EVOLUTION OF THE TRACKING CODE PLACET

A. Latina, Y. Levinsen, D. Schulte, CERN, Geneva, Switzerland
J. Snuverink, John Adams Institute at Royal Holloway, University of London, Surrey, UK

Abstract

The tracking code PLACET simulates beam transport and orbit corrections in linear accelerators. It incorporates single- and multi-bunch effects, static and dynamic imperfections. A major restructuring of its core has resulted in an improvement in its modularity, with some immediate advantages: its tracking core, which is one of the fastest available for this kind of simulations, is now interfaced toward three different scripting languages to further expand its simulation capabilities: Tcl/Tk, Octave, and Python. These three languages provide access to a vast and diverse library of scientific tools, mechanisms for parallel computing, and access to Java interfaces for control systems. Also, several new functionalities have been added to the PLACET core itself: parallel tracking to exploit modern multicore CPUs and clusters of computers, the possibility to track through the interaction region in presence of external magnetic fields (detector solenoid) and higher order imperfections in magnets. PLACET is currently used to simulate the CLIC Drive Beam, the CLIC Main Beam, CTF3, FACET at SLAC, and ATF2 at KEK amongst others.

INTRODUCTION

The PLACET tracking code [1] is in constant evolution to cope with the new simulation needs of its user community and with the R&D for future linear colliders. A constant effort is also made in order to update the code and take advantage of the most modern computer architectures, while keeping its interface easy and friendly to the user.

CODE IMPROVEMENTS

Interfacing with Octave and Python

Since its early days, the PLACET tracking capabilities have been accessed via a simple interface based on the Tcl/Tk scripting language. In recent years an interface to the Octave language for numerical computations was added to PLACET. The Octave extension was well received by the PLACET user community, because it simplified and significantly extended the PLACET instruction set with highly specialized numerical tools. On the other hand the internals of the implementation, that consisted of a full embedding of the Octave interpreter into the PLACET core itself, showed limitations over time, when Octave evolved and some changes in its API (Application Programming Interface) broke the PLACET-Octave maintainability.

To overcome this limitation, the interface between PLACET and Octave has completely been rewritten from scratch, this time making use of SWIG (Simplified Wrapper and Interface Generator) [2], a software development tool for building scripting language interfaces to C and C++ simulation codes, originally developed at the Los Alamos National Laboratories (now open-source). SWIG is a relatively big project, dynamic, and with an active user community. Choosing SWIG immediately gave several advantages, two of which are reported as an example:

- The maintainability of the interface and its portability to the latest version of Octave are provided by the SWIG developers team, with no efforts from us.
- SWIG allowed to create another useful interface: between PLACET and the Python language, which is widely used by the scientific community world-wide.

As a result from this effort, PLACET can now be interfaced both with Octave and Python, directly and simultaneously. Any script can include hybrid commands from both languages, embedded into the Tcl/Tk surrounding environment, in a very natural and convenient way:

```
# Define beamline, particle distribution
    etc..
Octave {
 R = placet_get_response_matrix("linac",
     "beam0", Bpms, Correctors);
 S = svd(R);
}

TestNoCorrection -beam beam0 -emitt_file
    emitt.dat

Python {
E = numpy.loadtxt('emitt.dat')
}
```

where PLACET: computes the response matrix R of the beamline 'linac' using the beam 'beam0'; extracts its singular values of R with Octave; tracks the 'beam0' through the active beam line, and then calls Python to post-process the emittance file.

Parallel Computing

OpenMP Deep parallel designs are often challenging to implement into existing codes, which are normally not written with future parallelization in mind. In these cases, OpenMP [3] can provide benefits of parallel computing to some extent on a single computer with multiple threads.

OpenMP is mostly written using preprocessor declarations, which means that the code can be compiled excluding these additions and work as a single threaded code. An example can be found in the new IR Tracking Module described later:

ISBN 978-3-95450-122-9

```
#pragma omp parallel for
for (int i=0; i<beam->slices; i++) {
    // loop over the particles
    part_step_particle(beamline,i,index);
}
```

The pragma statement, along with an extra compile flag is everything that is needed to execute the for loop in parallel, assuming the loop is thread safe.

OpenMP is utilizing shared memory, which is highly beneficial for code that need to exchange a lot of data between the processes during parallel execution. A good example in beam dynamics is simulation of collective effects, where the full particle distribution needs to be broadcasted to all threads before a collective effect can be calculated.

The main limitation of OpenMP is that one is limited to one machine, so a realistic maximum gain is typically around one order of magnitude. Another challenge can be unforeseen problems such as in the example above. This for loop is ran multiple times, and each time each thread will get an unpredictable selection of particles to track. The result is that the threads must exchange a lot of data, slowing down the simulation on multiple CPU machines. Such problems can be overcome, but are not always easy to spot when a lot of the parallelization is happening "behind the curtains".

OpenMP has been implemented in some parts of PLACET already, namely tracking loops of SBENDS when synchrotron radiation is inactive, and the IR Tracking module. These additions have been very minimal in terms of changes in the code. A speedup in specific simulations of a factor 3 to 6 have been obtained.

Distributed Tracking using MPI

A module to allow tracking to be performed over distributed systems has been recently added to PLACET using the MPI protocol [4]. MPI is a tool designed for high performance on both massively parallel machines and on workstation clusters. Within PLACET, MPI can be used to automatically distribute the computational load of tracking over multiple nodes of a cluster, then gathering the results afterwards. This approach can speed up the tracking by several factors. It must be noted that MPI can be used only in regions of the machine where collective effects can be neglected, as their implementation requires dedicated research which is still matter of study.

To access this new functionality, three new keywords have been added to PLACET. Two lattice elements: `MPI_Begin`, and `MPI_End`, which allows to enable / disable the MPI tracking in a specific parts of the lattice, and a new command: `MPI_TestNoCorrection` that performs the tracking. An example of PLACET MPI usage is the reported:

```
# Define beamline
BeamlineNew
Girder
MPI_Begin
 Drift -length 1
 Quadrupole -length 1 -strength -0.123
 Drift -length 1
 Quadrupole -length 1 -strength 0.123
MPI_End
BeamlineSet -name test

# Track the beam using MPI
MPI_TestNoCorrection -beam beam0 -survey
    None -mpirun "openmpirun -np 4"
```

The first part defines a beamline that can be tracked in parallel. The second part performs the tracking, using the OpenMPI implementation of MPI. In this case the computational load is distributed among 4 CPU's.

NEW FUNCTIONALITIES

IR Tracking Module

For the purpose of studying the impact from external fields on the beamline, we have developed what we call an IR Tracking Module. This module reads in an external field map from a file in addition to the beamline. The purpose for creating this module was to track the beam through the final focus system (FFS) in CLIC, including a detector solenoid field map. However, the code is fairly general, and can be used to track any field map for any parts of a beamline.

The integration method is a 4th order symplectic integrator [5]. The integration step can be set in the track command, by default set to 1 mm. Integration steps down to 1 μm have been checked and it was found that the error is sufficiently small for interaction point distributions with a step size of a few millimetres.

The input format for the field map is an ASCII table with the columns x [m], y [m], z [m], B_x [T], B_y [T], B_z [T]. A linear interpolation is done between the listed values in the table.

The tracking is at the moment implemented with a separate tracking command in TCL:

```
# Define beamline, particle distribution
    etc..
...
# Tracks the beam beam1 through active
    beamline, including the field map from
     fieldmap.txt in the calculation
TestIntRegion -beam beam1 \
 -emitt_file emitt.dat -synrad 1 \
 -angle 0.001 -step 0.005 -backward 0 \
 -writefirst 0 -filename fieldmap.txt
```

Worth noting in this example is the angle, step, writefirst, filename, and backward argument. Other arguments are available and behave the same way as for other tracking commands. All arguments are documented, and the documentation can be invoked with the command "Help TestIntRegion". The module and studies using this module will be described in more details in [6].

ISBN 978-3-95450-122-9

Cavity BPM and Short Range Wakefields

Cavity BPMs are used in several accelerator facilities and are planned to be used in the future linear collider. Excellent position resolution, down to tens of nanometers has been achieved, but the short-range geometric wakefields are a concern [7], especially for small beam emittances. In order to study these wakefields a new beamline element has been introduced in PLACET, `CavityBpm`, which is a `Bpm` with realistic short range and long range geometric wakefields. The long range wakefield description is similar to what was already existing for other `Elements`. There are two ways to input the short range wakefield, which needs to be calculated externally, e.g. with an electromagnetic field simulator like GdfidL [8], see figure 1 for an example:

- a list of kicks, with units $[\mu\mathrm{rad\ GeV}/\mu\mathrm{m}]$
- a spline which describes the wakefield, with units $[\mathrm{V}/\mathrm{mm}/\mathrm{pC}]$ vs $[\mathrm{m}]$, see figure 2 for an example.

Since the wakefield is dependent on the bunch length, the wakefield description needs to match the bunch shape. For this reason a second method is available that approximates the full bunch wakefield from a wakefield from a short Gaussian bunch.

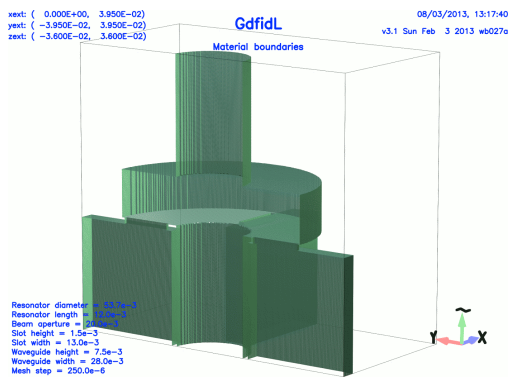

Figure 1: Geometry of the cavity BPM used at ATF.

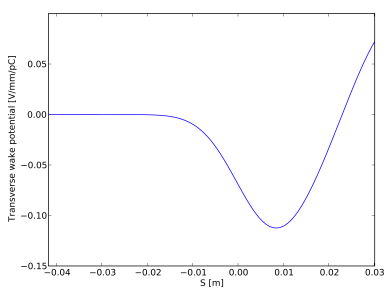

Figure 2: Short range geometric wake field of the cavity BPM used at ATF for a Gaussian shaped bunch length with standard deviation of 7mm.

The short range wakefield description is planned to be implemented in a more general way for every beamline `Element` very soon. 1cm

SIMULATION FRAMEWORKS

ATF Framework

The Accelerator Test Facility (ATF2) at KEK provides a test facility for the FFS of the future linear collider. In addition, beam instrumentation devices are deployed and developed. To study the various experiments done in the ATF2 extraction and FFS line, a simulation framework was setup. The framework simulates pulse-to-pulse and includes amongst others all possible dynamic imperfections including ground motion and multi-bunch simulation. For realistic comparison studies with the accelerator all accelerator magnet strength and position settings can be read in from the official settings files.

FACET Framework

The FACET user facility at SLAC is a unique R&D facility for experimental beam physics using the SLAC linac. PLACET has been used in the SLAC Main Control Center, to simulate and analyze the data of beam-based alignment techniques applied to the SLAC linac, as a flight simulator. A very good agreement between the simulations and the real-life data acquired has been seen, being able to predict the result of the orbit correction with excellent precision [9].

CONCLUSIONS

PLACET is evolving and new features are being added in order to cope with the simulation needs of the linac collider R&D community. New functionalities have been added: such as tracking in the interaction region with detector solenoid field map and a new beamline element, CavityBpm which takes into account realistic short and long geometric wakefields. To compute with the increasing computational need effective parallel solutions based on openMP / MPI have been put in place in the code, in a manner that is transparent to the users. The reliability of the code is being checked and verified more and more often in experimental tests, like those at ATF2 and FACET, where PLACET has proved to be a reliable and extremely useful tool.

REFERENCES

[1] `https://savannah.cern.ch/projects/placet`

[2] `http://www.swig.org`

[3] `http://openmp.org`

[4] `http://www.mcs.anl.gov/research/projects/mpi`

[5] Herr, W., *"Numerical and Computational Tools in Accelerator Physics"* `http://cern.ch/zwe/METHODS`

[6] Inntjore Levinsen, Y., Dalena, B., and Tomas, R., *"Integrated Symplectic Tracking of Solenoid Field Maps in PLACET"*, *to be published.*

[7] Snuverink, J. et al. "Short Range Wakefield Measurements of High Resolution RF Cavity Beam Position Monitors at ATF2", MOPWA052, IPAC13.

[8] `http://www.gdfidl.de`

[9] Latina, A. et al. "Results of Beam-Based Alignment Tests at FACET", *to be published.*

ISBN 978-3-95450-122-9