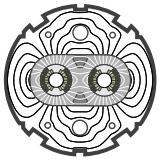

LHC Project Note 300

July 26, 2002
Frank.Schmidt@cern.ch

Run Environment for SixTrack

M. Hayes and F. Schmidt

Keywords: software, sixtrack, dynamic aperture

Summary

This note will describe how massive tracking campaigns can be performed with SixTrack [1] starting from a MAD-X input file of the LHC lattice. The idea is to launch these runs as automatically as possible with minimal knowledge of either UNIX scripts or SixTrack input.

1 Set-Up

In this note it is assumed that the user wishes to track through the LHC lattice on the public LSF cluster at CERN known as LXBATCH. However the run environment is modifiable to work on private machines, queues and with other machine lattices (see appendices D and E).

The user is assumed to have a large amount (roughly 1Gb) of disk space available for himself on AFS, this is normally referred to as `w1` workspace at CERN. The user must at this point contact his system administrator to arrange this space. This note will refer to this disk space as the user's `w1` directory.

The user should then create the directory tree by changing to this `w1` directory and running the command:

```
gtar xvfz\
/afs/cern.ch/group/si/slap/share/\
sixtrack/run_environment/run_environment.tgz\
COREHOME
```

This will create the entire directory subtree necessary starting with a sub directory `COREHOME`. The aim of this note is to jump-start an uninitialised user. Therefore, only a minimal set-up is described. This set-up allows tracking at injection energy for the example error table 9901 with a tune split of 5 (64.28, 59.31), without linear coupling terms and without beam-beam. The scripts are provided with sensible defaults for this scenario, so for an initial test run almost all variables can be left as their default values. The only exception is specifically mentioned in section 4.2. For more detailed help please contact the authors.

2 Overview

The run environment is a collection of four scripts which are all run from the directory

`COREHOME/track/sixjobs`

They are normally run in the following order:

1. `run_mad6t` This runs the MAD-X to SixTrack converter.
2. `run_six` This launches SixTrack jobs on the chosen batch processing farm.
3. `run_join10` This gathers already tracked jobs together and produces combined output files.
4. `run_post` This analyses either individually tracked jobs or jobs that have been combined using `run_join10`.

The scripts should normally be edited beforehand to specify the user's requirements. The user's favourite editing package may be used for this. The following changes are required to all scripts before running:

basedir This is the path to user's `w1` directory

LHCVers Version of LHC which is presently 6.

LHCSubVers The subversion in this example is 4.

LHCDescrip The name of the user's `.mask` file without the `.mask` ending (see section 3). In the example this is simply `lhc`. It is used as part of a directory tree to allow a separation from other runs.

LHCfullDescrip The long descriptor is meant to give more information about the run. It is made up of the `LHCVers`,`LHCSubVers` and `LHCDescrip`. A directory will be created at:

`<user's w1 directory>/sixtrack/fort16/fort16_$LHCfullDescrip`.

All SixTrack files needed for running are copied to this directory.

ista & iend Initial and final seed numbers. Typically the seeds are varied between 1 and 60. This insures to 95% that only 5% of all possible seeds exhibits a smaller dynamic aperture than the 60 seeds tested [2].

In the following sections the scripts are described in detail, with short descriptions of possible extra changes to be done. The example LHC lattice provided with the run environment is also referred to and the result of this note is the calculation of 1000 turns dynamic aperture of this lattice.

3 run_mad6t - MAD-X to SixTrack Conversion

The starting point is the user's MAD input file of his favourite LHC version (the provided example is v6.4). This file is assumed to load in and set up the machine the user wishes to track. 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 by the conversion script by a proper seed number. The example as provided only generates six seeds (to make it quick to run). A standard dynamic aperture study requires at least 60 seeds.

This is modified MAD input file is then known as *the mask file* and should be placed in the directory

```
COREHOME/track/sixjobs/mask
```

The provided example can be found in this directory as the file lhc.mask.

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 general variables described in section 2 should be set. Finally one variable specific to this script has to be set:

runtype This is the type of run required. As provided for the LHC this may be set to "inj" (injection energy) or "col" (collision energy). For more information on how to define runtypes for either the LHC or other machines please see appendix E.

Once these have been changed the script may be launched, which then produces the SixTrack input files consecutively for each seed. The duration depends on the complexity of the machine simulated. Up to three files per seed are produced: fc.2 (the main structure), fc.8 (the misalignments) and fc.16 (the higher order multipole fields). If the user hasn't set up misalignments (as in the example) or field errors the respective file will not be produced and an error message to the effect will be generated. If this is expected, the messages can be safely ignored. The fc.2 file must always be produced[1]. When run_six starts it prints out its temporary working directory. This directory contains, among other things, the MAD output per seed and can be useful for debugging purposes.

[1]The fc prefix generated by the MAD-X converter is then renamed into a fort prefix for historic reasons. If the file doesn't exist this will also produce an error message.

4 `run_six` - Launching Runs

The script `run_six` can be run after the usual adjustments and will automatically launch tracking jobs into the LSF batch queue[2]. The general variables listed in section 2 have to be set. The following variables are specific to this script.

tunex & tuney The required horizontal and vertical tunes.

emit The normalised LHC emittance.

e0 γ at injection energy.

dpini & dpmax At injection the initial relative momentum deviation is set to 'dpini=0.00075'. For the determination of the (nonlinear) chromaticity a wider range is used: 'dpmax=0.002'.

kstep Used to define the step width of the phase space angle. For further information see section 4.1. 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. This operation will not be performed for 'chrom=0'.

sussix To determine precise values for the detuning calculation this switch should be set to: 'sussix=1'. It uses the sussix program [3]. This option is only valid for the short run configuration (see section 4.1).

`run_six` can launch three different modes of tracking:

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`). The specific variables for this run are described in section 4.1.

2. Long run — This mode is meant for the dynamic aperture determination proper (activate with `long=1`). The specific variables for this run are described in section 4.2.

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.

Note that only one type may be done at any one time.

[2] To this end it uses the batch scripts in `COREHOME/track/utilities` The experienced user is free to modify these scripts. More details are available in the appendices.

4.1 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'. Take note that the variation from 'kini' to 'kend' is done in steps defined by 'kstep' (see section 4).

kmax This defines the number of phase space angles, e.g. 'kmax=5' means that each steps amounts to: $90^\circ/(kmax+1) = 15^\circ$.

4.2 Long Run

ns1l & ns2l Lower and upper amplitude range in beam σ. 30 pairs of particles are evenly distributed between 'ns1l' & 'ns2l'. 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.

Important: for a meaningful example run please make four subsequent runs with 'ns1l' & 'nsl2' being varied as: 10_12, 12_14, 14_16 and 16_18.

turnsl For the long term tracking we usually track for 100,000 turns. In the example it has been reduced to 1,000 for fast turnaround.

turnsle This variable should be set to the number of zeros of 'turnsl', i.e. '3' in our example. For the typical long–term run over 100,000 turns it has to be set to: 'turnsle=5'.

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 'writebinl=100'.

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: $90^\circ/(kmaxl+1) = 15^\circ$.

4.3 The output of `run_six`

When it is executed `run_six` begins to generate jobs and submits them to the appropriate batch queue. The output helps to identify how far it has reached. Apart from error messages this may be safely ignored. If an error message occurs, the temporary directory the script runs from is printed at the start of execution and contains all information required for debugging[3].

Long run jobs (100,000 turns) take between one and two hours each on a single LXBATCH machine. This depends mostly on the compexity of the lattice and the number of multipoles (including field errors). Typically a complete case requires tracking for 60 seeds, five phase space angles and two amplitude ranges, i.e. 600 jobs, which can be done in a day.

To monitor progress of the jobs one can use the command `bjobs`. For more information on the queueing system please consult the CERN computing web information on LSF batch queues. In special cases the code may be speeded up (see appendix A for more information).

After the jobs have been completed the results are written into two places (the directory structures are described in reference [4]). For the above example the first place is the directory tree starting with

`<user's w1 directory>/COREHOME/track/v6/s4/...`

There one finds the final SixTrack output file `fort.10.gz` as described in Appendix B. However the user does not have to look at this file since it is processed by the scripts `run_join10` and `run_post`.

Important: There is a safety feature built into the script `run_six`: A run will *not* be executed if there is a non empty `fort.10.gz` at the proper output location. If for some reason the run has to be repeated one has to first delete this file or remove the directory tree it is in[4].

Where all the other SixTrack output files have been stored is described in appendix C. These are normally necessary if the data stored in `fort.10.gz` appears wrong.

4.4 Tidying up after SixTrack

For details on special scripts to aid in the backing up of the `fort.10` data to tape (and thereby freeing more space) please see appendix F.

The run scripts copy the necessary input files for execution and the batch system records problems during the execution to the location:

`<user's w1 directory>/tmp`,

It is the user's responsibility to delete these directories from time to time. It is normally uninteresting to keep these `tmp` files once the tracking has worked.

[3] Most bugs are due to an initial run of SixTrack which finds the beta functions at the start of the lattice in order to calculate where to launch the particles. If this goes wrong all job submissions for that seed will fail. Check the `lin` file in the temporary directory for information as to why. If this is the case the file `betavalues` in the approriate `<user's w1 directory>/COREHOME/track/v6/s4/<mask file name>/<seed no.>` directory will also have to be removed before rerunning.

[4] If certain jobs crash due to system problems and not due to SixTrack errors, they will leave 'gaps' of missing `fort.10` files. These gaps may be filled using this feature by simply rerunning the `run six` script. These system problems include network errors and lost AFS cookies.

5 `run_join10` - Joining Runs

After all the batch jobs have run, the `fort.10.gz` files produced may be combined using the script called `run_join10`. The general variables from section 2 have to be set and the following ones specific to this script:

jsta & jend Initial and final range of amplitudes based on the values in the array e[]. The array 'e[jsta]' till 'e[jend+1]' is filled with the bounds of the amplitude ranges, e.g. 'e[jsta]_e[jsta+1]'. The joined file will be placed in a directory name 'e[jsta]–e[jend+1]'. To distinguish the original amplitudes ranges from the joined ranges a hyphen is used '–' instead of an underscore '_'.

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 steps amounts to: $90°/(kmaxl + 1) = 15°$.

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

turnsesta & turnsemax These variables refer to the number of zeros of the number of turns. They specify the range of number of turns to be combined. In most cases 'turnsesta = turnsemax'.

In our example it is expected that `run_six` has been run for the 4 amplitude ranges: 10_12, 12_14, 14_16 and 16_18. The execution of `run_join10` will then produce for each phase space angle a subdirectory '10–18' with the joined `fort.10` files.

6 `run_post` - Postprocessing Runs

After possibly joining the `fort.10` files they can be postprocessed to find chaotic boundaries and particle losses by using the script `run_post`. The general variables as described in section 2 have to be set as well as the following script specific ones:

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 steps amounts to: $90°/(kmaxl + 1) = 15°$.

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 '_'.

turnse This variable should be set to the number of zeros of number of turns processed, i.e. '3' 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.

a0 & a1 For the plotting in the ‘long’ mode the plotting range is chosen between ‘a0=8’ & ‘a1=20’ in units of beam σ.

iplot No plotting for ‘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 [5].
  - 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 to printer `112-4C15-HP4M`. Obviously, great care has to be taken to avoid a swamping of the printer. The graphic is stored as a file `test.ps.gz` at the location starting with

```
<user's w1 directory>/COREHOME/track/sixjobs/plot/v6/s4/....
```

The long mode postprocessing writes out a file which is starts with `fort.18`. Up to five of these files (one per angle scanned) can be written out. These are described in the next section.

7 The `fort.18` files

These files contain the following columns:

- Run name for particular seed.
- “Strict” chaotic boundary via slope method [5].
- “Certain” chaotic boundary via large distance in phase space method [5].
- 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.

There is a small awk script that finds the minimum and average dynamic aperture in this file and is invoked in the following fashion:

`awk -f minav.awk fort.18.v64lhc3.3`.

Of course, it is only useful if several seeds have been tracked.

8 Acknowledgements

We are very thankful to E. McIntosh who made the NAP system possible and for his continuous and enthusiastic support for that system and all other systems. M. Böge helped greatly in upgrading the UNIX scripts.

References

[1] 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).

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

[3] 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.

[4] 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.

[5] 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,
the poster version: http://wwwslap.cern.ch/frs/report/conj97post.ps.Z
and contribution to the workshop on "New Ideas for Particle Accelerators", Santa Barbara, November 1996.

A Speeding up SixTrack

SixTrack has been optimised for speed. The set-up provided with the run environment allows full tracking of any combination of misalignments and errors.

However if a particular tracking run requires no tilt errors a 10% faster version of SixTrack is available by replacing the line in `run_six`

```
classl=sixtracking_direct_tilt
```

with

```
classl=sixtracking_direct_fast
```

Please Note:

1. This ignores all tilt errors but skew elements are still treated correctly.
2. This gives no warning that it ignores tilt errors. Please use cautiously.

B 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 [1].

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

In the supplied set-up, the scripts automatically tar all other output files and store them directly to castor storage in a similar directory structure as the `fort.10` files, i.e.

```
/castor/cern.ch/user/m/myself/COREHOME/track/v6/s4/...
```

where `m/myself` is user's initial followed by his userid. In order to view these files they have to be copied from castor (using `nsls` and `rfcp`) and then untarred.

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 also finds the tracking 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 complete SixTrack manual [1].

If the direct to castor facility has to be disabled, please change the line in `run_six`

```
classl=sixtracking_direct_tilt
```

to

```
classl=sixtracking_tilt
```

This will then create the directory structure directly to the user's `w1` directory.

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 – 58	Dummy
59 – 60	Internal use

D Other systems

The scripts `run_mad6t`, `run_join10` and `run_post` have been designed to run on any machine connected to AFS. If AFS is unavailable or the user wishes to use a local disk, the following changes have to be made:

- The `basedir` command has to be set appropriately.
- The directories

  `COREHOME/track/sixjobs/bin`

  and

  `COREHOME/track/sixjobs/inc`

  have links to binaries used by the system. These will have to be replaced by the actual binaries (for the appropriate architecture).
- The castor-free version of the utility files will probably have to be used. See appendix C for details.

The `run_six` script needs these changes and in addition requires:

- The absolute path of the SixTrack executable in the utilities directory and in the script itself has to be changed.

If the batch processing system is not LSF or a different configuration of it, the `bsub` commands in `run_six` will have to be updated appropriately for the system. If no batch processing system is available the utility file may be run directly from `run_six` by simply removing the `bsub` commands. Please note that LSF options are also present in the utility files.

E Other lattices

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

`COREHOME/track/sixjobs/control_files`

to use. The value of the `runtype` variable simply specifies 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 [1].

F Castor Backup

There is a problem of disk space taken by the vast amount of tracking runs. Although all files are compressed a non negligible amount of space is taken for each run (see the comment in section 4.2 concerning `writebinl`). Clearly each user has only the amount of space allocated to him in his `w1` directory. As he begins to track more and more cases the amount of space left reduces and he is forced to move older cases to long-term tape storage. CERN provides the castor system for this. A set of scripts to do backups to castor is provided. Eric McIntosh is thanked for providing the following scripts:

castorbackup This allows to backup the data directories independent of their size by producing a series of 'tar' files each with a little less than 1Gbytes onto `/tmp` which are then transfered to castor and deleted from `/tmp`. In our example the data directory tree `lhc` can be backed up via:

```
castorbackup <user's w1 directory>/COREHOME/track/v6/s4/lhc \
                    /castor/cern.ch/user/m/myself
```

where in `m/myself` is the first letter of the user's userid followed by the user's userid. Help can be acquired with

```
castorbackup -h
```

In case the user is not registered on 'castor' the script will inform them whom to contact.

castorrecall Script to recall files to the original or any other location. It will in fact recall all the 'tar' files created in the backup operation.

sixmkdir Make a subdirectory on castor.

sixrm Removal of files or directories (option -r) on castor.

sixdir Listing directories on castor.

Clearly the user is free to use the standard castor tools (`rfcp`,`rfrm` and `nsls`) as well.