COMMAND LINE INTERFACE TO TRACY LIBRARY

B. Nash, Brookhaven National Laboratory, Upton, NY 11973, USA

Abstract

We describe a set of tools that interface to the Tracy particle tracking library. The state of the machine including misalignments, multipole errors and corrector settings is captured in a 'flat' file, or 'machine' file. There are three types of tools designed around this flat file: 1) flat file creation tools. 2) flat file manipulation tools. 3) tracking tools. We describe the status of these tools, and give some examples of how they have been used in the design process for NSLS-II.

INTRODUCTION

There are a large number of particle tracking codes in use in the accelerator physics community. There is MAD8, MAD-X, PTC, SIXTRACK and others. The strengths and weaknesses of each may be described in terms of physics capabilities, ease of interface, and managability of the source code. The Tracy code has a long history with changes that affected each of these areas.

First, we should clarify that at present, most versions of Tracy are libraries, and that to use them to track particles, one must write C/C++ code that links to this library, or calls the functions. Although this provides great flexibility in some sense, due to the full power of a compiled language, this paradigm does not lead to stable units of computation that may be used to build other computations. In essence, one must continually write one's own interface. From this perspective, the Tracy library provides raw material out of which to build a tracking code, but is not optimal as is.

A tracking code must create an internal model of the accelerator to be tracked, and then have mechanisms for tracking through each element. Typically, the internal model is created in two steps. In the first step, a lattice file is used to represent the basic set of element families and their layout. This may consist of many repeated cells. Next, errors may be introduced that distinguish one cell from another. Thus, the lattice is no longer symmetric. One may call this structure 'flat'. The steps in the creation of this 'flat' file may be rather elaborate, involving various error/assymetries, and corresponding correction algorithms, that attempt to model what occurs in a real machine. In this case, one may want to represent a given configuration by a so-called 'flat file' which is simply an element by element description of that configuration. It is then desirable that a tracking code can read in that flat file and track particles directly. This breakdown was originally described by Forest and Nishimura [1]. This approach is also essentially followed by the code Elegant which contains a lattice file and an additional parameter file that can introduce the appropriate errors to create the flat structure.

The purpose of the present paper is to describe a solution to creating an interface to Tracy, centered around the creation, manipulation, and tracking of flat files.

TOOLS

The tools are broken into flat file creation tools, flat file manipulation tools, and tracking tools.

Flat File Creation: Parameter File Lattice File, Errors File

The flat file is created using a tool called *leac* ('load errors apply correction'). The code reads in a parameter file that includes the ingredients needed to make the flat file. An example is shown below

```
# input files
in_dir        ./
lat_file      CD3July17_DW_IVU
fe_file       specsApr28.fe
ae_file       CDR.ae
ap_file       basic.aper
# flags
bare_dynap    false  // if to compute DA
# parameters
s_cut         2.0  // sigma to truncate gaussian
n_stat        1    // num of flat files to create
n_scale       3    // times to scale misalign. errors
n_orbit       5    // iterations for orb. corr.
bpm_name      BPM  // BPM name
h_corr        HCM  // h. corrector name
v_corr        VCM  // v. corrector name
gs            GS //girder start symbol
ge            GE // girder end symbol
#
n_lin         0  // iterations for coupling corr.
qt            SQ
disp_wave_y  20.0e-3 // value of disp. in disp. wave
VDweight      1e3   // weight for vert. disp. in SVD
HVweight      1e0   // weights in SVD for coup. corr.
VHweight      1e0   // weights in SVD for coup. corr.
#
N_calls       6   // ID corr num calls
IDCquads      QH2 QH3 QL1 QL2 QL3  // quads for ID corr.
N_steps       8         // number of steps in ID corr.
N_Fam         5         // number of quad families used
scl_nu        1.0  //weight for SVD for global tunes
```

In this parameter file, the lattice file is named 'CD3July17_DW_IVU.lat' the field error file is 'specsApr28.fe', the misalignment file is 'CDR.ae', and the physical aperture file is 'basic.aper'. The net result of this is to create a flat file based on this lattice, with misalignments, field errors and apertures based on these files. Two correction algorithms are available: orbit correction, and optics correction due to insertion devices. (See [4] for more details on this algorithm.) The other parameters in the file set parameters for these correction algorithms. The flat file is

simply a listing of elements. A drift, for example, is given by

```
dh0                  4     1     1
   0    0    0
 -5.000e-02  3.799e-02 -1.250e-02  1.250e-02
  4.650e+00
```

The first line gives the element name followed by the family number, the kid number, and then the element number. In this example, the drift element called 'dh0' is family number 4. This is the first of such elements and the first element in the sequence. The second line covers the type code, the integration method and the number of integration steps: i.e. specifying the parameters for the symplectic integration through the element. The third line gives the rectangular physical apertures for the element. The final line gives the length. The types are as follows: marker: -1, drift: 0, multipole: 1, cavity: 2, thin kick: 3, wiggler: 4.

An insertion device using Halbach expansion may be specified in the flat file as follows:

```
ivu1                76     1   853
   4    1 600
 -5.000e-02  3.799e-02 -1.250e-02  1.250e-02
  3.000e+00  2.000e-02
 1
  1  0.000e+00  1.0992e-01  0.000  0.000  0.000
```

As in the example of the drift, the first line is the family number, kid number and element number. Thus, this in vacuum undulator has family number 76, this is the first such IVU, and it is the 853 element in the sequence. The next line says that the element is a wiggler, using symplectic integrator 1 with 600 integration steps. The next line gives physical aperture limits. The next line gives the IVU length in meters, (3 m, in this case) followed by the wavelength in meters (20 mm in this case). Next we get the number of harmonics, followed by a listing of them. Here we just have a single harmonic. The harmonics are specified by first giving the harmonic number, then the value of k_{xv} (1/m), $\frac{B_y}{B\rho}$ (1/m), k_{xh}, $\frac{B_x}{B\rho}$ (1/m), ϕ.

The additional files are the alignment error file (.ae), field error file (.fe), and aperture file (.aper). An example alignment error file is

```
seed 2310
girder rms   100.0e-6 100.0e-6 0.5e-03
quad rms    30.0e-6  30.0e-6 0.2e-03
sext rms    30.0e-6  30.0e-6 0.2e-03
```

The seed gives the seed for the random number generator in case of random misalignments. The lines specify element names or family names, followed by rms or sys to specify whether to use random or systematic errors. The following columns give the horizontal then vertical then roll errors. In the case of rms, these are the rms values for a Gaussian distribution, truncated at s_cut as given in the parameter file.

An example field error file is

```
sext rms 0.025 20 1.0E-05 1.0E-05
quad sys 0.025 6 1.0e-4 0.0
```

Finally, an example aperture file is

```
all -50e-3 38e-3 -12.5e-3 12.5e-3
ivu1 -50e-3 38e-3 -2.5e-3 2.5e-3
qm2 -30e-3 30e-3 -10e-3 10e-3
```

This gives all elements horizontal apertures of -50 mm, 38 mm, and vertical apertures of +/- 12.5 mm. It also puts small vertical gaps of +/- 2.5 mm in the IVU and +/- 30 mm horizontal apertures in the maximum dispersion quadrupoles to represent photon absorbers.

FLAT FILE MANIPULATION TOOLS

Once we have the flat file, we may want to do things to it that would correspond to actions on a real machine. For example, change the tune or the chromaticity. These tools perform these actions and replace the old flat file with a new one with the new settings. The manipulation tools that are available are

```
add_aper
add_field_err
change_chrom
change_tune
id_scale
idcor
orbcor
skewcor
```

These commands respectively add physical apertures, add multipole field errors, change the chromaticity (using two sextupole families), change the tune (using two quadrupole families), scale on or off the insertion devices, perform a global optics correction to the insertion devices, perform an orbit correction, and perform a coupling correction. The first five of these have been used frequently, whereas the others may require further work.

TRACKING TOOLS

Once we have a flat file we are interested in, we may perform various tracking actions. We can compute the linear lattice functions, we may compute the eigenemittances and equilibrium beam sizes, or we can track particles with given initial conditions. The code is connected to Laskar's NAFF routines so that tracking may be used to create a frequency map. Some selected calculation tools are shown in the following Table.

tool	action	file extension(s)
get_cod	closed orbit around ring	cod
dnu	compute $\nu_{x,y}(x, y, \delta)$	nudx, nudz, nudp
dynap	find the dynamic aperture	da
fmap	x-y frequency map	fmap
fmapdp	$x - \delta$ frequency map	fmapdp
get_em	find emittances	
get_sig	find beam envelopes	sig
get_twiss	find Twiss parameters	twi
magtol	find DA vs. field errors	mag
track	track initial condition	trk
tsck_full	mom. aper. and Lifetime	ma

An additional feature that has been implemented is naming of the output files based on the flat file name. This allows one to keep track of different configurations within the same directory, and provides a self-documenting ability by having the parameter file describe the flat file configuration.

EXAMPLES

These tools have been used during the design process for the NSLS-II lattice. An example $x - y$ frequency map with errors is shown in Fig. 2 and tunes vs. momenta are shown in Fig.1 . By computing a frequency map along with tune variations and other quantities, a given solution may be evaluated. One study which benefited from these tools was in understanding the impact of multipole errors on the dynamic aperture and Touschek lifetime. More details of these studies are given in [5] and [6].

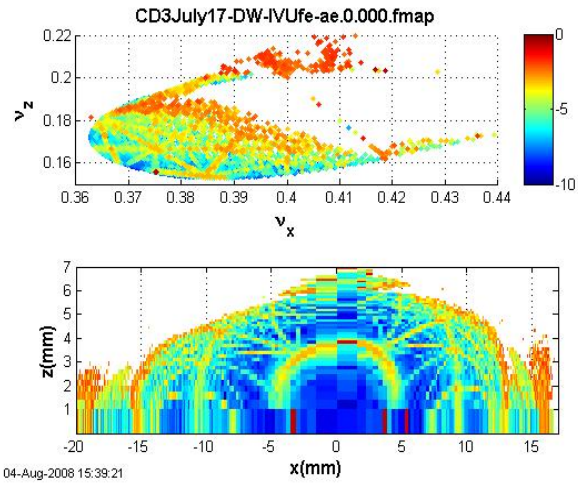

Figure 1: Example of an $x - \delta$ frequency map with errors.

FURTHER INFORMATION

The tools may be downloaded from a Subversion repository on Sourceforge[7]. Many of the tools are documented on a SourceForge wiki page [8]. For further information on these tools, please contact the author.

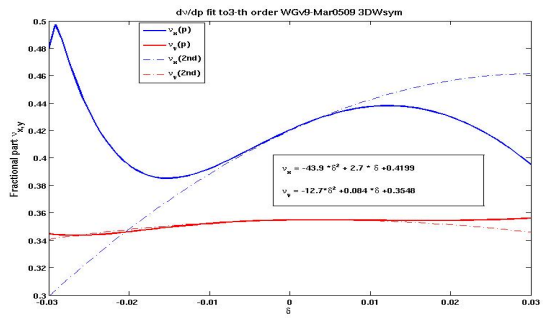

Figure 2: Tunes vs momentum computed with the NAFF algorithm on NSLS-II lattice including damping wigglers.

ACKNOWLEDGMENTS

We thank the NSLS-II design team, and in particular Sam Krinsky for support of this work. We acknowledge Johan Bengtsson for providing us with a version of the Tracy library and for help in writing the flat file creation tool. The C version of Tracy was translated from Pascal by Michael Boege who also wrote the eigenemittance algorithm. The orbit correction code was written by Igor Pinayev, the insertion device correction algorithm by Timur Shaftan, the frequency map analysis software and link to Radia kick maps by Laurent Nadolski. Special thanks to Lingyun Yang for much help with the open source process, replacing the Numerical Recipes library with GSL, and much more help with general programming. It has been a pleasure to collaborate with him. Finally, much thanks to Steve Kramer for learning these tools and using them to help us understand the beam dynamics of NSLS-II. The results he found with these tools along with the ability to cross check other codes have made the effort worthwhile.

REFERENCES

[1] E. Forest and H. Nishimura, "Vertically Integrated Simulation Tools for Self Consistent Tracking and Analysis," *In the Proceedings of IEEE Particle Accelerator Conference, Chicago, Illinois, 20-23 Mar 1989, pp 1304.*

[2] M. Borland, "elegant: A Flexible SDDS-Compliant Code for Accelerator Simulation", LS-287

[3] S. Krinsky, "Accelerator Physics Challenges for the NSLS-II Project", Proc. PAC '09

[4] T. V. Shaftan, J. Bengtsson and S. L. Kramer, "Control Of Dynamic Aperture With Insertion Devices,", Proc. EPAC '06

[5] B. Nash and W. Guo, Impact of Higher-Order Multipole Errors on Dynamic and Momentum Aperture, Proc.PAC09.

[6] B. Nash and S. Kramer, "Touschek Lifetime Calculations for NSLS-II", Proc. PAC '09

[7] http://libtracy.svn.sourceforge.net/

[8] http://libtracy.wiki.sourceforge.net/