RECENT DEVELOPMENTS FOR THE HEADTAIL CODE: UPDATING AND BENCHMARKS

D. Quatraro, G. Rumolo, B. Salvant
European Organization for Nuclear Research (CERN),
CH-1211 Genève 23, Switzerland

Abstract

The HEADTAIL code models the evolution of a single bunch interacting with a localized impedance source or an electron cloud, optionally including space charge.

The newest version of HEADTAIL relies on a more detailed optical model of the machine taken from MAD-X and is more flexible in handling and distributing the interaction and observation points along the simulated machine. In addition, the option of the interaction with the wake field of specific accelerator components has been added, such that the user can choose to load dipolar and quadrupolar components of the wake from the impedance database Z-BASE. The case of a single LHC-type bunch interacting with the realistic distribution of the kicker wake fields inside the SPS has been successfully compared with a single integrated beta-weighted kick per turn. The current version of the code also contains a new module for the longitudinal dynamics to calculate the evolution of a bunch inside an accelerating bucket.

INTRODUCTION

The HEADTAIL code [1] was originally developed in 2000 with the goal of studying the interaction between a proton or positron bunch and an uniformly distributed electron cloud with fixed density. The code was later extended to include the interaction of a single bunch with a localized source of impedance (resonator or resistive wall) using the wake field approach. Where possible, the code was successfully benchmarked against analytical solutions, other codes or experimental results [2].

The HEADTAIL model of high intensity effects was based on applying to each of the bunch macroparticles an integrated kick (from electron cloud, space charge or wake field) in one or more points uniformly distributed around a machine. The macroparticles would then be transported with the 1-turn (or fraction of turn) transport matrix between these points. To have a better modeling of space charge as well as to be able to introduce localized kicks from wake fields of known structures, we decided to develop a more detailed description of the lattice of the machine. Therefore, a sector map type transport around the accelerator has been introduced, as explained in Section 2. The new machine description is based on a newly developed link between HEADTAIL and MAD-X [3]. The transport matrices are generated running MAD-X from inside HEADTAIL. The syntax and criteria that define the selection and the meaning of the lattice stations at which the high intensity effects are applied to the beam (and between which the beam is optically transported) are discussed in detail in the following sections. Besides, the new option to load the dipolar and quadrupolar components of the wake fields directly from the impedance database Z-BASE [4] is described.

THE SECTOR MAPS AND THE CHROMATICITY MODEL

In this section we explain how transport and the chromaticity are treated in the code. As we can see further, each of the lattice stations can be either an interaction point (space charge, electron cloud or wake field interaction) or an observation point (see next section).

HEADTAIL first gets tune and chromaticity values from the standard `input.cfg` file and then runs MAD-X to generate the lattice, matching those values. Once the twiss file of the machine is produced, some lattice stations are selected (Fig. 1), and their twiss parameters, phase advances and chromaticities are written in a separated file in order to build the transport matrices across these points.

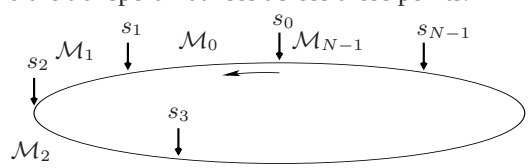

Figure 1: Arrangement of the several elements along the ring. s_j is the starting point for the transport through the $\mathcal{M}_j$ matrix.

The following transformation is applied to each particle phase space coordinate $\mathbf{x} = (x, x', y, y')$

$$\mathbf{x}\left(s_{j+1}\right) = \mathcal{M}\left(s_{j+1}|s_j\right) \cdot \mathbf{x}\left(s_j\right) = \mathcal{M}_j \cdot \mathbf{x}\left(s_j\right) \qquad (1)$$

where the 4×4 matrix $\mathcal{M}$, containing the twiss parameters, is given by

$$\mathcal{M}\left(s_{j+1}|s_j\right) = \begin{pmatrix} \mathbf{M}_x\left(s_{j+1}|s_j\right) & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_y\left(s_{j+1}|s_j\right) \end{pmatrix}, \qquad (2)$$

with $\mathbf{0}$ being the zero 2×2 matrix and $\mathbf{M}_i\left(s_{j+1}|s_j\right)$ the linear transport matrices between the station j and $j+1$. No linear coupling has been taken into account for the transport. Chromaticity is taken into account through the momentum dependent chromatic shift of the phase advance, while the change of the beta functions due to the different momentum is neglected. The transport matrix for the generic off-momentum particle will be constructed according to:

$$\mathcal{M}_j^{Chr.}\left(s_{j+1}|s_j\right) = \mathcal{M}\left(s_{j+1}|s_j\right)\mathcal{M}\left(s_j|s_j;\Delta\psi_{j+1,j}\right) \qquad (3)$$

being $\Delta\psi_{j+1,j} = \delta\xi_{j+1,j}$ and $4\pi \cdot \xi_{j+1,j} = \int_{s_j}^{s_{j+1}} ds\,[k(s) - m(s)\,D(s)]\,\beta(s)$. The code reads the values of the β, α, ψ, ξ functions for both planes from the MAD-X output and builds the transfer matrices Eq. (2) or Eq. (3) particle by particle depending on whether the user has decided to take into account the chromatic effects or not. The values $\xi_{j+1,j}$ are the variables called `DMUX` and `DMUY` in the MAD-X output.

USE OF THE HEADTAIL

The `hdtl` HEADTAIL executable needs the following input files to run.

i) `input.cfg` In this file the bunch parameters as well as some of the machine characteristics are written. The twiss parameters are directly taken from from standard output of MAD-X, which also matches both the tunes and the chromaticities to the values given in the `input.cfg` file.

ii) `lattice.dax` This is the MAD-X input file. This file must contains a call to the file `matchtune_hdtl.cmdx` (see next point) in order to perform the matchings.

iii) `matchtune_hdtl.cmdx` This is the matching module.

iv) `element.Jdip.dat, element.Jqua.dat` In this version of the code there is also the possibility to take into account the tabulated wake field functions (dipolar and quadrupolar components) for each selected element. This was, for example, the case of the wake field functions for the SPS kickers. Once we have gained these information `hdtl` can use them to simulate the bunch dynamics.

Once that MAD-X has finished to run, the file `machinelattice.txt` is produced, containing all the basic information for the linear transport.

There are several output files in addition to those written in the former version of the code. These new files mainly concern the structure of the ring.

i) `TWISS.dat` This file contains the twiss parameters of the whole machine for each element.

ii) `selectedmachinelattice.txt` This file contains the twiss parameters and the name of the elements selected by the user and that the code will use to track the particles.

iii) `Wake_resume.dat` This file is written to summarize the name, the component and the values of the used wake fields at each element.

iv) `lengths.dat` This file is written in order to check each of the length used to apply both the space charge and the wake field forces.

v) `ELEMENTS_NAMES.txt` This file is written in order to distinguish among the possible actions happening at each selected lattice station.

The choice of the interaction points throughout the machine depends on the command line arguments. In the `hdtl` syntax there are interaction and observation points. The former can be space charge, wake field and electron cloud interaction points and the latter are locations along the machine where the information about the centroid position and the bunch length are stored.

i) **Space charge forces** There are three options to select these points. The first option is to uniformly sample the β_x function, and so the bunch transversal sizes $\sigma_x \approx \sqrt{\beta_x \epsilon_x}$, in the whole range of its variation for growing values along the machine length. The second option applies the same criterion in the y plane. The third option is to randomly sample the β function along the machine.

ii) **Observation point, electron cloud and wake field interaction points** The code can take into account electron cloud and wake field interaction forces as well as interaction points through the `ELEMENTS_NAMES.txt` file. To each chosen observation point its output file is written, containing the centroid position and the length of the bunch. All this information is contained in the file called `bpmname.dat`, `bpmname` being the name of the observation point element.

The transfer matrices are built in the `hdtl` code.

THE SYNTAX OF THE CODE

Once all the needed files have been obtained all the options and the names of the several elements are given just typing on the shell.

Table 1: Outline of the arguments command line for `hdtl`. argv$j$ stands for the $j-$th argument to type on the shell. The number of the arguments depends on how many elements we want to use.

arg	option	type	description	comment
1	-	`*char`	MAD-X input file with lattice structure	should write the file `machinelattice.txt`
2	0	`int`	sampling through all β_x range of variation	-
	1	`int`	sampling through all β_y range of variation	-
	2	`int`	random choice of both β_x and β_y	-
	3	`int`	no space charge forces	-
3	-	`int`	# of space charge interactions per turn	compulsory if argv2$= 0, 1, 2$
>4	-	`*char`	elements' name	keywords `ec wf ob`

In Tab. 1 the several command line inputs are listed and explained in more details. Concerning the last arguments, as already mentioned, the syntax rule is the following: the elements belonging to one particular class (electron cloud, wake field and observation points) are those whose names contain one of the string following the respective keyword (`ec wf ob`) up to the next one or up to the end of the typed commands. All these names are read from the MAD-X standard output. Example: the command `hdtl sps.dax 2 200 ec MBB wf MKV ob BPV` runs the MAD-X file `sps.dax` in order to produce the normal output file `machinelattice.txt` randomly selecting 200 points for the space charge interaction. Therefore, the elements whose name contains the string `MBB` are taken into account as electron cloud interaction points, as well as the elements whose name contains the string `MKV` are taken into account as wake field interaction points and the observation points are those whose names contain the string `BPV`. It is worth mentioning the fact that the code can recognise all those elements whose name contains that character string. In Fig. 2 we can look at the position of the several elements for about one sextant of the SPS ring.

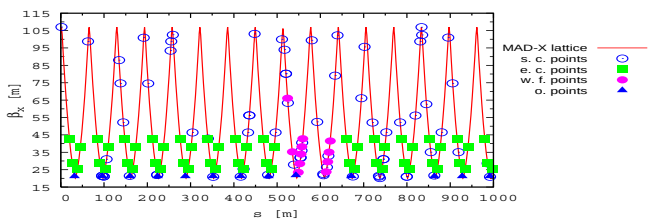

Figure 2: Example of the sampling along the machine. The several type of elements are shown. The command to generate that lattice was `hdtl sps.dax 2 500 ec MBB wf MKV ob BPV`

RESULTS FOR TMCI AT THE SPS

As already mentioned above we have benchmarked the latest `hdtl` version with the previous one, studying the Transverse Mode Coupling Instability (TMCI) in the SPS ring. Performing the benchmark we took into consideration only the kickers as sources of impedances. The former version of the code uses the one-kick approximation for the interaction of the wake fields with the bunch. The wake function $W_\perp(z)$ of each kicker is weighted using the local β function, so that the total kick is obtained from

$$\hat{W}^i_\perp(z) = \frac{\sum_{j=0}^{N} \beta^i_j \cdot W^i_{\perp,j}}{\bar{\beta}} \qquad i = x, y \qquad (4)$$

where N is the total number of kickers. The latest version of the code uses the following model for each impedance source

$$\int_{s_j}^{s_j+\Delta s_j} ds\, f_n(n) = \kappa \left({}^iW_n^{Dip.} \hat{n} + {}^iW_n^{Quad.} n \right) \qquad (5)$$

where $\kappa = e^2/N$, $\hat{\cdot}$ means averaging over the spatial coordinate and ${}^iW_n^{Dip.}$, ${}^iW_n^{Quad.}$ are respectively the dipolar and the quadrupolar component of the wake field at the i-th element with $n = x, y$. In Fig 3, we show the growth rate versus the bunch intensity.

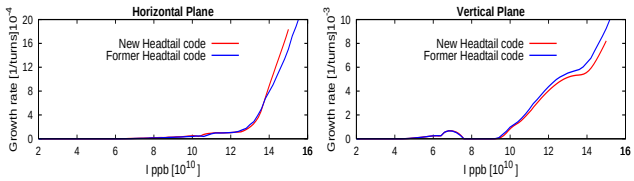

Figure 3: Growth rate comparison between the one kick approximation and latest model approach. The agreement between the two curves is pretty good.

In addition, in the context of the localization of the vertical impedance sources in the SPS using beam intensity based techniques, we used the latest version of the code [6]. Simulations were performed to better understand the effectiveness of this technique and the imposed constrains.

ACCELERATING BUCKET

A major improvement in the longitudinal dynamics modeled by HEADTAIL concerns the extension to track bunches during acceleration. A new line in the input line is used to specify the required accelerating rate and new options for the flag for longitudinal motion have been added to simulate an accelerating bucket in simple or double harmonic, or optionally including higher terms in η (for simulating bunches crossing transition). The model has been applied to do studies of transition crossing at the PS [7].

CONCLUSIONS

We have presented a new version of the HEADTAIL code with its new options. We have benchmarked the results for the case of the SPS TMCI. The agreement between the former one-kick approximation and the current version is excellent.

In addition the BPM data analysis has been carried out in order to localize the sources of impedances (for the SPS kickers).

REFERENCES

[1] G Rumolo and F. Zimmermann, SL-Note-2002-036-AP; Geneva: CERN, 2002

[2] G. Rumolo and E. Métral, *Simulations of single bunch collective effects using HEADTAIL*, in Proc. of ICAP06

[3] http://mad.web.cern.ch/mad/

[4] O. Brüning, CERN-SL-96-069-AP; Geneva: CERN, 1996

[5] E. Métral at al. , *CERN SPS Impedance in 2007* CERN-AB-2008-008; Geneva: CERN, 2008

[6] G. Arduini at al. , *Transverse impedance localization using intensity dependet optics*, this conference.

[7] S. Aumon, *Beam Instability Studies at Transition Crossing in the CERN PS*, this conference.