mbtrack2, A COLLECTIVE EFFECT LIBRARY IN PYTHON

A. Gamelin∗, W. Foosang, R. Nagaoka, Synchrotron SOLEIL, Gif-sur-Yvette, France

Abstract

This article introduces mbtrack2, a collective effect library written in python3. The idea behind mbtrack2 is to build a coherent object-oriented framework to work on collective effects in synchrotrons. mbtrack2 is composed of different modules allowing to easily write scripts for single bunch or multi-bunch tracking using MPI parallelization in a transparent way. The base of the tracking model of mbtrack2 is inspired by mbtrack, a C multi-bunch tracking code initially developed at SOLEIL [1]. In addition, many tools to prepare or analyse tracking simulations are included.

A COLLECTIVE EFFECTS LIBRARY

The collective effects library in mbtrack2 has been designed to be a successor to mbtrack, a successful C code for multi-bunch particle tracking [1]. As mbtrack, it allows to track hundreds of bunches with thousands of macro-particles in each bunch and integrate both single and multi-bunch collective effects.

The first difference is that, while mbtrack was built for a fast execution time using a close-to-machine language like C, mbtrack2 uses python, a higher level language, which provides a slower execution time but a shorter development time, a scriptable approach and an easier access to the simulation data.

mbtrack2 is written following an object-oriented programming paradigm which allows to build flexible tracking scripts using the different modules. Internally, the code uses well optimised python standard modules, like numpy or scipy, to allow for relatively fast tracking compared to pure python. When used in multi-bunch mode, each bunch can be treated in a different core using the python implementation of the MPI standard, mpi4py [2]. Both modes, single or multi-bunch, can be used interactively to have an easy access to the simulation data.

Finally, mbtrack2 is open source and distributed with a BSD 3-Clause license. It is available on SOLEIL gitlab instance [3]. In the spirit of open source software, mbtrack2 has also been made for others to be able to use it, so the code is well documented and reference articles are cited on the source code when necessary (for example to explain which formula is used and where it comes from).

Impedance and Wake Functions

In mbtrack2, each impedance and wake function are represented as python objects. These objects contain all the necessary information to define them physically, like the frequency points, the impedance data, the type of the impedance (dipolar, quadrupolar, ...) and its corresponding direction.

These impedance or wake function objects can be created using the usual analytic formulas or directly loaded from other codes like CST [4] or 1W2D [5].

These impedance or wake function objects can be combined with a bunch profile to compute a wake potential used in the tracking, but also for many other calculations. A method allows to calculate the loss (or kick) factor in both time domain and frequency domain. Several methods allow to go from wake function to impedance using FFT and inversely from impedance to wake function.

Combining these methods, we also provide a way to compute a pseudo wake function $W_∥$ from simulated wake potential data, usually obtained from external sources like CST [4], by using the deconvolution method [6]:

$$W_∥(z) = FT^{-1} \left[ \frac{FT[W_p(z)]}{\check{\rho}(\omega)} \right] = FT^{-1}[\check{Z}_p(\omega)], \quad (1)$$

where $FT$ and $FT^{-1}$ denote respectively the Fourier transform and the inverse Fourier transform. $W_p(z)$ is the longitudinal wake potential, while $\check{\rho}(\omega)$ is the Fourier transform of the source bunch profile, usually considered to be Gaussian.

The accuracy of the pseudo wake function in Eq. (1) can be verified by performing a convolution on it with the same source bunch profile, the original wake potential data should be reconstructed. In Fig. 1, the wake potential of a beam position monitor (BPM) in SOLEIL storage ring calculated using CST is presented in comparison with the reconstructed one obtained from the deconvolution method in mbtrack2. It can be seen that, in that case, the reconstructed wake potential can be as precise as the original one. Of course, the validity of the pseudo wake function is limited to the frequency range of $\check{\rho}(\omega)$ or, equivalently, the frequency range of the computed $\check{Z}_p(\omega)$.

![Figure 1: Wake potential of a BPM in SOLEIL storage ring from a CST simulation in solid blue line. Convolution of the wake function $W_∥$ obtained by the deconvolution method with the source bunch profile $\rho$ in dashed orange line. The rescaled source bunch profile $\rho$ is plotted in the solid green line.](image)

MC5: Beam Dynamics and EM Fields

D11: Code Developments and Simulation Techniques
Several impedances and wake functions can then be grouped together in a wakefield element corresponding to a given physical object in the accelerator producing both longitudinal and transverse wakes. These wakefield elements can then be assembled into an impedance model object which takes into account the ring optics.

Other Tools

Using the same framework as for the tracking code, many small tools were added to the collective effect library. For example, a module provides functions to compute instability thresholds for various instabilities using well known analytic formulas. Other modules provide both impedances and/or wake functions for various elements (like the resistive wall, resonator model, tapers, ...) using different models from the literature.

An analytical method to obtain the equilibrium bunch density distribution with an arbitrary number of active or passive RF cavities was also added [7]. This method is restricted to uniform filling and is mostly used to study the impact of active or passive harmonic cavities. We found a very good agreement between the results using this method and the tracking using mbtrack2 [7].

TRACKING CODE

The tracking part of the code is based on two types of objects:

- A Bunch class which stores the 6D coordinates of each macro-particle using a numpy array.
- A Beam class which stores each of the different bunch objects and handles the MPI parallelization using mpi4py [2].

Using this architecture, the embarrassingly parallel elements of the code (i.e. when there is no interaction from bunch to bunch) are written for the bunch class for simplicity and extended behind the hood for the beam class.

Tracking Elements

Up to now, the tracking code mostly uses a one turn map approach. The basic equations for tracking are divided in three elements for the longitudinal map, the transverse map and the effect of synchrotron radiation and quantum excitation. The equations used for the longitudinal and transverse maps are identical to the ones used in mbtrack as described in [8].

The synchrotron radiation treatment is, however, slightly different from mbtrack as it directly uses a fixed (input quantity) damping time and not a dynamic damping time computed from the losses per turn. This approach is similar to one used in PyHEATDAIM [9].

Standard elements like a perfect RF cavity (simple sine wave model) or different types of physical apertures are also provided.

Wake Potential

The single bunch collective effects can be introduced by using the WakePotential class. This class can compute a wake potential (for example a monopolar wake for the longitudinal direction, or a dipolar/quadrupolar wake for the transverse directions) from any uniformly sampled wake function via the WakeFunction class.

First, the bunch macro-particles are sorted into bins using a variable grid to compute the bunch charge density profile and its dipole moment. Then the bunch profile is interpolated on the wake function time base which is used to perform the convolution to get the wake potential.

The resulting wake potential is shown in Fig. 2 for the case of the dipolar resistive wall wake function for a circular beam pipe. The bunch has been shifted by 50 µm in the horizontal direction to have a quasi uniform dipolar moment and by 25 ps in the longitudinal direction in order to show the interpolation better.

If both the wake function and its corresponding impedance are provided to the WakePotential class, a method can compare the loss/kick factors computed in time domain using the computed wake with the frequency domain computation for a Gaussian bunch to check the accuracy of the wake potential, see Table 1.

Table 1: Kick Factor Computed in Time Domain by Numerical Integration Using the Dipolar Wake Potential Shown in Fig. 2 and in Frequency Domain for a Gaussian Bunch

<table>
<thead>
<tr>
<th>k_x,dip [V/C/m]</th>
<th>Time Domain</th>
<th>Frequency Domain</th>
<th>Relative Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4.064 285×10^{16}</td>
<td>4.063 953×10^{16}</td>
<td>0.008178 %</td>
</tr>
</tbody>
</table>

Cavity Resonator

The case of the long range interactions, multi-bunch and multi-turn, introduced by high Q resonators can be dealt with
using the CavityResonator class. This type of object can be
used to model active or passive RF cavities of any harmonic
in a fully self consistent way. For active cavities, the cavity
voltage is the sum of the beam loading voltage and of the
generator voltage. Similarly, this class can also be used to
investigate instabilities driven by cavity high order modes
(HOM). The implementation of this class is very similar to
what has been recently added in mbtrack [10].

Monitors

There are several built-in data recording classes in
mbtrack2 depending on the type of the data to save. For
example, a BunchMonitor can record information about a
given bunch such as its current, the emittances, the mean
values and the standard deviation of the 6D coordinates, etc.
The recording frequency is adjustable depending on the user.
The output is written in an HDF5 format using h5py pack-
age [11]. This format provides an organized binary file and
the data from the different monitors is written in a single file
corresponding to a given simulation run.

Apart from this BunchMonitor class, there are many other
monitors to save tracking information about the beam, the
bunch profiles, the wake potentials, the tunes, etc. This type
of feature is inspired from PyHEADTAIL [9], which has been
a great inspiration in general for the structure of the tracking
part of this code.

Every monitor is then associated with a dedicated plotting
routine that offers several options to visualize the data. The
HDF5 file can also be opened independently to access the
raw numerical data.

BENCHMARKING

To verify the accuracy of mbtrack2, its simulation re-
results are compared with mbtrack [1] and MOSES [12]. The
simulation is set up as follow: a single bunch composed
of 10⁵ macro-particles is submitted to a resonator model
impedance [13]. The bunch current is scanned and for each
current step the bunch is tracked for approximately 2-3 damp-
ing times.

Micro Wave Instability

Figure 3 shows the relative energy spread of the bunch
with respect to the bunch current obtained from mbtrack2
and mbtrack. Each point is the average value over one
current step, which is 1 mA, and the uncertainty along the
vertical axis is the standard deviation. From the plot, an
increase of the energy spread, which is a sign of the Micro
Wave Instability (MWI), can be seen at between 14-15 mA.
A good agreement between the two codes can clearly be seen
within the scan resolution limit.

Transverse Mode Coupling Instability

The Transverse Mode Coupling Instability (TMCI) thresh-
hold current obtained from mbtrack2, compared with
mbtrack and MOSES, is presented in Table 2. mbtrack2
yielded slightly higher threshold than the other two codes,
yet they were comparable considering that the largest differ-
ce between these values was 8.9 %.

Table 2: TMCI Thresholds. Resonator parameters:
\( R_s = 1 \, \text{kΩ}, f = 10 \, \text{GHz}, Q = 0.7 \).

<table>
<thead>
<tr>
<th></th>
<th>mbtrack2</th>
<th>mbtrack</th>
<th>MOSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current [mA]</td>
<td>8.6</td>
<td>8.4</td>
<td>7.9</td>
</tr>
</tbody>
</table>

NEXT STEPS

As mbtrack2 is a work in progress, there are many ele-
ments that we would like to add in the future. An important
feature which is still missing in mbtrack2 is the addition of
the general case of multi-bunch and multi-turn wakefields
(which is available in mbtrack for resistive wall). Another
future development should be to add longitudinal and trans-
verse bunch by bunch feedback systems.

ACKNOWLEDGEMENTS

The authors would like to thank N. Yamamoto for the
initial development of the CavityResonator functions in
mbtrack and his help in the transcription of this part in
mbtrack2. The authors also thanks D. Amorim for sharing
his code which we used as a basis for the impedance and
wakefield classes.

REFERENCES

tive Effects in SOLEIL and Diamond Using the Multiparticle
Tracking Codes SBTRACK and MBTRACK”, in Proc. 23rd
Particle Accelerator Conf. (PAC’09), Vancouver, Canada,

eldistributed computing using python”, Advances in Wa-

collectiveeffects/mbtrack2

Figure 3: The energy spread as a function of bunch current
obtained from mbtrack2 compared to mbtrack. Resonator
parameters: \( R_s = 1 \, \text{kΩ}, f = 10 \, \text{GHz}, Q = 0.7 \).


