EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
European Laboratory for Particle Physics

The SixDesk Run Environment for SixTrack

F. McIntosh, R. De Maria

Abstract

This document replaces the "Run Environment for SixTrack" [1], and describes how massive tracking campaigns can be performed with "SixTrack" [2] starting from a MAD-X input file of the LHC lattice, the so called mask file. It describes a new set of UNIX BASH or Korn shell scripts which allow the use of the Berkeley Open Infrastructure for Network Computing, BOINC [**?**]) as an alternative to the Linux LSF batch system.

Administrative Secretariat
LHC Division
CERN
CH-1211 Geneva 23
Switzerland

Geneva, September 5, 2012

Version 10 20th August, 2012.

1 Objectives

The principal objective of the SixDesk run environment was to allow a physicist to run a tracking campaign, on either the CERN LSF batch system or BOINC, using the familiar SixTrack run environment on Linux. At the same time, the opportunity was taken to group all user modifiable parameters into two files, `sixdeskenv` and `sysenv`, and to speed up `run_mad6t`, the madX to SixTrack conversion, by running in parallel in batch. The specification of tune scans, fractional amplitude ranges and steps, and various other physics options were also added to the `run_six` script/command. All SixDesk scripts report activity to the `sixdesk.log` file and exit with an error code if an unrecoverable error is detected. The messages are optionally sent to `STDOUT`, your screen in an interactive session.

2 Getting Started

All the user modifiable parameters have been collected into the two scripts `sixdeskenv`, `sysenv` obviating the need to look through all scripts and make matching changes, simplifying the usage of the scripts, and limiting the risk of error.

It is assumed that there is a large amount (at least 1GB) of disk space available on AFS to be used as a workspace which we shall call `w1` for illustration. The `w1` directory is normally a link in $HOME to a directory with the same name on an AFS scratch disk (scratch0, scratch1, etc). Note that by default access to all files will be limited; it is useful to do a `fs setacl w1 system:anyuser rl` so that everything in the workspace can be read by support.

A workspace is created by e.g.

```
cd $HOME
mkdir scratch0/w1
ln -s ~/scratch0/w1 w1
cd w1
fs setacl . system:anyuser rl
```

and the SixDesk environment is created by

```
cd w1
svn checkout https://svn.cern.ch/reps/sixdesk
```

or

```
cd w1
gtar xvzf /afs/cern.ch/user/f/frs/public_html/\
SixTrack_run_environment/SixDesk_run_environment.20August2012.tgz
```

where the first method will get the latest SVN committed version and the second tar file will be updated (and the name changed) in parallel with the SVN commits.

This will create a directory subtree starting with the directory `sixjobs`. All the scripts are stored here and the user should always do a `cd w1/sixjobs` before executing any commands. It may be necessary to prefix any typed commands with a `./` if the shell PATH does not include the current directory `.`. (The source version of each script can be found in `sixjobs/scripts` as will be discussed later.)

Other older and newer versions and updates can be found in the same public_html directory.

3 Setting up a Study

To facilitate the description of the various procedures we use this workspace `w1` and a study `job_tracking` as an illustration using the mask file `job_tracking.mask` included in the release. Note that the name of the study is contained in the variable `LHCDescrip` i.e. if we have a study called `lhc_nob1` then we must define it with `export LHCDescrip=lhc_nob1`. It should also be noted that amplitudes and amplitude ranges are specified in beam σ. There are several supported types of study, short studies using SixTrack and possibly Sussix, long tracking studies using SixTrack, and DA map production using the SixTrack DA version. Performing a tracking study involves running many jobs for an LHC configuration with many different initial phase space amplitudes and angles and linear tunes. The LHC configuration is defined by a mask file in the `sixjobs/mask` sub-directory and the initial conditions in the `sixdeskenv, sysenv` files.

The first step is to edit the `sixdeskenv` script and, if necessary, the `sysenv` script.

NOTA BENE:It is strongly recommended to issue a `set_env` command after any modification to these files so that they are saved in the `studies/study` directory as described later. Note also, that as shown for the variable `LHCDescrip`, all values must be exported.

export LHCDescrip=job_tracking the name of the study and of the mask file
basedir=/afs/cern.ch/user/$initial/$LOGNAME
scratchdir=/afs/cern.ch/user/$initial/$LOGNAME/scratch0
trackdir=$scratchdir/$workspace
sixtrack_input=$scratchdir/sixtrack_input/$workspace/$LHCDescrip but may also be changed to use an existing set of `sixtrack_input` files.

The above defaults, apart from the name of the study, are usually satisfactory but, once chosen, cannot be changed easily. Other parameters of the run can be decided later.

At this point:

```
cd ~/w1/sixjobs
set_env
```

will create the full directory structure of the environment. As already mentioned this command should also be used to save modified versions of the `sixdeskenv, sysenv` files before doing anything else. The command also reports and logs changes. (Remember to do a `set_env` as mentioned earlier after any modification of these two files.)

While it is recommended to run one study per workspace, most users have multiple studies in the workspace. The command `ls studies` can be used to list them and `set_env "name of the study"` to switch between studies. This switch copies the `sixdeskenv, sysenv` files from the `studies` directory to `sixdekhome` i.e. the current `sixjobs` directory.

An important new feature is that it is possible to execute many commands on a specific study without switching. This facilitates running multiple studies in the same workspace and using commands in batch jobs. Thus, instead of switching, a command can be suffixed with an optional ”study” and/or ”platform”. Examples are shown later. This option is NOT available for the backup or recall, study or workspace, commands described in section 19.

The `print_env` command reports the complete environment to the screen; the command `mywhich` reports a few important values.

Finally note that it is vital that the same versions of SixTrack and madX are used for all cases in a study. Ideally the madX `run_mad6t` jobs should be run on the same type of computer

to avoid small numeric differences. Compatible versions of these programs for LSF and BOINC are specified in `sysenv`, the so-called "pro" versions. Any changes will be notified.

4 Overview of the Data Structure

The `set_env` will have created the `sixtrack_input`, `studies/study`,`track` and `work` directories, as well as various logfile links and directories. The files `sixtrack_input, study, track, work` are in fact links to the actual directories. An empty file with the name of the study is also created to facilitate SHELL command completion.

The `sixtrack_input` directory will later (after a successful mad6t run) hold all the SixTrack input mother files `fort.3.mother1` and `fort.3.mother2` derived from the mother files in the `control_files` directory as well as a `fort.3.mad` and a `fort.3.aux` and one `fort.2_"seedno".gz`, `fort.8_"seedno".gz`, `fort.16_"seedno".gz` for each seed in the range `istamad` to `iendmad` as defined in `sixdeskenv`. In addition it will hold one `mad.dorun_mad6t*` directory for each `run_mad6t` command, with one mad6t LSF job `mad6t_"seedno".lsf`, one input file `"study"."seedno"`, one LSF job log `"study"_mad6t_"seedno".log` and one mad6t output file `"study".out."seedno"` for each seed.

The `studies` directory will hold one directory for each study, in turn containing the `sixdeskenv, sysenv` files for that study.

The `track` directory will become a hierarchy containing all the SixTrack data and results as follows:

Level 1 Seed typically 1 to 60
general_input containing normalised emittance and gamma
Level 2 simul for long tracking and/or
trans momen for short/sussix runs
Level 3 tunex_tuney e.g. 64.31_59.32
Level 4 amplitude range e.g. 18_20 and 18-22 after post-processing
Level 5 turns exponent e.g. e5 for 10^5 turns
Level 6 the phase space angle e.g. 67.5
Level 7 input(links), fort.2,3,8,16.gz and the result fort.10.gz as well as the LSF jobs and logs

A typical lowest level structure (after the study has been completed), for workspace w1, study job_tracking, long run (simul), seed 1, tunes 64.28_59.31, amplitude range 10_12, 10^5 turns, phase space angle 1.5, is:

```
~/w1/sixjobs/track/1/simul/64.28_59.31/10_12/e5/1.5
fort.10.gz
fort.16.gz ->
~mcintosh/scratch0/sixtrack_input/w1/job_tracking/fort.16_1.gz
fort.2.gz ->
~mcintosh/scratch0/sixtrack_input/w1/job_tracking/fort.2_1.gz
fort.3.gz
fort.8.gz ->
~mcintosh/scratch0/sixtrack_input/w1/job_tracking/fort.8_1.gz
job_tracking%1%s%64.28_59.31%10_12%5%1.5.log
job_tracking%1%s%64.28_59.31%10_12%5%1.5.lsf
```

Note that input files are in fact links to sixtrack_input, in order to save disk space, apart from the fort.3.gz which is of course different for each job. The result file is fort.10.gz described in

section 9. The .lsf and .log files are the LSF job file and the LSF job log file for the particular case.

It should be noted that there is a one to one mapping between the name of a case and the directory where the input and output files are stored. The general form of the name of a case in a `long` run is

```
study%seed%s%tune_range%amplitude_range%turns_exponent%angle
```

. In the workspace this case would be found in the directory

```
track/'seed'/simul/'tune_range'/'amplitude_range'/e'turns_exponent'/'
```

. To give a specific example, a case named

```
job_tracking%1%s%62.31_60.32%10_12%5%18
```

would be found in the `w1/sixjobs/track` directory for this study in the sub-directory

```
sixjobs/track/1/simul/62.31_60.32/10_12/e5/18
```

this particular case being for seed number 1, tunex and tuney 62.31 and 60.32, amplitude range 10 to 12, 10^5 turns, angle 18. (The directory simul could also be instead trans or momen for short/sussix runs and the letter 's' in the name being replaced by 't' or 'm'.)

This case name is used in the database, to name the LSF jobs and logs, and is also included in the SixTrack `fort.6` output. Note that as warned in a comment in the `sixdeskenv` file this name must NOT have a % character nor two consecutive underscores.

The `work` directory will contain the database flat files for reporting and managing all the tasks and jobs of a particular study. The most interesting files are `work/taskids`, `work/completed_case` `work/incomplete_cases` and the sub-directory `work/lsfjobs (or boincjobs)`. The master file is called `taskids` and contains one line for every case of a study with the case name followed by one or more LSF or BOINC taskids. The `run_status` command reports on the status of the study as described in section 8.

There are also the other directories:

bin containing links to various utility programs used by `run_join10` and `run_post`

control_files the SixTrack mother files for the collision/injection/beam 2 cases which are used to generate the SixTrack input files

inc other mask files and the prepare_fort.3 script

mask the LHC description mask files for madX

plot the plotting mask files and any plots produced by the post processing

sixdeskTaskIds the study TaskIds for BOINC

scripts the source and copies of the commands

utilities the various LSF job masks for `run_mad6t` and `run_six`

(Here we will add a picture of the hierarchy and perhaps a screen shot.)

5 Running mad6t to produce the basic SixTrack input files

NOTA BENE: It is essential to have a $LHCDescrip.mask file in the subdirectory `mask` in order to run madX to produce the SixTrack input files. This mask file in turn references many LHC Database files and this often requires some checking. Sample mask files can be found in the directory mask:

```
jobref503_withbb_coll.mask   jobrefslhc_withbb_coll.mask
jobref503_withbb_inj.mask    jobref503_inj.mask
jobrefslhc_inj.mask          jobrefslhc_withbb_inj.mask
```

```
jobref503_coll.mask          jobrefslhc_coll.mask
job_tracking.mask
```

The file job_tracking.mask is copied from /afs/cern.ch/eng/lhc/optics/SLHCV3.01 and all the others from /afs/cern.ch/eng/lhc/optics/SLHCV2.0 **??**.

In order to investigate a series of random seeds the particular seed number in the MAD input file has to be replaced by a variable name:

```
Set, SEEDSYS , 1 ;
Set, SEEDRAN , 1 ;
```

becomes

```
Set, SEEDSYS , %SEEDSYS ;
Set, SEEDRAN , %SEEDRAN ;
```

The place keepers `%SEEDSYS` and `%SEEDRAN` will be replaced automatically by a proper seed number by `run_mad6t` based on the current seed number based on the `istamad and iendmad` variables defined in `sixdeskenv`.

Please note that the MAD-X `sixtrack` command takes its information from the last `Twiss, save` commmand. It is a sensible precaution to put these commands consecutively in the MAD script and mask files.

The version of madX to be used for the conversion runs is defined in `sysenv` and defaults to the current production version. At this point it is also necessary to specify in `sixdeskenv`

pmass=938.272046 The mass of the proton [4] which can be reset to 938.271998 for backwards compatibility with earlier studies.

bunch_charge=1.1500e+11 New bunch_charge variable for fort.3.mother1_[col/inj]

runtype= inj or coll for injection/collision for an LHC lattice. For more information on how to define runtypes for either the LHC or other machines please see section **??**.

beam= null or b1 or B1 for Beam1, b2 or B2 for Beam2

CORR_TEST= 0 or 1 if check_mad6t is to copy the corrector strengths for each seed into one file in sixtrack_input

fort_34= If null the fort.34 files will not be copied to the sixtrack_inputdir. These files define the multipole strengths for the linear lattice when doing Second Order Detuning and Distortion (SODD) analysis in expert mode.

istamad=1 first seed for madX

iendmad= normally 60 (maximum 64) but it is recommended to use iendmad=1 until the results of the `run_mad6t` are considered satisfactory

madclass= 8nm for 8 normalised CPU minutes, or 1nh for 1 normalised hour for the `run_mad6t` LSF jobs where 8nm is often enough but 1nh may be necessary.

After a `set_env`, a `run_mad6t` or `run_mad6t -i` may be performed. The -i option means run interactively and get the output to the screen or redirected to a file and can be useful for testing the mask file and mad6t run. The `platform` option is ignored as mad6t runs are either performed on the desktop or on LSF. For reasons of numerical compatibility all the mad6t runs should be performed on the same type of machine. Subsequently, LSF jobs may be used by run_mad6t, one per seed, and they will run in parallel. In the case of multiple studies in a workspace, either a `set_env "study"` can be performed or the study can be specified on the command as `run_mad6t` "study". In all cases, the success/failure/correctness of these runs should be verified. The script `check_mad6t` checks what it can but it is essential to have a look at the madX output. This output can be found in the most recent `sixtrack_input/mad.run_mad6t*`

directory. Once the madX run has completed successfully for one seed, `iendmad` can be set to the desired value (typically 60/64) and a new `run_mad6t` command performed. Every `run_six`, see later, starts with an internal `check_mad6t`.

6 Problems, cleaning up, and support

There are often problems at this initial stage: at any time the command `rm -r work/* track/* sixtrack_input/*` will completely clean up and allow a restart from the beginning. Help and diagnostics and error messages will be found in the `sixdesk.log` file, one per study [1)].

7 run_six - Launching SixTrack Runs

It is usual to commence a study with some short runs or a few long runs using LSF, perhaps with just a few seeds.

The script `run_six` is used after setting the various variables as explained below. It will automatically launch tracking jobs into the LSF batch system using the batch scripts in the directory `utilities` or into the BOINC job submission buffer. It is recommended to use LSF batch for short runs and for exploratory studies, or when the binary files are required for detailed examination. BOINC is recommended for production runs and for large studies of more than a few thousand jobs. Note that BOINC does not produce the fort.20 graphics nor return any binary files. Only the fort.10 resultt file is returned which is enough for the subsequent post-processing. In addition note that the platforms LSF and BOINC should not both be used in a single study; rather, once the preliminary investigations are complete using LSF, a new study can be cloned to run full scale production with BOINC using the same `sixtrack_input files` thus avoiding another redundant `run_mad6t`. The experienced user is free to modify the script as explained later. The following variables, all exported, as specified in the `sixdeskenv` file are used by this and subsequent scripts.

tunex & tuney The required horizontal and vertical tunes. These may be different for collision and injection as shown later.

emit The normalised LHC emittance.

e0 Energy of reference particle depending on runtype

dpini & dpmax At injection the initial relative momentum deviation is set to 'dpini=0.00075' and at top energy it is set to 'dpini=0.00027'. For the determination of the (non-linear) chromaticity a wider range is used: 'dpmax=0.002' (see below).

kstep Used to define the step width of the phase space angle. For further information see section 7.1. The phase space angle is related to the emittance ratio via $\phi = \arctan\left(\sqrt{\epsilon_y/\epsilon_x}\right)$, where the emittance is defined as $\epsilon_z = A_z * A_z/\beta_z$, for z=x,y.

dimen The dimensionality of phase space can be chosen between 4 and 6. In the latter case the full six–dimensional tracking is done including cavities.

chrom To correct for slight differences between MAD and SixTrack the chromaticity is routinely corrected by setting 'chrom' to 1 and using

chrom_eps=0.0000001 This operation will not be performed for 'chrom=0' but `chromx chromy` will be used instead.

[1)] Help with these procedures is always available from Eric.McIntosh@cern.ch, by telephone, or by SKYPE to mcintosh94. Accelerator Physics issues should be initially addressed to Massimo.Giovannozzi@cern.ch or Riccardo.de.Maria@cern.ch.

chromx=2. and
chromy=2. being the values used when 'chrom=0'.
sussix To determine precise values for the detuning calculation this switch should be set to: 'sussix=1'. It uses the sussix program [**?**]. This option is only valid for the short run configuration (see section 7.1). (Problems have been found when using `sussix` in the 6D case.)

`run_six` can handle three different modes of tracking: Normally initial investigations are carried out with `short` runs (and possibly `sussix`).

1. Short run — This run mode is used to find chromaticity and detuning as a function of δ and amplitude respectively. Typically this is done with just 1,000 turns (activate with `short=1` in `sixdeskenv`). The other variables in `sixdeskenv` for this run are described in section 7.1.
2. Long run — This mode is meant for the dynamic aperture determination proper (activate with `long=1` in `sixdeskenv`). The other variables for this run are described in section 7.2.
3. Differential Algebra (DA) run — If high order Taylor maps are needed this is the mode to use (activate with `da=1`). This mode is mostly for expert use. Please ask the author (FS) how to make best use of it. In this case `run_six` calls the porgrams readda, dalie4 and dalie6 in the directory `bin`.

Note that only one type of run may be chosen at any one time; one and only one of the `sixdeskenv` variables `short, long, da` may be set to 1.

The short/long runs both use seeds as specified by
ista=$istamad Start seed
iend=$iendmad End seed
As shown above, the default seed range `ista, iend` for `run_six` is the same as that used for `run_mad6t`, namely `istamad, iendmad`. These values may be changed at any time, for example to submit jobs for a limited range of seeds, but must clearly be a subset of the run_mad6t values.

7.1 Short Run

ns1s & ns2s Lower and upper amplitude range in beam σ.
nss Amplitude step in beam σ.
turnss Number of turns which is usually set to 1,000 in this mode.
turnsse This variable should be set to the number of zeros of 'turnss', i.e. '3' in our example, it becomes part of the data directory structure. Therefore, if one decides to redo this analysis at say 10,000 turns one specifies 'turnsse=4' and subsequently the data are stored separate from those produced with 'turnsse=3'.
writebins This defines after how many turns data are written to output files. In this mode it should always be set to: 'writebins=1' since all turns are needed to find the tunes as a function of amplitude.
kini & kend Initial and end angle in phase space. Typically set from '1' to 'kmax=5' (see next variable). By specifying 'kini=0' the nonlinear chromaticity is calculated as well (which uses the 'dpmax' setting) and thereafter the initial angle is set back to: 'kini=1'. Note that the variation from 'kini' to 'kend' is done in steps defined by 'kstep'.
kmax This defines the number of phase space angles, e.g. 'kmax=5' means that each step `kstep` is of: $90°/(kmax+1) = 15°$.

reson=0 switch for Guignard resonance calculation

7.2 Long Run

ns1l & ns2l Lower and upper amplitude range in beam σ. This range is sub–divided into ranges of `nsincl` σ. In each job 30 pairs of particles are evenly distributed in each subrange. The close–by pairs are used to find the onset of chaos. Typically we find that a variation 2 σ is sufficiently dense to find the minimum dynamic aperture with a precision of 0.5 σ.

nsincl 2 σ is standard. A smaller step, of 0.5 say, can give better results.

turnsl For the long term tracking we usually track for 100,000 turns or more.

turnsle This variable should be set to the number of zeros of 'turnsl', i.e. '5' in our example.

writebinl This defines after how many turns data are written to output files.

Important: make sure that `writebinl` is large enough otherwise huge amounts of data will be created. Occasionally that may be of use, however in most cases make sure that no more than a total of 1,000 turns are recorded. This implies that for 'turnsl=100000' the variable should be set to at least 'writebinl=100'. When running on BOINC, rather than LSF, the binary files are not returned but SixTrack is checkpointed every `writebinl` turns. It is recommended to set `writebinl` to 10000 for BOINC runs.

kinil & kendl Initial and end angle in phase space. As in the 'short run' mode the variation from 'kinil' to 'kendl' is done in steps defined by 'kstep'.

kmaxl This defines the number of phase space angles, e.g. 'kmaxl=5' means that each steps amounts to: $90°/(kmaxl + 1) = 15°$. Thus the actual angles are computed by dividing 90 by kmaxl+1, so 5, 19, 59 for example are reasonable choices. The choice of `kmaxl` is discussed in Ref. [**?**].

Now the other physics and system parameters must be defined in the `sixdeskenv` file if the defaults are not suitable. When the platform is defined as LSF, LSF job class definitions will be required:

platform=LSF or may be set to BOINC.

longlsfq=1nd sufficient for 100,000 turns, 60 particles

classs=sixmedium for short runs.

classda=sixda for the sixda jobs requiring large memory

sixdeskforce=0 Should normally be left at 0 but may be set to 1 or 2 (see later)

The LSF job class defaults are normally satisfactory, but `longlsfq` should be set to `2nd` or to `1nw` if performing more than 10^5 turns.

ibtype=0 or 1 to use the Erskine/McIntosh optimised error function of a complex number

idfor=1 the closed orbit is added, if set to 0 the initial co-ordinates are unchanged

sixdeskpairs=30 Normal value for 60 particles

Then we have, depending on the `runtyp`:

```
if test $runtype = "inj"
then
```

e0=450000. (energy)

gamma=479.6 (gamma)

dpini=0.00075 (initial relative momentum deviation)

```
elif test $runtype = "col"
then
```

e0=7000000.

gamma=7460.5
dpini=0.00027

```
fi
```

dpmax=0.002 maximum momentum deviation for a short term run

Next we have the tunes again depending on the runtype:

tune=0

In this case the `run_six` will make a special local LSF run to compute the tunes. Alternatively the tunes may be specified and in particular a tunescan can be performed where the tunes will be computed on a straight line from (tunex,tuney) with gradient deltay/deltax up to and including (tunex1,tuney1). The tunes must be 10 .le. tune .lt. 100 in format dd.dd[d][d]. The folowing example specifies the tunes (64.28,59.31) with no scan.

```
if test $runtype = "inj"
then
```

tunex=64.28 Start value
tuney=59.31 Start value
deltax=0.001 Increment to tunex
deltay=0.001 Increment to tuney
tunex1=64.28 End value
tuney1=59.31 End value

Similarly for collision we have:

```
elif test $runtype = "col"
then
```

tunex=64.31
tuney=59.32
deltax=0.001
deltay=0.001
tunex1=64.31
tuney1=59.32

The total number of jobs/cases can be computed as the product of the number of seeds, the number of tune values, the number of amplitude intervals and the number of angles. The total number of cases and progress is reported in the work directory and can be examined with the `run_query` or `run_status` commands.

Each batch job returns a result file `fort.10.gz` to the `track` tree/hierarchy. For LSF studies, if the CASTOR switch is on in the `sysenv` file, all the result files, including the binary files and the fort.6 output, are compressed in a tar file and written to `$CASTOR_HOME/direct_track/*` where the `direct_track` tree matches the `track` tree in AFS.

It is not recommended to run a study of more than 30,000 jobs/cases in a single workspace but up to 100,000 still works. A study can be split over two or more workspaces, perhaps by seed number, and the results combined.

Problems may, indeed often, arise when it is necessary to run the script more than once, either because of a system crash, batch daemon not responding, jobs lost, or some other error. The script `run_six` can be rerun as often as necessary, but by default will not re-submit jobs. The `run_six` can either be submitted as a batch job and/or with a different seed range (`ista, iend`). However, the `run_six` script now maintains a copy of each LSF job in

$sixdeskwork/jobs directory and a file $sixdeskwork/JobIds containing one line for each case. Each line contains the JobName, the directory for the input files, and the fort.10.gz output, and the associated LSF job_ID(s). If run_six is called more than once, it ignores cases where a non-zero fort.10.gz exists, but otherwise deletes the tracking input files and re-generates them. Even simpler recovery is available with the run_missing_jobs command. *It is essential to wait for existing LSF jobs to complete, or to cancel them, and do at least two* run_status *commands before invoking* run_missing_jobs. This procedure has proven to be extremely effective. In the worst of all cases wait for, or cancel, outstanding batch jobs with the bkill 0 command, and delete everything except the sixtrack_input with a rm track/* work/* command and restart.

With BOINC the situation is simpler; near the end of a study a run_incomplete_cases command will re–submit jobs for those cases where no results are available.

8 Monitoring the progress of the study.

The script run_status looks into the database work directory and updates and summarises the status of the study. The run_query command gives a quick look without any updating. It first counts the number of cases in $sixdeskwork/JobIds. It reports the number of LSF batch jobs generated, possibly more than one per case, and does an LSF bjobs to report on job status. Finally it performs a rather time-consuming search to find the number of unfinished cases. It also produces the files completed_jobs, incomplete_jobs and possibly missing_jobs in the $sixdeskwork directory. When all cases are complete the run_join10 procedure can be initiated. While the run_status command may take some time, many minutes for 50,000 cases, it makes it very easy indeed to recover from LSF failures. For example, in a recent study with 43,100 jobs over 800 jobs failed. One run_missing_jobs command re-submitted them automatically to successfully complete the tracking. As an alternative, the run_six command can be re-issued and will not re-submit jobs if sixdeskforce = 0, will not re-submit jobs which have been completed successfully if sixdeskforce = 1 AND a fort.10.gz result file has been created.

There are also cases of database corruption. These are cleaned up, after all running LSF jobs have terminated, by the correct_cases command. Overall the environment has been proven to be rather robust and almost all errors can be recovered.

9 The fort.10 File

The structure of the fort.10 files is shown in Tables 1 and 2, which have been taken from the official SixTrack manual [2] and updated with recent additions. All values are double-precision numbers, values encoded if necessary. There is one (very long) line per particle.

10 The Other SixTrack Output Files and What the Run Environment Does With Them.

By default the scripts automatically tar all output files and store them directly to CASTOR. In order to view these files they have to be copied from CASTOR using the standard CASTOR tools like rfcp, rfrm and nsls, nsrm, nsmkdir etc. The option --help is availble for each of these commands.

Although the full description of all output files is beyond the scope of this note; an ascii file worth looking at is the fort.6.gz which gives an explicit description of all operations and possible failures of the SixTrack run as seen by the program. Also one finds the tracking

Table 1: Post-processing data of the `fort.10` file

# of 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 $\frac{\Delta p}{p_o}$
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

Table 2: Post-processing data of the `fort.10` file continued

# of Column	Description
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)
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	SixTrack Version (encoded in double precision)
53 – 58	Closed Orbit
59	Total turns all particles
60	Tracking CPU time in seconds

data in the binary files `fort.90.gz` down to `fort.61.gz`, which may be useful for further analysis. These and other files are all described in the SixTrack manual [2]. Note that these files are not available from BOINC runs which are for full scale production; LSF should be used for testing and debugging.

11 Post-processing with run_join10

`run_join10` gathers the results of completed jobs and produces combined output files at level 4 in the `track` tree. It also deletes any results of previous `run_join10` commands. The result directory is named `$ns1l-$ns2l` e.g. 14-20 as compared to the other amplitude directories e.g. 14_16, 16_18, etc. There is one additional parameter for run_join10 namely `turnsemax` which is set automatically by `sixdeskenv` to $turnsle or $turnsse for long/short studies. Missing files are reported to `sixdesk.log` but even incomplete results may be useful. This procedure makes use of the utility program `joinf10` in the directory `bin`.

12 Post-processing with run_post

After possibly joining the `fort.10` files they can be postprocessed to find chaotic boundaries and particle losses by using the script `run_post`, which in turn uses the program `read10b` in the directory `bin`.

There are a couple of options for plotting. The `sixdeskenv` variables `iplot` and `kvar` specific to `run_post` are used. If `iplot=1` a plot is produced for each seed and the results can be found in the `plot` directory. Note that setting `iplot` to 1 will produce a possibly huge amount of data, even if it is compressed with gzip. The variable `kvar` should be set to 1 to obtain the DA as a function of angles for a `long` study, and the DA over ALL seeds and angles is plotted for each angle even if `iplot=0`.

The following variables are used, almost certainly with the same values as used by `run_six` and `run_join10`:

kinil & kendl Initial and end angle in phase space. The variation from 'kinil' to 'kendl' is done in steps defined by 'kstep' (see below).

kmaxl This defines the number of phase space angles, e.g. 'kmaxl=5' means that each step is of: $90°/(kmaxl + 1) = 15°$.

kstep Used to define the step width of the phase space angle.

Ampl The amplitude range in sigma. To distinguish the original amplitudes ranges from the joined ranges a hyphen is used '–' instead of an underscore '_'. `Ampl` is set automatically using `ns1s/ns2s` or `ns1l/ns2l`.

turnse This variable should be set to the number of zeros of number of turns processed, i.e. '5' in our example, it is part of the data directory structure.

short In the example 'short=0' means that the mode short run is not activated.

long In the example 'long=1' means that the mode long run is activated.

iplot No plotting for each seed if 'iplot=0'. If this flag is set to '1' or '2' the following graphics are produced:

- Short run
  - Chromaticity, i.e. the tune versus δ.
  - Horizontal and vertical detuning each in one plot for all tracked phase space angles.
  - Tune foot print, i.e. vertical tune versus horizontal tune with the amplitude as a parameter.

- Long run
  - End value of the distance in phase space 'd(turns)' of 2 initially close–by particles as a function of initial amplitude.
  - Fitted slope of $\log d(turns)$ versus $\log(turns)$ of the distance in phase space of 2 initially close–by particles as a function of initial amplitude. For details of the meaning of these two chaotic definitions please refer to reference [**?**].
  - Survival plot, i.e. survival time versus initial amplitude.
  - Horizontal and vertical smear as a function of initial amplitude.
  - Phase space averaged amplitude versus initial amplitude.

  For 'iplot=2' these plots are automatically printed using your normal Linux $PRINT_CMD. Obviously, great care has to be taken to avoid a swamping the printer. The graphics are stored as files like `test.ps.gz` in the directory `sixjobs/plot/"study"` with the same tree structure as the `track` tree.

The result files from `run_post` in long mode are called DAres*, one per angle, and are stored in the `sixjobs` directory itself. The contents and format are defined in the next section 13 Here it should be noted that setting `kvar=1` causes `run_post` to take a long time, even a very long time of many hours, for a large number of angles. If both `iplot` and `kvar` are set to 0 the DAres* files are produced anyway and can be processed with `run_awk` or your own procedures.

13 The DAres files

These files contain the following columns:

- Run name for particular seed.
- "Strict" chaotic boundary via slope method [**?**].
- "Certain" chaotic boundary via large distance in phase space method [**?**].
- Dynamic aperture concerning the phase space averaged amplitude (preferred value).
- Raw dynamic aperture concerning initial amplitude (to be used with care).
- Lower bound of tracked amplitude range.
- Upper bound of tracked amplitude range.

`run_awk` uses the DAres..... files from run_post, reports to the screen, and produces a simple plot file named DAres.....plot.

14 Running studies in batch and/or parallel

In summary a study can be run in the following steps:

1. Create a study by `set_env` and after creating the `mask` file in the `mask` directory do a `run_mad6t`. Repeat as often as necessary until `check_mad6t` is successful and the `mad6t` output has been verified.
2. For a large study use `run_six` in a batch job by:

   ```
   bsub -q1nd 'cd w1/sixjobs;run_six job_tracking LSF'
   ```

   to submit all cases of the study `job_tracking` in workspace `w1` using LSF. The progress of the command can be monitored by e.g. `tail -f sixdesk.log` or `bpeek "LSFJobID"`.
3. Use `run_status` or `run_status job_tracking`, as frequently as convenient until all cases are complete or all batch jobs have terminated. The command `correct_cases` can be used if the database appears to be inconsistent (but only after all LSF jobs have terminated). It checks each case and updates the complete/incomplete status. If there are incomplete cases with LSF do a `run_missing_jobs`. With BOINC use a run_incomplete_cases

to speed up the completion of the study or to handle the tail of incomplete cases. When all cases have been completed do a

4. `run_join10` or `bsub -q1nd 'cd w1/sixjobs;run_join10 job_tracking'` followed by a
5. `run_post` or `bsub -q1nd 'cd w1/sixjobs;run_post job_tracking'`

15 Lockfiles

It is necessary to use locks in order to avoid conflicting modifications to a file. A directory is locked if it contains a file sixdesklock in read only mode (444). If a script fails or dies for some reason or a batch job is killed , this lock file may be left behind. The script `check_locks` reports lock status for the current study; `check_all_locks` for the workspace. If the study itself is locked, it will be reported first as such a lock inhibits further checking. `unlock "directory"` frees the lock. The `unlock_all` unlocks all locks but is deprecated. If in doubt about a lock, and the information shown by the check commands, simply do a `check_lock "directory"` or `cat sixdesklock` and `ls -l sixdesklock` in the relevant directory, to see which script, process and machine are holding the lock and since when. The general philosophy is to lock the study so that only one operation at a time is permitted. Most command wait for the study to be unlcoked but the `run_results` command for BOINC just exits on the grounds that it will be run periodically, nornally as an `acrontab` entry.

16 BOINC

The sixdesk environment was designed to make it as transparent as possible to use LSF (or the now obsolete in house CPSS Windows desktop system) and later the Berkeley Open Infrastructure for Network Computing BOINC [**?**]. This has made over 100,000 PCs available for tracking studies. There is one additional step required, namely `run_results`, to retrieve result files from the BOINC Web server and move the fort.10.gz to the appropriate directory. Only this file is returned, no `fort.6` or binary or graphics files and nothing is written to CASTOR. To use BOINC the `sixdeskenv` file is modified to

platform=BOINC

and that is all.

1. `run_mad6t` operates identically as for LSF.
2. `check_mad6t` as for LSF.
3. `run_six` as for LSF.
4. `run_status` as for LSF.
5. `run_results` must now be called to get the fort.10.gz file from the BOINC server. It can be called regularly, automatically, by using an acrontab entry in AFS, and there is an example `acrontab.entry` in the `sixjobs` directory. In addition a `run_incomplete_cases` script has been implemented to allow the use of multiple tasks for a case. This is particularly useful towards the end of a run, when say 90% of the cases are complete, in order to speed up completion of the study.
   Unlike LSF which returns all result files directly to the `track` tree with no checking and over-writing existing results, BOINC compares the results file fort.10.gz with any existing result and reports any differences to `sixdesk.log`.
6. when all cases are complete the run_join10 and `run_post` procedures are used as with LSF.

17 Modifying the scripts/commands

All modifications should be made to the scripts in the directory `scripts`. The source script beginning with the characters "my" should be modified when it exists rather than the derived script e.g. modify `myrun_six` and NOT run_six. The script `domyseds`, which operates on all the files/scripts in the file `allscripts`, should then be executed to expand macros in scripts named "my...." to produce the actual command scripts, and tocopy all the commands to `../sixjobs`.

Here is a more or less alphabetic list of the "my" scripts:

```
[my]backup_study
[my]backup_workspace
[my]check_all_locks
[my]check_lock
[my]check_locks
[my]check_mad6t
[my]correct_cases
[my]delete_study
[my]dorun_mad6t called by run_mad6t
[my]get_all_betavalues
[my]mywhich
[my]recall_study
[my]recall_workspace
[my]rerun_all_cases
[my]run_awk
[my]run_incomplete_cases
[my]run_incomplete_tasks
[my]run_join10
[my]run_missing_jobs
[my]run_post
[my]run_query
[my]run_results
[my]run_six
[my]run_status
[my]set_env
[my]unlock
[my]unlock_all
```

and here is a list of some other useful scripts/commands without macros.

```
bresume_all
bstop_all
exec_env
get_wpro
minav.awk called by run_awk
print_env
query_all
sub_wpro
```

18 The Subroutines

These subroutines are all defined in the file mydot_profile and may therefore use macros themselves. A `. ./dot_profile` statement is issued when starting a script in order to make them available. (Parameters need to be documented.) They are called by the corresponding "my" macros as shown.

```
sixdeskmess() mymess
sixdeskmktmp() mymktmp
sixdeskmktmpdir() mymktmpdir
sixdeskexit() myexit
sixdeskunlock() myunlock
sixdesklock() mylock
sixdesktunes() mytunes
sixdeskinttunes() myinttunes
sixdeskamps() myamps
sixdeskrundir() myrundir
```

The scripts:

```
[my]dot_boinc
[my]dot_bsub
```

are effectively subroutines used by run_six. The scripts

```
[my]dot_env
[my]dot_profile
```

are called by almost everyone to establish the environment and make the subroutines available.

19 Backing up the workspace

The scratch disks at CERN are not backed up (unlike your $HOME) for the time being. It is recommended that you periodically run a backup so that data can be recovered in case of a scratch disk failure. It is also recommended to a `backup_study [job_tracking]` when a study `job_tracking` is complete. The study may then be deleted safely.

These backups can be done rather simply in a batch job. First is shown an example of backing up your scratch disks and then the rather more sophisticated facilities available for SixDesk tracking studies and workspaces.

NOTA BENE: all castor_backups save the logs in your /castor/logs. Although these are gzipped and might be useful if you forget the contents of a backup you may also just want to delete them to save space in your HOME directory. The contents of backups can be always be determined from the backups themselves. Note also that for all backups, links are NOT followed. Further a link to a non-existent file is NOT backed up.

1. Backing up scratch disks Everyone has a CASTOR account and a default $CASTOR_HOME directory /castor/cern.ch/user/$INITIAL/$LOGNAME. All the ns commands like nsls, nsrm, nsmkdir etc use this by default. In this example all the backups are in the CASTOR directory scratch_backups. In the examples below "date" is of the form yyyy-mm-dd (and mydate is just -yyyy-mm-dd). The batch job script, which must have execute permission, is as follows:

```
#!/bin/sh
# Do a dated (by day) backup of all scratch disks to
```

```
#  $CASTOR_HOME/scratch_backups
# and log the reports in ~/backup_scratch.reports
#
cd $HOME
mydate="-`date -Idate`"
echo "" >> backup_scratch.reports
echo "Scratch backup reports$mydate" >> backup_scratch.reports
echo "Scratch backup reports$mydate"
echo "" >> backup_scratch.reports
for i in `ls -d scratch*`
do
  echo "Backing up $i to scratch_backups/$i$mydate" >> backup_scratch
  echo "Backing up $i to scratch_backups/$i$mydate"
  castor_backup $i scratch_backups/$i$mydate
  nsls -l scratch_backups/$i$mydate >> backup_scratch.reports
  nsls -l scratch_backups/$i$mydate
done
```

If the job runs to completion (as can be checked in the LSF STDOUT or the backup_scratch_reports) a lost disk, scratch0 say, can be restored with a

```
cd $HOME
castor_recall scratch_backups/scratch0"mydate"/scratch0
```

where ”mydate” identifies the backup to be used. nsls scratch_backups will show all available backups. Clearly this can also be done in a batch job. To recall to a different place

```
cd $HOME/scratch99
castor_recall scratch_backups/scratch0"date"/scratch0 .
```

2. Backing up SixDesk workspaces and directories. The intended usage is that in order to free up disk space a study can be backed up and then deleted. It may later be recalled to exactly the same workspace or to a new different workspace. The backup_workspace might be used to backup a complete set of studies. A study may be recalled from either a workspace backup or a study backup.

Deleted studies are never backed up; a non-deleted study cannot be recalled to the same workspace. Deleting a study does NOT delete sixjobs and all sixdeskenv/sysenv files are kept in the studies directory. All DAres* files are also preserved.

All backup/recall commands should be issued from the sixjobs directory as usual except for recall_workspace where you must be in the workspace itself.

In these examples ”date” is of the form ddmmyy. So, normally, for example:

```
cd ~w1/sixjobs
set_env job_tracking
backup_study
delete_study
```

and subsequently

```
cd ~w1/sixjobs
set_env job_tracking
recall_study w1%job_tracking%"date"
```

or

```
cd ~w1/sixjobs
set_env job_tracking
recall_study w1%job_tracking%"date" w99
```

In the latter case the sixdeskenv files are edited to reflect the new workspace. In all recalls an existing sixjobs directory is never overwritten, but in all cases the sixtrack_input, track, work, and plot data are recalled along with logfiles if possible.

To recall several studies to a new workspace, the first recall will recall sixjobs as well as the data; afterwards, cd to the workspace to recall other studies.

As mentioned a study can be similarly recalled from a workspace backup by

```
set_env job_tracking
recall_study w1%"date" to restore it to w1
```

or

```
set_env job_tracking
recall_study w1%"date" w99 to a new empty workspace.
```

Finally a complete workspace can be recalled to the same workspace or a different one.

To backup a workspace

```
cd ~/w1/sixjobs
backup_workspace
```

and to recall

```
cd ~/w1
recall_workspace w1%"date"
```

or

```
cd ~/w99
cp SOMEWHERE/recall_workspace .
./recall_workspace w1%"date"
```

All backups can be found in $CASTOR_HOME/workspace_backups with names like
w1%job_tracking%"date" for a study backup
or
w1%"date" for a workspace backup.
nsls workspace_backups will list them all.

All backups are automatically restartable from checkpoints as they may take some time and are subject to the usual system failures. A file backup_study.list or backup_workspace.list is used for restarting from the point of failure. Thus a backup must be completed before a new backup is started in the same workspace (or the file backup_study.list or backup_workspace.list must be deleted). The workspace is locked to ensure only one backup at a time and no switching of studies.

These procedures may seem complicated but the simple backup, delete, recall are easy to perform and the complications are necessary to avoid destroying data and to recover from system failures. Details of the CASTOR backups/ recalls can be found in the castor_backup.log or castor_recall.log.

20 Additional Notes

The scripts `dot_env`, `dot_errmess`, and `dot_exit` are called by the others to obtain the lock and set the environment, issue error messages and log them, and to exit freeing the lock.

The scripts `dot_bsub` and `dot_boinc` are called by `run_six` to submit a batch job to LSF, or a task to BOINC.

21 Other lattices

Clearly any lattice file may be processed by the run environment not just the LHC. The mask file has to be changed as appropriate. Other changes are described in the main part of this note, including aesthetic name changes. Also required are definitions of the `runtype` (see section 5). This variable and the variable `beam` specify which of the `fort.3.mother1` and `fort.3.mother2` from the directory

```
sixjobs/control_files
```

to use. The value of the `runtype and beam` variables simply specify the suffix of the files to be used from this directory. For how these files are constructed the user is refered to the SixTrack manual [2].

22 Acknowledgements

Thanks to R. Demaria, M. Giovannozzi and T. Risselada for acting as guinea pigs and for many helpful suggestions.

References

[1] M. Hayes and F. Schmidt, "Run Environment for SixTrack", LHC Project Note 300, (see also http://wwwslap.cern.ch/frs/SixTrack_run_environment/manual.ps).

[2] F. Schmidt, "SixTrack: Version 3, Single Particle Tracking Code Treating Transverse Motion with Synchrotron Oscillations in a Symplectic Manner, User's Reference Manua", CERN/SL/94–56 (AP)
(see also http://wwwslap.cern.ch/frs/Documentation/doc.htmlx).

[3] H. Grote, "Statistical significance of dynamic aperture calculations", Beam Physics Note 34.

[4] http://physics.nist.gov/cuu/Constants/index.html http://physics.nist.gov/cuu/Constants/index.html.

[5] R. Bartolini and F. Schmidt, "SUSSIX: A Computer Code for Frequency Analysis of Non–Linear Betatron Motion", presented at the workshop "Nonlinear and Stochastic Beam Dynamics in Accelerators – A Challenge to Theoretical and Computational Physics", Lüneburg, September 29 – October 3, 1997, CERN SL/Note 98–017 (AP), http://wwwslap.cern.ch/frs/report/sussix_manual_sl.ps.gz http://wwwslap.cern.ch/frs/report/sussix_manual_sl.ps.gz.

[6] M. Giovannozzi and E. McIntosh, "Parameter scans and accuracy estimates of the dynamical aperture of the CERN LHC", EPAC'06, June 2006, Edinburgh.

[7] M. Böge and F. Schmidt, "Data Organisation for the LHC Tracking Studies with SIXTRACK", LHC Project Note 99,
http://wwwslap.cern.ch/frs/report/lhc_pro_note99.ps.Z http://wwwslap.cern.ch/frs/report/lhc_pro_note99.

[8] M. Böge and F. Schmidt, "Estimates for Long–Term Stability for the LHC", LHC Project Report 114, presented in part at the Particle Accelerator Conference, Vancouver, 12–16 May, (1997), AIP Conference

Proceedings 405 (1996), http://wwwslap.cern.ch/frs/report/conj97lhc.ps.Z http://wwwslap.cern.ch/frs/report/conj97lhc.ps.Z,
the poster version: http://wwwslap.cern.ch/frs/report/conj97post.ps.Z http://wwwslap.cern.ch/frs/report/conj97post.ps.Z
and contribution to the workshop on “New Ideas for Particle Accelerators”, Santa Barbara, November 1996.

[9] “Berkeley Open Infrastructure for Network Computing”, http://boinc.berkeley.edu