CERN-SL Note / 97-27 (AP)

Collision Schedules

John M. Jowett

18 March 1997

1 Introduction

Consider a circular collider with a set of, in general, irregularly spaced bunches in each beam. Where do these bunches collide? This is the "collision schedule". The bunches move on the space-time torus formed by the product of the circumference and the revolution time. This can be represented as a square on a 2D screen or paper, with opposite edges identified.

Some years ago, I wrote a first version of this Mathematica notebook while studying pretzel schemes with irregularly spaced bunches in LEP. It was helpful in implementing the module of the WIGWAM program that finds all the encounters and computes parasitic beam-beam effects at each of them. Following Eberhard Keil's call for a labelling or characterisation of the collision schedule at the Forum on Beam-beam Effects in the LHC on 6/3/97, I thought it worth distributing as a contribution to the discussion. It may also be of technical interest as an application of *Mathematica*. Of course, alternative approaches to the problem are contained or implicit in references quoted by others at the Forum.

1.1 Access to this notebook

This notebook is best read interactively. Section 2 defines a function **`encounters`** that solves the problem and the remaining sections give worked examples, including the LHC. A mininal knowledge of *Mathematica* should be enough to follow these and apply the function. To understand it in detail and go beyond the examples given will require more knowledge of *Mathematica*, including the rudiments of functional programming. Although developed under Windows, the notebook is available in the formats of *Mathematica* 3.0 and 2.0 as

```
/afs/cern.ch/user/j/jowett/public/math/BeamBeam/CollisionSchedule.nb
/afs/cern.ch/user/j/jowett/public/math/BeamBeam/CollisionSchedule.ma
```

In addition, the generally useful functions defined in the initialisation cells of the notebook have been saved in a compact "package" file

```
/afs/cern.ch/user/j/jowett/public/math/BeamBeam/CollisionSchedule.m
```

If you like, you can just load this package and then treat your own examples, perhaps as part of some "script". The path /afs/...jowett/group/... leads to the same files.

2 Definition of the function encounters

With length in units of the circumference and time in units of the revolution time, the trajectories of bunches of each beam starting at azimuth a at time $t = 0$ are maps of a torus into a circle

```
sp[a_, t_] := Mod[a + t, 1];
sm[a_, t_] := Mod[a – t, 1]
```

The bunches starting at *a* and *b* encounter each other at times:

```
t1[a_, b_] := Mod[(b – a) / 2, 1];
t2[a_, b_] := Mod[1 / 2 + (b – a) / 2, 1]
```

The positions of these encounters are

```
encounters[a_ /; NumericQ[a], b_ /; NumericQ[b]] :=
 {Mod[(a + b) / 2, 1], Mod[(1 + a + b) / 2, 1]};
encounters[{a_, b_}] := enc[a, b];
```

(The syntax **`/;NumericQ[a]`**, etc., means that the expression **`encounters[a,b]`** will only evaluate for numerical arguments. The function **`encounters`** is "overloaded" to deal with other sorts of arguments in the additional definitions which follow.)

Beams will be represented as a list of azimuths $s \in [0, 1\rangle$ representing the positions of the bunches at some time $t = 0$. In fact, a beam is really an equivalence class of such lists under the functions sp and sm, parametrised by the argument *t* (rotations on the circumference).

The function can be extended to accept beams as arguments and generate the complete "collision schedule" alluded to by Eberhard Keil as a 3000×3000 matrix (for the LHC) at the Forum. Here it appears as a matrix of pairs of collision points. The row and column within the matrix tell you which bunches collide there. Examples will be shown below.

```
encounters[beam1_List, beam2_List] := Outer[encounters, beam1, beam2]
```

The following extension of the definition returns only the collisions occurring between azimuths s_1 and s_2. It will be explained in the section on "Two ring colliders" below.

```
encounters[beam1_List, beam2_List, {s1_, s2_}] := Block[{encs, pos},
  encs = encounters[beam1, beam2];
  pos = Position[encs, s_ /; (s1 <= s && s <= s2)];
  Map[(Drop[#, {3}])&, MapThread[Append, {pos, Extract[encs, pos]}]]
  ]
```

```
General::spell1 :
 Possible spelling error: new symbol name "encs" is similar to existing symbol "enc".
```

The function **`encounters`** is the complete solution of the collision schedule problem.

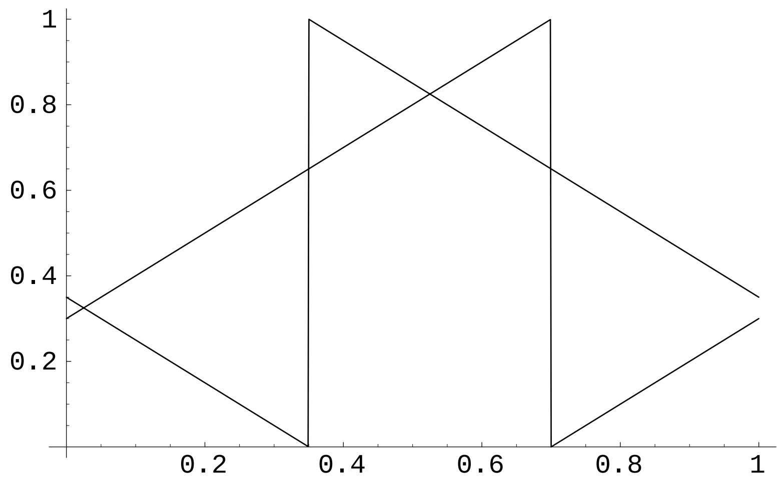
(Unfortunately, the function **`Extract`** is not available in *Mathematica* versions prior to 3.0. The package file **`CollisionSchedule.m`** contains a slightly more complicated version of **`encounters`** that will work with all versions.)

3 Basic understanding

Consider a single bunch of one beam starting at *a* colliding with a single bunch from the other beam starting at *b*. Choose *a* and *b* at random and look at what happens by evaluating the following compound expression. The vertical segments in the plots show hwo the square is mapped into a torus.

```
a = 3 / 10; b =  7 / 20;
Print["Initial positions: ", {a, b}];
Plot[{sp[a, t], sm[b, t]}, {t, 0, 1}];
Print["Times of encounters: ", {t1[a, b], t2[a, b]}];
Print["Position of each beam at first encounter: " ,
 {sp[a, t1[a, b]], sm[b, t1[a, b]]}];
Print["Position of each beam at second encounter: ",
 {sp[a, t2[a, b]], sm[b, t2[a, b]]}];
Print["Test encounter position function:" , encounters[a, b]]
```

Initial positions: $\left\{\frac{3}{10}, \frac{7}{20}\right\}$



Times of encounters: $\left\{\frac{1}{40}, \frac{21}{40}\right\}$

Position of each beam at first encounter: $\left\{\frac{13}{40}, \frac{13}{40}\right\}$

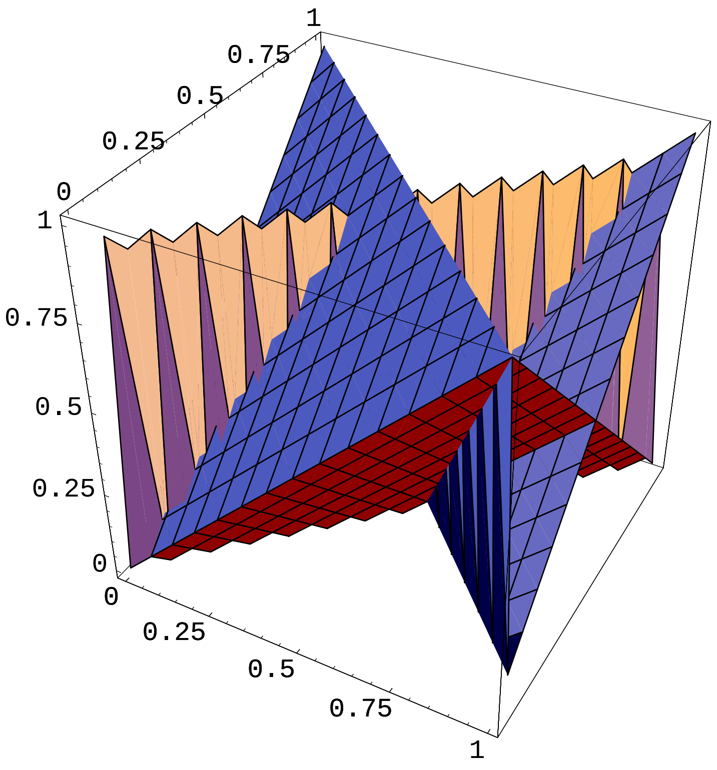
Position of each beam at second encounter: $\left\{\frac{33}{40}, \frac{33}{40}\right\}$

Test encounter position function: $\left\{\frac{13}{40}, \frac{33}{40}\right\}$

4 Global pictures

For a more global perspective we can look at the geometry of sheets of intersecting trajectories on the space time torus. Time is vertical, the other axes are the initial positions of the bunches of each beam. This maps a 3-torus (embedded in a 4D Euclidean space, if you like) into a cube in our 3D space with opposite faces identified.

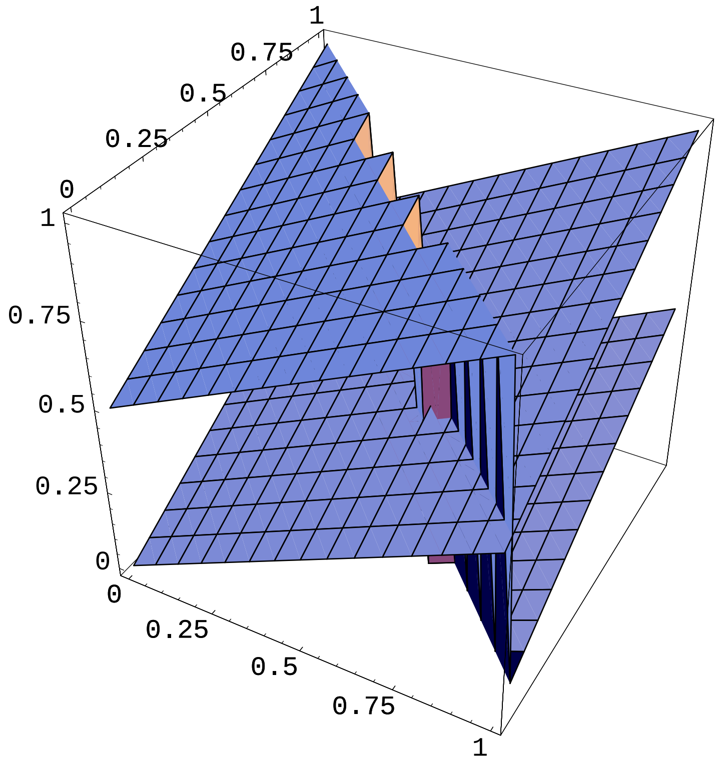
```
aa = Plot3D[sp[x, y], {x, 0, 1}, {y, 0, 1}, DisplayFunction -> Identity];
bb = Plot3D[sm[x, y], {x, 0, 1}, {y, 0, 1}, DisplayFunction -> Identity];
Show[{aa, bb}, DisplayFunction -> $DisplayFunction, BoxRatios -> Automatic]
```



```
- Graphics3D -
```

Plot the two collision points on the vertical axis as a function of initial positions in the same way.

```
aa = Plot3D[First[encounters[x, y]],
  {x, 0, 1}, {y, 0, 1}, DisplayFunction -> Identity];
bb =
 Plot3D[Last[encounters[x, y]], {x, 0, 1}, {y, 0, 1}, DisplayFunction -> Identity];
Show[{aa, bb}, DisplayFunction -> $DisplayFunction, BoxRatios -> Automatic]
```



```
- Graphics3D -
```

5 Examples

5.1 All the encounters between two arbitrary beams

Here is an example for two beams with a random floating-point distribution of a few bunches for illustration (in reality, of course, the bunch positions should always differ by a multiple of the inverse of the RF harmonic number)

```
beam1 = Table[Random[], {5}]
{0.377582, 0.557663, 0.774711, 0.649021, 0.941059}
```

```
beam2 = Table[Random[], {3}]
{0.338029, 0.518704, 0.327615}
```

```
collisions = encounters[beam1, beam2];
collisions // TableForm
0.357805      0.448143      0.352598
0.857805      0.948143      0.852598

0.447846      0.538183      0.442639
0.947846      0.0381831     0.942639

0.55637       0.646708      0.551163
0.0563702     0.146708      0.0511632

0.493525      0.583863      0.488318
0.993525      0.0838626     0.988318

0.639544      0.729881      0.634337
0.139544      0.229881      0.134337
```

Results are invariant if we shift buckets left in one beam and right in the other by the same amount (equivalent to a shift in time).

```
(encounters[beam1 + .1, beam2 - .1] - collisions ) // Chop
{{{0, 0}, {0, 0}, {0, 0}}, {{0, 0}, {0, 0}, {0, 0}},
 {{0, 0}, {0, 0}, {0, 0}}, {{0, 0}, {0, 0}, {0, 0}},
 {{0, 0}, {0, 0}, {0, 0}}}
```

We can always take particular bunches in each beam.

```
encounters[beam1[[5]], beam2[[2]] ]
{0.729881, 0.229881}
```

Of course, much of the data in the allCollision matrix is redundant in cases of real interest. Let us go on to illustrate how it can be condensed.

5.2 Two-ring colliders

In a two-ring collider like the tau-charm Factory or LHC, collisions only happen in the range of azimuths where the two rings come together. Here is one way to treat this, retaining all the information about which bunches are colliding. Consider a common section between s_1 and s_2 and define a function that tests whether a collision point is in it.

```
hit[s_, s1_, s2_] := If[ s1 < s && s < s2 , s, Null]
```

Continuing with the example of the previous subsection, we can apply this to the collision object at the appropriate depth to show where the real collisions are:

```
hits = Map[(hit[#, .45, .55])&, collisions, {3}];
hits // TableForm
Null        Null        Null
Null        Null        Null

Null        0.538183    Null
Null        Null        Null

Null        Null        Null
Null        Null        Null

0.493525    Null        0.488318
Null        Null        Null

Null        Null        Null
Null        Null        Null
```

Collecting the indices gives a list of the bunches which actually collide

```
hitspos = Position[hits, _Real]
{{2, 2, 1}, {4, 1, 1}, {4, 3, 1}}
```

The corresponding collision points are

```
hitsencs = Extract[hits, hitspos]
{0.538183, 0.493525, 0.488318}
```

(In *Mathematica* versions before 3.0, the syntax `Map[ (Part[hits,Apply[Sequence,#]])&,hitspos]` will return the same reult.)

From this we can build an object which contains all the information about which bunches collide and where they do so:

```
MapThread[Append, {hitspos, hitsencs}]
{{2, 2, 1, 0.538183}, {4, 1, 1, 0.493525}, {4, 3, 1, 0.488318}}
```

The third element can be discarded, to give a set of collisions identified by

{<bunch number from beam 1>,< bunch number from beam 2>,< azimuth of collision>}

```
Map[(Drop[#, {3}])&, %]
{{2, 2, 0.538183}, {4, 1, 0.493525}, {4, 3, 0.488318}}
```

Now we can understand the final part of the definition of `encounters`. It overloads the previous part of the definition to incorporate the calculations above in a single function that returns the collisions in a common section of a ring. This considerably reduces the volume of data in cases like the LHC. We can test it on the present example:

```
encounters[beam1, beam2, {.45, .55}]
{{2, 2, 0.538183}, {4, 1, 0.493525}, {4, 3, 0.488318}}
```

Full LHC-sized examples can be done with sufficient computer memory (see later). Rings with two or more common sections are easily treated by combining the collisions in each section.

5.3 Bunch trains

Here is a function to make bunch trains: each train consists of nBunches bunches spaced by the parameter spacing (given in units of the circumference).

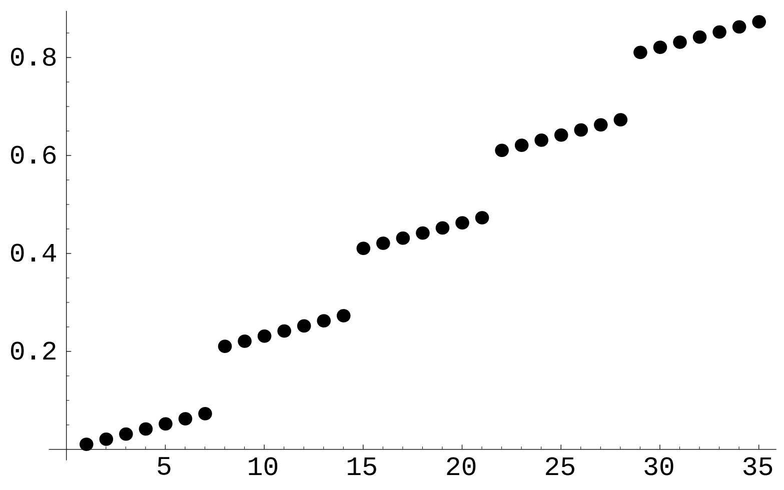
```
fractab[k_] := Table[j / k, {j, 1, k}]
```

```
trains[nTrains_Integer, nBunches_Integer, spacing_] :=
 Mod[Table[k / nTrains + fractab[nBunches] nBunches spacing, {k, 1, nTrains}], 1]
```

```
trains[2, 4, 1 / 21]
```

$$\left\{\left\{\frac{23}{42}, \frac{25}{42}, \frac{9}{14}, \frac{29}{42}\right\}, \left\{\frac{1}{21}, \frac{2}{21}, \frac{1}{7}, \frac{4}{21}\right\}\right\}$$

```
ListPlot[Sort[Flatten[trains[5, 7, 1 / 96]]], Prolog -> AbsolutePointSize[5]]
```



```
- Graphics -
```

As an example we can take bunch trains with an extra "pilot" bunch in the first beam.

```
beam1 = Join[Sort[Flatten[trains[4, 2, 1 / 24]]], {1 / 48}]
```

$$\left\{\frac{1}{24}, \frac{1}{12}, \frac{7}{24}, \frac{1}{3}, \frac{13}{24}, \frac{7}{12}, \frac{19}{24}, \frac{5}{6}, \frac{1}{48}\right\}$$

```
beam2 = Sort[Flatten[trains[4, 2, 1 / 24]]]
```

$$\left\{\frac{1}{24}, \frac{1}{12}, \frac{7}{24}, \frac{1}{3}, \frac{13}{24}, \frac{7}{12}, \frac{19}{24}, \frac{5}{6}\right\}$$

It's often convenient to multiply by the harmonic number. Take it to be 96 for illustration.

```
collisions = 2 96 encounters[ beam1, beam2];
collisions // TableForm
8        12       32       36       56       60       80       84
104      108      128      132      152      156      176      180
12       16       36       40       60       64       84       88
108      112      132      136      156      160      180      184
32       36       56       60       80       84       104      108
128      132      152      156      176      180      8        12
36       40       60       64       84       88       108      112
132      136      156      160      180      184      12       16
56       60       80       84       104      108      128      132
152      156      176      180      8        12       32       36
60       64       84       88       108      112      132      136
156      160      180      184      12       16       36       40
80       84       104      108      128      132      152      156
176      180      8        12       32       36       56       60
84       88       108      112      132      136      156      160
180      184      12       16       36       40       60       64
6        10       30       34       54       58       78       82
102      106      126      130      150      154      174      178
```

List the collision points in order:

```
Sort[Flatten[collisions]]
{6, 8, 8, 8, 8, 10, 12, 12, 12, 12, 12, 12, 12, 12, 16, 16, 16, 16, 30, 32, 32,
 32, 32, 34, 36, 36, 36, 36, 36, 36, 36, 36, 40, 40, 40, 40, 54, 56, 56, 56, 56,
 58, 60, 60, 60, 60, 60, 60, 60, 60, 64, 64, 64, 64, 78, 80, 80, 80, 80, 82, 84,
 84, 84, 84, 84, 84, 84, 84, 88, 88, 88, 88, 102, 104, 104, 104, 104,
 106, 108, 108, 108, 108, 108, 108, 108, 108, 112, 112, 112, 112,
 126, 128, 128, 128, 128, 130, 132, 132, 132, 132, 132, 132, 132,
 132, 136, 136, 136, 136, 150, 152, 152, 152, 152, 154, 156, 156,
 156, 156, 156, 156, 156, 156, 160, 160, 160, 160, 174, 176, 176,
 176, 176, 178, 180, 180, 180, 180, 180, 180, 180, 180, 184, 184, 184, 184}
```

List only the distinct collision points and see how many there are:

```
distinctCollisions = Union[Sort[Flatten[collisions]]]
{6, 8, 10, 12, 16, 30, 32, 34, 36, 40, 54, 56, 58, 60, 64, 78, 80, 82,
 84, 88, 102, 104, 106, 108, 112, 126, 128, 130, 132, 136, 150, 152,
 154, 156, 160, 174, 176, 178, 180, 184}
```

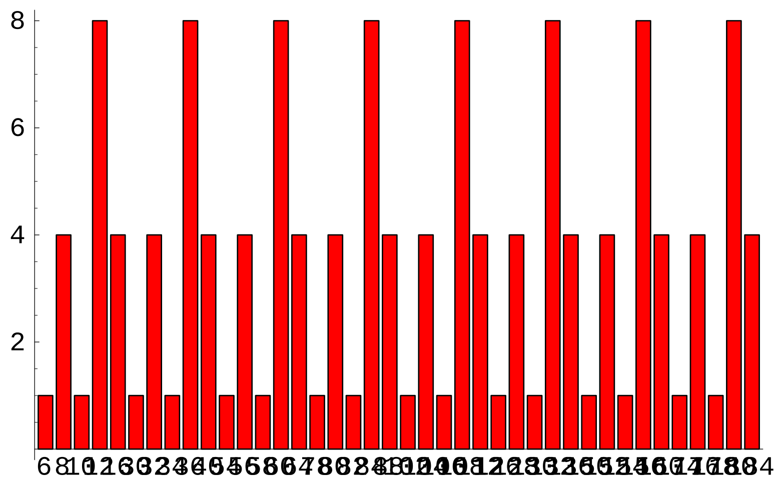
```
Length[distinctCollisions]
40
```

After loading some standard packages, we can look at the frequency distribution of the collisions

```
<< Statistics‘DataManipulation‘; << Graphics‘Graphics‘
```

```
collisionFrequencies = Frequencies[Sort[Flatten[collisions]]]
{{1, 6}, {4, 8}, {1, 10}, {8, 12}, {4, 16},
 {1, 30}, {4, 32}, {1, 34}, {8, 36}, {4, 40}, {1, 54}, {4, 56},
 {1, 58}, {8, 60}, {4, 64}, {1, 78}, {4, 80}, {1, 82}, {8, 84},
 {4, 88}, {1, 102}, {4, 104}, {1, 106}, {8, 108}, {4, 112}, {1, 126},
 {4, 128}, {1, 130}, {8, 132}, {4, 136}, {1, 150}, {4, 152}, {1, 154},
 {8, 156}, {4, 160}, {1, 174}, {4, 176}, {1, 178}, {8, 180}, {4, 184}}
```

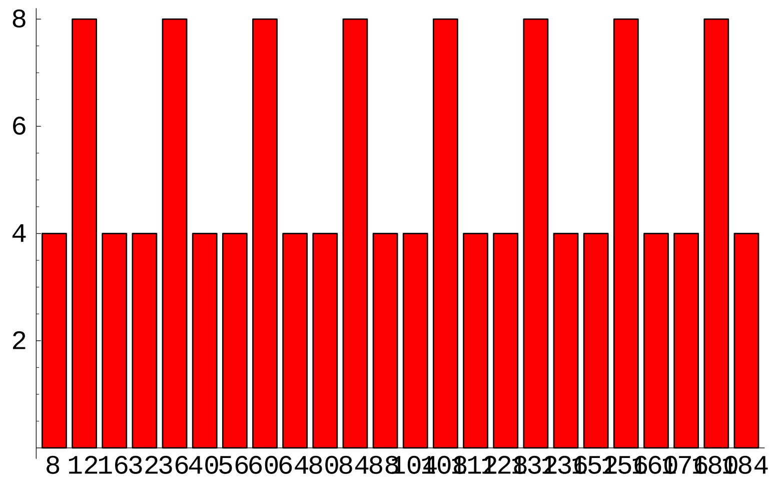
```
BarChart[collisionFrequencies]
```



```
- Graphics -
```

Since we know that if the two beams have identical bunch structure, each collision happens an even number of times, this plot shows how the pilot bunch generates many new collision points. Without it the frequency distribution is:

```
BarChart[Frequencies[Sort[Flatten[2 96 encounters[beam2, beam2]]]]]
```



```
- Graphics -
```

5.4 Collision schedules for the LHC

The circumference of the LHC and the length of the common section of beam pipe on each side of IP1 are

```
circLHC = 26658.883 Meter;
lIP1 = 75 Meter;
```

The RF system for the LHC works on the harmonic

```
hLHC = 35640;
```

which factorises as follows

```
FactorInteger[hLHC]
{{2, 3}, {3, 4}, {5, 1}, {11, 1}}
```

The bunch train in the LHC consists of 2835 bunches spaced 10 RF buckets apart (printed here in abbreviated form):

```
beamLHC1 = Table[10 k / hLHC, {k, 0, 2835 - 1}];
Short[beamLHC1, 2]
```

$$\left\{0, \frac{1}{3564}, \frac{1}{1782}, \frac{1}{1188}, \frac{1}{891}, \frac{5}{3564}, \frac{1}{594}, \frac{7}{3564}, \frac{2}{891}, \frac{1}{396}, \frac{5}{1782}, \ll 2814 \gg, \frac{2825}{3564}, \frac{157}{198}, \frac{257}{324}, \frac{707}{891}, \frac{943}{1188}, \frac{1415}{1782}, \frac{2831}{3564}, \frac{236}{297}, \frac{2833}{3564}, \frac{1417}{1782}\right\}$$

The full collision schedule data for the two beams are very large. For weak-strong simulations in particular, it is more useful to conside one bunch of the opposing beam. As an example, let's take the last bunch

```
beamLHC2 = Take[beamLHC1, {-1}]
```

$$\left\{\frac{1417}{1782}\right\}$$

(Actually, the initial conditions for the two beams should be shifted by an amount depending on their injection phases. I have not done this here.)

Here are all the encounters that bunch 1 of beam2 has with bunches of beam 1 in the straight section on the right of (and including) IP1:

```
collIP1R = encounters[beamLHC1, beamLHC2, {0, lIP1 / circLHC}]
```

$\{\{731, 1, 0\}, \{732, 1, \frac{1}{7128}\}, \{733, 1, \frac{1}{3564}\},$
$\{734, 1, \frac{1}{2376}\}, \{735, 1, \frac{1}{1782}\}, \{736, 1, \frac{5}{7128}\}, \{737, 1, \frac{1}{1188}\},$
$\{738, 1, \frac{7}{7128}\}, \{739, 1, \frac{1}{891}\}, \{740, 1, \frac{1}{792}\}, \{741, 1, \frac{5}{3564}\},$
$\{742, 1, \frac{1}{648}\}, \{743, 1, \frac{1}{594}\}, \{744, 1, \frac{13}{7128}\}, \{745, 1, \frac{7}{3564}\},$
$\{746, 1, \frac{5}{2376}\}, \{747, 1, \frac{2}{891}\}, \{748, 1, \frac{17}{7128}\}, \{749, 1, \frac{1}{396}\},$
$\{750, 1, \frac{19}{7128}\}, \{751, 1, \frac{5}{1782}\}\}$

This may be more useful converted to distances from the IP, suppressing the bunch number from beam 2,

```
collIP1R = collIP1R /. {xx_, yy_, zz_} -> {xx, zz circLHC}
{{731, 0}, {732, 3.74002 Meter}, {733, 7.48005 Meter}, {734, 11.2201 Meter},
 {735, 14.9601 Meter}, {736, 18.7001 Meter}, {737, 22.4401 Meter},
 {738, 26.1802 Meter}, {739, 29.9202 Meter}, {740, 33.6602 Meter},
 {741, 37.4002 Meter}, {742, 41.1403 Meter}, {743, 44.8803 Meter},
 {744, 48.6203 Meter}, {745, 52.3603 Meter}, {746, 56.1003 Meter},
 {747, 59.8404 Meter}, {748, 63.5804 Meter}, {749, 67.3204 Meter},
 {750, 71.0604 Meter}, {751, 74.8005 Meter}}
```

Thus the bunch under consideration in beam 2 meets bunch number 731 of beam 1 at IP1; it then meets bunch number 732 at 3.74 m to the right of IP1 and so on.

On the left of IP1, the bunch has the encounters

```
collIP1L = encounters[beamLHC1, beamLHC2, {1 - lIP1 / circLHC, 1}]
        /. {xx_, yy_, zz_} -> {xx, zz circLHC}
{{711, 26584.1 Meter}, {712, 26587.8 Meter}, {713, 26591.6 Meter},
 {714, 26595.3 Meter}, {715, 26599. Meter}, {716, 26602.8 Meter},
 {717, 26606.5 Meter}, {718, 26610.3 Meter}, {719, 26614. Meter},
 {720, 26617.7 Meter}, {721, 26621.5 Meter}, {722, 26625.2 Meter},
 {723, 26629. Meter}, {724, 26632.7 Meter}, {725, 26636.4 Meter},
 {726, 26640.2 Meter}, {727, 26643.9 Meter}, {728, 26647.7 Meter},
 {729, 26651.4 Meter}, {730, 26655.1 Meter}}
```

5.5 Input for MAD

Let us sketch how to generate a MAD description of the beam-beam collisions in the LHC, continuing with the example of the previous section. One must define appropriate beambeam elements and then generate sequence editor commands to install them.

```
Directory[]
C:\MATH\BeamBeam
```

Define a function that generates MAD commands on two separate files since they need to be executed separately by MAD.

```
MADinstall[filed_, filei_, {bunch_, s_}] := Block[{},
  element = "bb" <> ToString[bunch];
  WriteString[filed, element <> ":", "BEAMBEAM \n"];
  WriteString[filei, "Install,element= ", element, ", at=", s / Meter, "\n"]]
```

To install all these elements, we can map this function over the complete list of encounter points on both sides of the IPs.

```
bbdefs = OpenWrite["bbdefs.mad"];
bbinstall = OpenWrite["bbinstall.mad"];
Map[ (MADinstall[bbdefs, bbinstall, #])&, Join[collIP1L, collIP1R]];
Close[bbdefs]; Close[bbinstall]
bbinstall.mad
```

The contents of the two MAD scripts are

```
!! bbdefs.mad
```

```
bb711:BEAMBEAM
bb712:BEAMBEAM

<<< many lines not shown >>>

bb751:BEAMBEAM
```

```
!! bbinstall.mad
```

```
Install,element= bb711, at=26584.1
Install,element= bb712, at=26587.8

<<< many lines not shown >>>

Install,element= bb751, at=74.8005
```

However the beam-beam element should include the separation of the beams, given by the local orbit, and the dependence of the opposing beam size on the local optical functions. This is done in WIGWAM by propagating the values from the nearest end of an element and computing the variation of the size of the beam accordingly. One way of doing this in MAD is to use the above files in a first step to dump out an OPTICS table at the BEAMBEAM elements. In a second step, the table can be read into Mathematica and used (with an extended version of the function MADinstall) to construct a new version of the file bbdefs.mad where each beam-beam element has the appropriate parameters added to it. The details are straightforward and are not given here.

6 Conclusions

The problem of collision schedules in one- or two-ring colliders can be solved with a single function defined in a few lines of Mathematica code. The potentially large volume of data describing the encounters between bunches in a machine like the LHC can be reduced considerably. It is then easy to generate a description of the collision schedule in the MAD input language.