, `fort.3.local`, etc...) and the concerned folders (e.g. `sixtrack_input`, `track`, `work`, `study`, etc...) must already exist.

2.1.2 Step-by-Step Guide

This little guide will show the user how to set-up an the external scan and how to proceed, step by step.

2.1.3 Implementation

Everything is contained in the `scans.sh` user script and the `dot_scan` `bash` library.

Chapter 3

Giudelines and Common Pitfalls

3.1 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).

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