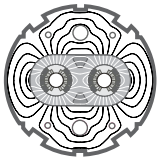

LHC Project Note 344

May 25, 2004
Werner.Herr@cern.ch

Particle tracking with MAD-X including LHC beam-beam interactions

W. Herr, AB Division, CERN

Keywords: beam-beam, tracking, MAD-X

Summary

Particle tracking with beam-beam interactions is necessary to study the stability of the LHC beams for different operational conditions. The MAD-X program allows the introduction of beam-beam elements into the lattice and a realistic simulation. To allow an efficient evaluation of different scenarios and problems, a set of tools has been developped, based on the MAD-X program. The techniques necessary for flexible and advanced tracking studies including LHC beam-beam interactions are described in this note. Alternatively, the described techniques can be used to provide the necessary input for massive tracking studies with SIXTRACK.

1 Introduction

Tracking studies are the usual tool to investigate the stability of a beam, in particular in the presence of non-linear fields. Such tracking studies were extensively used to evaluate the dynamic aperture of the LHC [1] at injection and collision energies. A very important non-linearity in the LHC is the beam-beam interaction and to evaluate the long term stability, it must be taken into account. Tracking studies with beam-beam interactions are used for several reasons:

- Determine the detuning (footprints)
- Evaluate the dynamic aperture in the presence of beam-beam effects
- Study coherent beam-beam effects

While the last effect usually requires special programs (e.g. BEAMX [2]), the first two can be studied [3, 4, 5] with the MAD-X program [6] or dedicated tracking programs such as SIXTRACK and its run environment [7, 8, 9]).

In this note I shall describe which input is necessary and how it is prepared, in particular for the more advanced tracking of selected bunches.

2 Tracking with LHC beam-beam effects

In general, the beam-beam interactions in a tracking program are represented by a series of localized, non-linear kicks, computed from the particle's coordinates using the lattice and beam parameters. These interactions can be either head-on or parasitic, so-called long-range interactions.

2.1 Bunch to bunch differences

The large number of bunches in the LHC add significant complications to this procedures, in particular due to the crossing angle and so-called PACMAN bunches [10]. Due to the LHC bunch filling schemes [11, 12, 13] the different bunches can experience very significantly different beam-beam effects and this should be taken into account for all tracking studies. In previous studies only the nominal bunches, i.e. with the full collisions scheme, were simulated. Naively one might expect that these bunches, which experience the maximum integrated beam-beam effects, are the most unstable. However, the other bunches will have different tunes and orbits and occupy different parts in the tune diagram and usually the machine is optimized for the nominal bunches which, for standard filling schemes, are the mayority of the bunches [11]. For a given bunch (i.e. at a specified bucket or position number) the correct collision scheme at the appropriate longitudinal positions in the ring must be applied. For example the computation of tune footprints requires the insertion of 120 long range and 4 head-on beam-beam elements [4]. Since the optical parameters at these positions are in general different from each other, a more automated procedure is highly useful.

A set of utilities has been produced to simplify the setting up of the beam-beam tracking in MAD-X which will be described in the following.

2.2 Head-on collisions

In the experimental collision points of the LHC, the bunches collide not exactly head-on, but at a small crossing angle, either in the horizontal or vertical plane [14]. In order to simulate this crossing angle, the bunches are sliced into several parts with equal intensity. In the standard setup, 5 slices per bunch are used. This breaks the symmetry between the two planes and couples the longitudinal and transverse planes.

2.3 Parameter space

To allow a good evaluation of the effect of the beam-beam interactions on the beam stability, it must be possible to implement different scenarios which affect the strength and type of the beam-beam effects. Such important parameters are:

- Crossing schemes, i.e. horizontal versus vertical crossing planes and alternating crossing schemes.
- Bunch spacing, leading to a different number of long range interactions.
- Bunch filling schemes, which largely determine the collision schedules of the different bunches [11, 12, 13].

To efficiently study the different options and combinations between them, a flexible procedure is required where switches should be used wherever possible.

2.4 Self-consistent parameters

The different beam-beam effects for the different bunches [10] lead to slightly different closed orbits and in turn to a slightly changed beam separation which may lead to a slightly different beam-beam effects. The full treatment therefore requires the self-consistent calculation of the beam separation which is used in the beam-beam element. Usually this effect is considered a negligible perturbation and is ignored. Although this may be justified for most applications, I show how this information can easily be incorporated.

3 MAD-X setup

Particle tracking in MAD-X requires a complete description of the machine, i.e. the sequence of elements and their strengths. At the localized positions of the beam-beam interactions, a beam-beam element is inserted as a thin lens.

3.1 Preliminaries

Tracking with MAD-X requires the lattice in the form of thin lenses. The MAD-X command **makethin** can be used to convert a sequence with thick lenses. To setup beam-beam elements, it is required to define the properties of the **opposing** beam. Contrary to most other colliders the LHC has two separate rings and at a given longitudinal position the optical parameters for the two rings are in general not the same. It is therefore necessary to calculate the optics for both rings and to always use the appropriate values. This is true although any weak-strong tracking needs to be done only for one beam. Any strong-strong effects on the single particle behaviour (such as self-consistent beam separation) have to be implemented explicitly in the setup and are the subject of part of this note.

Most important is the correct computation of the crossing and separation schemes since they determine the geometry and therefore the strength of long range beam-beam interactions. They are usually pre-computed and can be switch on and off with scaling parameters (e.g. on_x1, on_sep1 etc.). Setting $on\_x1 = 1.0$ gives the nominal crossing angle in interaction point 1. This strategy also applies to the experimental spectrometer magnets in IP2 (ALICE) and IP8 (LHCb) together with their compensation magnets (on_alice,on_lhcb).

3.2 Beam-beam element in MAD-X

A general beam-beam element in MAD-X is defined as:

```
bb: beambeam, sigx=SIGMAX,sigy=SIGMAY,
              xma=X,yma=Y,charge=Q;
```

where $SIGMAX$ and $SIGMAY$ are the horizontal and vertical beam sizes of the opposing beam, Q is the particle's charge and X and Y are the horizontal and vertical displacements of the opposite beam with respect to its ideal orbit. Parameters important for the beam-beam effects, such as number of particles per bunch ($NPART$) and beam emittances EX and EY, and the ENERGY are specified in the **BEAM** command [6]. Other relevant parameters such as bunch length (SIGT, for head-on collisions with a crossing angle) and relative energy spread (SIGE) are specified also with the **BEAM** command.

This is sufficient to compute the kicks from head-on (i.e. in general: XMA = YMA = 0) and long range interactions when the beams are separated (in general: XMA and/or YMA $\neq 0$).

3.3 Calculation of parameters for beam-beam elements

The required parameters are usually taken from a previously calculated optics table, e.g. at a beam-beam position marked as *bbpos*1, and where needed from the parameters entered with the **BEAM** command [6]. For example:

```
bb: beambeam, sigx=sqrt(bbpos1->betx*beam->ex),
              sigy=sqrt(bbpos1->bety*beam->ey),
              xma=bbpos1->x,yma=bbpos1->y,charge=beam->charge;
```

The strategy is therefore:

1 Insert **beam-beam markers** at the position of head-on or long range beam-beam interactions.

2 Calculate the unperturbed optics with the markers in place.

3 Provide the **beam-beam elements** at the position of the markers.

This requires to run MAD-X at least once to prepare the input for the tracking study.

Considering the large number of collisions in the LHC (4 head-on and up to 120 long range collisions) this procedure must be simplified by the use of *macros* which are available in MAD-X. I use one particular feature to write formatted output into a file using macros. This output can be in the form of regular MAD-X input and therefore can be read in and executed by MAD-X, i.e. the input commands are prepared by MAD-X itself for later use. It is particularly useful that names and strings can be constructed from variables and loop indices. We can therefore prepare the full description of a beam-beam element, using the information from the computed optics and write it onto a file. This file can be used either in the same run or can be stored and read in again later on.

It has been mentioned before that some parameters must be taken from the opposing beam, such as beam size and intensity. Therefore they must be inserted in the beam-beam

elements correctly. If it is not obvious from the commands, (e.g. the USEed sequence), for the attributes of the **BEAM** command one can add the sequence name to the attributes, i.e. $beam->ex$ can be replaced by $beam\%lhcb2->ex$ to indicate that the attributes should be taken for the sequence lhcb2. In my macros I am not using this feature since it reduces the flexibility.

As a very simple example I show the macro definition:

```
instbb(nn, kk): macro = {
  print, text="install, element= bbipnnlkk, at=-kk*long_b, from=ipnn;";
  print, text="install, element= bbipnnrkk, at=+kk*long_b, from=ipnn;";
}
```

When it is called like:

```
k = 1;
n = 5;
while (k < 6)
{
  exec, instbb($n, $k );
  k = k + 1;
}
stop;
```

it will provide a list of install commands that can be used in MAD-X (here to install long-range beam-beam elements around interaction point 5, assuming $long\_b$ is half of the bunch spacing):

```
install, element= bbip5l1, at=-1*long_b, from=ip5;
install, element= bbip5r1, at=+1*long_b, from=ip5;
install, element= bbip5l2, at=-2*long_b, from=ip5;
install, element= bbip5r2, at=+2*long_b, from=ip5;
install, element= bbip5l3, at=-3*long_b, from=ip5;
install, element= bbip5r3, at=+3*long_b, from=ip5;
install, element= bbip5l4, at=-4*long_b, from=ip5;
install, element= bbip5r4, at=+4*long_b, from=ip5;
install, element= bbip5l5, at=-5*long_b, from=ip5;
install, element= bbip5r5, at=+5*long_b, from=ip5;
```

This type of macros is used extensively to set up all the beam-beam elements. Their behaviour can easily be controlled by switches and flags.

A typical macro defining a beam-beam element looks like:

```
bbele(nn,cc): macro = {
!--- macro defining parasitic beam-beam elements on left side of IP;
!--- nn = IP number, cc = cc-th parasitic encounter
print, text="bbipnnlcc: beambeam,";
print, text="          sigx=sqrt(ipnnlcc->betx*beam->ex),";
```

```
print, text="          sigy=sqrt(ipnnlcc->bety*beam->ey),";
print, text="          xma=ipnnlcc->x,yma=ipnnlcc->y,";
print, text="          charge:=on_lrnnl;";
}
```

resulting in e.g. (if called with bbele(1, 10):

```
bbip1l10: beambeam,
          sigx=sqrt(ip1l10->betx*beam->ex),
          sigy=sqrt(ip1l10->bety*beam->ey),
          xma=ip1l10->x,yma=ip1l10->y,
          charge:=on_lr1l;
```

The parameters can be taken from the optics calculation (like in the example above) or from another MAD-X input file. Of course, the markers (e.g. here: ip1l10) must be installed before the optics calculation. In this example, the charge on_lr1l could be used to switch on and off the interaction.

3.4 Different collision patterns

To provide sufficient flexibility, we have divided the preparation run into three separate parts:

1 Define beam-beam markers for all possible interactions.

2 Set up beam-beam elements for all possible interactions.

3 Run MAD-X with the TWISS command to calculate and fill the beam-beam elements using the optical functions calculated at the markers.

4 Install these beam-beam elements at the appropriate positions for a given bunch.

5 Store the completed elements or the whole sequence on a file.

This procedure has a non obvious advantage: when bunches are treated which have a collision pattern different from the nominal pattern (i.e. all possible interactions), one has to only modify the step 4, i.e. install only those beam-beam elements which are really needed. Furthermore, the results from steps 1 to 3 can be stored and re-used in later runs.

3.5 Definition of collision pattern

In general, every bunch follows its own collision schedule and specific pattern of head-on and long range interactions. A given pattern is therefore only valid for a specific bunch, (or bunches with the identical pattern).

In order to compute the correct pattern, a program $COLSCH$ was written which provides the necessary information. It computes various statistics on the collision pattern such as number of long range interactions, missing head-on collisions and many others. For all bunches the complete collision schedule is computed and if requested it is printed to a

file. This includes the list of all pairs of interactions. For a specified bunch the relevant information is made available in a format digestible by MAD-X. This information can be used to manipulate the installation of the beam-beam elements (step 4 in the procedure described above), and only the required beam-beam elements are installed.

For the computation, it is necessary to provide to *COLSCH*:

- Pattern of interaction points around the machine.
- Filling pattern of the bunches.

All information is provided in the form of two separate input files of the types shown below:

```
A file with the collision pattern (coll.in):

        1   2   -15  +15
      892   2   -15  +15
     3565   2   -15  +15
     6235   2   -15  +15

A file with the filling pattern (fill.in):

#Number of groups
 8
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     8 0     72 1     39 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     8 0     72 1     39 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     8 0     72 1     39 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     8 0     72 0     39 0
```

This example assumes a bunch spacing of 25 ns, i.e. 3564 possible bunch positions around the ring and therefore 7128 possible collision points. The bunches are specified by their position around the ring, i.e. with a number between 1 and 3564 for the spacing of 25 ns. Please note that some numbers will not exist since not all possible positions are filled [10, 12]. All collision points, head-on as well as long range, are specified by their number, i.e. from 1 to 7128 in this example. In the first file (coll.in) the collision points are specified. The first column indicates the position of the central collision point, the second column whether a head-on collision will take place (2) or not (0). The third and fourth column give the number of long range collisions on either side (negative values for left side, positive values for right side) of the central collision point. It should be mentioned, that the collision point in IP8 is shifted by 3/2 times 25 ns and the head-on collision is therefore at 6235. The

symmetric collision scheme would require 6238 for IP8. It must be pointed out, that these are **all possible** head-on and long range collision points, whether they are active or not, depends on the position of a bunch in the train. This type of descriptions is very flexible and additional head-on or long range interactions can be defined easily. Even simpler is the shift of an already defined interaction point by changing its position in the first column.

The description of a full LHC bunch train is given in the second file ($fill.in$). It should be interpreted such that we have 72 bunches (code 1) spaced by 25 ns, followed by 8 empty places (code 0), followed by 72 bunches and so on, until all 3564 are specified. The columns are put into 8 groups in this example and the number of groups is read in at the beginning of the file. The number of groups can be chosen to maximize the readability of the file, a priori it is a free parameter. For other bunch spacings or a different pattern the scheme has to be modified.

The program $COLSCH$ takes the desired bunch number as an argument and uses the above description to provide the collision scheme of that bunch in MAD-X language.

```
Usage:
      colsch   5
```

The output is written into the result file coll.sched. The resulting file can be inserted into a MAD-X run and will control the step 4 of the installation procedure when it is used together with the appropriate macro definitions.

The corresponding input files for a bunch spacing of 75 ns are shown below. The program $COLSCH$ will recognize the number of bunches automatically and hence the number of potential interaction points.

```
   1   2   -5  +5
 298   2   -5  +5
1189   2   -5  +5
2079   2   -5  +5

#Number of groups
 8
     24 1     2 0     24 1     2 0     24 1     14 0     0 0     0 0
     24 1     2 0     24 1     2 0     24 1     14 0     0 0     0 0
     24 1     2 0     24 1     2 0     24 1     2 0     24 1     15 0
     24 1     2 0     24 1     2 0     24 1     14 0     0 0     0 0
     24 1     2 0     24 1     2 0     24 1     14 0     0 0     0 0
     24 1     2 0     24 1     2 0     24 1     2 0     24 1     15 0
     24 1     2 0     24 1     2 0     24 1     14 0     0 0     0 0
     24 1     2 0     24 1     2 0     24 1     14 0     0 0     0 0
     24 1     2 0     24 1     2 0     24 1     2 0     24 1     15 0
     24 1     2 0     24 1     2 0     24 1     14 0     0 0     0 0
     24 1     2 0     24 1     2 0     24 1     14 0     0 0     0 0
     24 1     2 0     24 1     2 0     24 1     2 0     24 0     15 0
```

3.6 Self-consistent variables

In the standard tracking studies, the separation between the two beams is assumed constant and is calculated from the unperturbed machine optics. In principle, the separation between the two beams is changed by the beam-beam interaction and requires a self-consistent calculation of the new closed orbits of the two beams [15, 16].

This is done with the program *TRAIN*. The calculation with *TRAIN* is preceded by a MAD-X run which computes the optics parameters at all possible beam-beam elements and the second order maps between them for both beams.

The *TRAIN* program uses the same input files as *COLSCH* to set up the collision schedule (i.e. *fill.in* and *coll.in*). Next it finds an initial closed orbit from the linear one-turn map with all beam-beam elements switched off. It then iterates over all bunches in both rings until a self-consistent orbit (i.e. distance between the bunches in the two rings at the interaction points) is found with all beam-beam elements switched on. The bunch sizes are assumed to be unchanged in this process. Once all orbits are known, each bunch pair is tracked with the second order maps to get the tunes, chromaticities and dispersion.

For the tracking with MAD-X, we need the orbit values at the beam-beam encounters for a given bunch. A bunch can be specified to *TRAIN* and its orbit values at all beam-beam encounters are printed to a file. This file together with the appropriate macro-definitions, completes the definition of self-consistent beam-beam elements. The difference between the self-consistent and the perturbative treatment is normally rather small, since the machine is designed for that. However, is certain cases such as ultimate intensities and other crossing schemes, this difference can become important.

The appendix II gives a listing of the completed beam-beam elements around interaction point 5 and generated as described for bunch number 8. The 5 beam-beam elements with a charge of 0.2 are the sliced head-on collisions, each representing 1/5 of the beam-beam interactions.

Of course bunch number 8 has only 7 long range interactions on the left side of the interaction point (and all 15 on the right side).

4 Examples

Examples of the described programs and features can be found in [17]. It contains the following programs and command files:

colsch

Takes the input from the files *fill.in* and *coll.in* and prepares the collision schedule *coll.sched*. It takes the wanted bunch number as a command line argument. If the specified bunch does not exist, i.e. corresponds to a gap in the train, a warning message is issued and the program is terminated.

tracoll.madx

Using the official LHC data files, it sets up all necessary beam-beam elements. Their installation is controlled by *coll.sched*. This file is produced by a preceding execution of *colsch*. Certain interaction regions can be entirely switch on and off by flags in *tracoll.madx*. It tracks particles with different amplitudes and analyses the tunes to produce a file *foot_print* (executing the commands **dynap** and **foot**) which can be visualized e.g. with gnuplot to produce the tune footprints.
(usage with gnuplot : *plot "foot_print" with lines*).

prep.madx

A MAD-X run to prepare the necessary input for the *ztrain* program. It can be used without modification for any bunch filling scheme where the spacing is a multiple of 25 ns and the length of the common part of the vacuum chamber is smaller than 60 m on both sides.

ztrain

This is a slightly modified version of *TRAIN* and takes the optics and maps produced by *prep.madx* and with the directive from *train.in* calculates the self-consistent parameters. The self-consistent orbits are written to the two files *beam*1.*orb* and *beam*2.*orb* for the selected bunch number. The bunch is selected in the input file *train.in* which is shown in appendix III. The first bunch that appears in the line *list of out_bunches* is selected. For its purpose as a front-end to trackng with MAD-X, this input file is sufficient. For all more advanced calculations that can be performed with *ztrain*, another set of parameters is required.

trascon.madx

A MAD-X run to track particles on the self-consistent orbits stored in *beam*1.*orb* and *beam*2.*orb*. In case the particles are tracked through the LHC sequence for beam 1, the latter must be used. The collision schedule is defined in the file *coll.sched* produced as described. In order to get meaningful results, it is highly recommended to use a consistent (i.e. the same) bunch number for the preparation with *ztrain* and *colsch*.

macros

These files contain predefined macros used by the examples to set up beam-beam elements etc.

foot

A program that takes the table "dynaptune" from the command **dynap** and converts it into a file that can be plotted as a footprint (i.e. takes care of connecting the mesh points etc.).

5 APPENDIX I

The latest version of the filling scheme looks like:

```
File with filling pattern:
#Number of groups
 8
     72 0     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     8 0     72 1     39 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     8 0     72 1     39 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     8 0     72 1     39 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     38 0     0 0     0 0
     72 1     8 0     72 1     8 0     72 1     8 0     72 1     39 0
```

6 APPENDIX II

Complete set of beam-beam elements for bunch number 8 around IP5.

```
bbip5pl7: beambeam,
    sigx:= 0.0007949737386,sigy:= 0.0008692015205,
    xma:=0.003850011,yma:= -1.839e-05,
    charge:=  1;
bbip5pl6: beambeam,
    sigx:= 0.0007130751649,sigy:= 0.0007130751648,
    xma:=0.003452958,yma:=-1.5128e-05,
    charge:=  1;
bbip5pl5: beambeam,
    sigx:= 0.0005942939162,sigy:= 0.0005942939162,
    xma:=0.002877272,yma:=-1.2657e-05,
    charge:=  1;
bbip5pl4: beambeam,
    sigx:= 0.0004755302744,sigy:= 0.0004755302744,
    xma:=0.002301614,yma:=-1.0186e-05,
    charge:=  1;
bbip5pl3: beambeam,
    sigx:= 0.0003568018215,sigy:= 0.0003568018215,
    xma:=0.001725993,yma:= -7.715e-06,
    charge:=  1;
bbip5pl2: beambeam,
    sigx:= 0.0002381611904,sigy:= 0.0002381611904,
    xma:= 0.00115042,yma:= -5.244e-06,
    charge:=  1;
bbip5pl1: beambeam,
    sigx:= 0.0001198694302,sigy:= 0.0001198694302,
    xma:= 0.00057492,yma:= -2.773e-06,
    charge:=  1;
bbip5l2: beambeam,
    sigx:= 1.594031652e-05,sigy:= 1.594031653e-05,
    xma:= 7.441999894e-06,yma:=  -3.020000129e-09,
    charge:=  0.2;
bbip5l1: beambeam,
    sigx:= 1.586568505e-05,sigy:= 1.586568506e-05,
    xma:= 2.548249889e-06,yma:= -3.02000012e-09,
    charge:=  0.2;
bbip5: beambeam,
    sigx:=  1.58531655e-05,sigy:=  1.58531655e-05,
    xma:=  -4.330001137e-09,yma:=  -3.020000114e-09,
    charge:=  0.2;
bbip5r1: beambeam,
    sigx:= 1.586568505e-05,sigy:= 1.586568505e-05,
```

```
    xma:=  -3.414249967e-06,yma:=  -3.020000108e-09,
    charge:=  0.2;
bbip5r2: beambeam,
    sigx:= 1.594031653e-05,sigy:= 1.594031652e-05,
    xma:=  -8.307999971e-06,yma:=  -3.020000099e-09,
    charge:=  0.2;
bbip5pr1: beambeam,
    sigx:= 0.0001198694302,sigy:= 0.0001198694301,
    xma:=-0.00057615,yma:=  2.043e-06,
    charge:=  1;
bbip5pr2: beambeam,
    sigx:= 0.0002381611905,sigy:= 0.0002381611903,
    xma:=    -0.001152014,yma:=  4.389e-06,
    charge:=  1;
bbip5pr3: beambeam,
    sigx:= 0.0003568018216,sigy:= 0.0003568018214,
    xma:=-0.00172795,yma:=  6.735e-06,
    charge:=  1;
bbip5pr4: beambeam,
    sigx:= 0.0004755302745,sigy:= 0.0004755302743,
    xma:=    -0.002303935,yma:=  9.081e-06,
    charge:=  1;
bbip5pr5: beambeam,
    sigx:= 0.0005942939163,sigy:= 0.0005942939161,
    xma:=    -0.002879956,yma:= 1.1427e-05,
    charge:=  1;
bbip5pr6: beambeam,
    sigx:=  0.000713075165,sigy:= 0.0007130751647,
    xma:=    -0.003456007,yma:= 1.3773e-05,
    charge:=  1;
bbip5pr7: beambeam,
    sigx:= 0.0008692015207,sigy:= 0.0007949737384,
    xma:=    -0.004213039,yma:= 1.5405e-05,
    charge:=  1;
bbip5pr8: beambeam,
    sigx:=  0.001123262812,sigy:= 0.0007884629068,
    xma:=    -0.005441869,yma:= 1.5325e-05,
    charge:=  1;
bbip5pr9: beambeam,
    sigx:=  0.001403592443,sigy:= 0.0007565669835,
    xma:=-0.00671924,yma:= 1.4752e-05,
    charge:=  1;
bbip5pr10: beambeam,
    sigx:=  0.001542610005,sigy:= 0.0008004900362,
    xma:=    -0.007320783,yma:= 1.5657e-05,
```

```
    charge:=  1;
bbip5pr11: beambeam,
    sigx:=  0.001533617994,sigy:= 0.0009221909425,
    xma:=    -0.007219774,yma:= 1.8084e-05,
    charge:=  1;
bbip5pr12: beambeam,
    sigx:=  0.001360570143,sigy:=  0.001143896707,
    xma:=    -0.006346487,yma:= 2.2472e-05,
    charge:=  1;
bbip5pr13: beambeam,
    sigx:=  0.001147227525,sigy:=  0.001387836215,
    xma:=    -0.005283438,yma:= 2.7295e-05,
    charge:=  1;
bbip5pr14: beambeam,
    sigx:=  0.001041525619,sigy:=  0.001498098382,
    xma:=    -0.004715182,yma:=  2.949e-05,
    charge:=  1;
bbip5pr15: beambeam,
    sigx:=   0.00103198502,sigy:=  0.001472561096,
    xma:=    -0.004583487,yma:= 2.9011e-05,
    charge:=  1;
```

7 APPENDIX III (train.in)

```
# collision scheme input file
./scheme.apr00
# write or execute flags:
# 1=write, 2=coll, 3=frequ, 4=equ, 5=set, 6=alt files, 7=orbit, 8=2nd-order,
# 9=w_detail, 10=b_curr, 11=emitt, 12=unused
 1 0 0 0 0 0 1 0 0 0 0 0
# 1=full_coll 2=nturn 3=debug 4=# of out_bunches 5=out_pos 6=write_norm
# 7=xi_fact 8=hofact 9=amp_bunch (0=all,- every..) 10= amp_fac (see below)
# 11=lumi_hist 12=beam_2 offset (half-buckets)
# 1    2   3   4   5   6   7   8   9  10  11  12
  1    0   0   4  16   0   1   1   0   0   0   0
# list of out_bunches
   8 0 0 0 0 0 0 0 0 0 0 0 0 0
# orbit output file
train.orb
# general output file
train.list
# number of bunches for tune
0
# beam 1 sequence name
lhcb1
# beam 2 sequence name
lhcb2
# beam 1 sequence terminator
ip3.m
# beam 2 sequence terminator
ip3.p
# start amplitude in sigmas (x + y); amp_fac=0: t_gauss, else value
0 0 0 0
#0.1d0 0.d0 0 0
# 1 below:
# b_curr = 0: beam current spread sigma (units of beam-current)
# b_curr = 1: factor for even bunches
# b_curr = 2: flat +-
# 2 below: rel. sigma for emittance (t_gauss)
# emitt = 0: emittance spread sigma (units of emittance)
# emitt = 1: factor for even bunches
# emitt = 2: flat +-
# emitt = 3: factor for first half of bunches
0.0 0.0
# random seed
12345678.d0
# end
```

References

[1] H. Grote, L.H.A. Leunissen and F. Schmidt; *LHC Dynamic Aperture at Collision*, LHC Project Note 197 (1999)

[2] W. Herr and F. Jones; *Parallel computation of beam-beam interactions including longitudinal motion* ,Proc. PAC 2003, Portland 12.- 16. May 2003 (2003), CERN-AB-2003-019-ABP (2003).

[3] W. Herr and J. Miles; *A Comparative Study of Beam-beam Tune Footprints for Colliding Beams with a crossing angle and offset vertex in LHC V4.1*, LHC Project Note 04 (1995)

[4] H. Grote and O. Meincke; *Tune footprints for collision optics 5.0*, LHC Project Note 161 (1998)

[5] Y. Luo and F. Schmidt; *Dynamic Aperture Studies for LHC Optics Version 6.2 at Collision*, LHC Project Note 310 (2001)

[6] *The MAD-X Home Page, http://frs.home.cern.ch/frs/Xdoc/mad-X.html.*

[7] F. Schmidt; *SixTrack: Version 3, single particle tracking code treating transverse motion with synchrotron oscillations in a symplectic manner*, CERN/SL/94-56 (AP)

[8] M. Hayes and F. Schmidt; *Run environment for Sixtrack*, LHC Project Note 300 (2003).

[9] F. McIntosh and F. Schmidt; *A new run environment for Sixtrack*, LHC Project Note (2004) not yet published.

[10] W. Herr; *Effects of PACMAN bunches in the LHC*, LHC Project Report 39 (1996)

[11] W. Herr; *Missing head-on collisions for different LHC filling schemes*, LHC Project Note 321 (2003)

[12] P. Collier; *New base line proton filling scheme*, Minutes of the 2nd LHC Technical Committee, 19.2.2003 (2003).

[13] R. Garoby; *Base line proton filling scheme with 72 bunches*, Minutes of the 59th LHC Parameters and Layout Committee, 12.4.2000. (2000)

[14] W. Herr; *Features and implications of different LHC crossing schemes*, LHC Project Report 628 (2003)

[15] H. Grote; *Self-consistent orbits with beam-beam effects in the LHC*, Proc. of EPAC 2000, Vienna, 26. - 30. 6. 2000, (2000), 1202.

[16] H. Grote and W. Herr; *Self-consistent orbits with beam-beam effects in the LHC*, Proc. of the 2001 workshop on beam-beam effects, FNAL, 25.6.-27.6.2001, (2001)

[17] *Example page for beam-beam tracking*, http://cern.ch/Werner.Herr/BBEX.