Notes on PTC-Orbit

Miriam Fitterer

July 23, 2012

Contents

Chapter 1

Documentation

Documentation of PTC:

- About the philosophy and general concept of PTC and details about the tracking routines: [2] (pdf)
- Kind of user guide: Kind of user manual [3] (pdf)

Documentation of Orbit on the SNS orbit website and directly the orbit manual or the local copy (pdf) Orbit was written using SuperCode. Documentation in (ps)

Chapter 2

Creating a flat file for PTC-Orbit

The flat file for ptc-orbit can be created with the help of MADX-PTC. The lattice is defined as usually as a sequence in MADX. The sequence should be optimized in the sense that:

- no elements should be split up, if they are not physically split up in the real machine (e.g. no half quadrupoles) Split up magnets increase the number of integration steps without increasing the order of the symplectic integrator and with it the precision.
- All markers should be thrown out. Additional markers slow down the simulation as they are additional unnecessary nodes.
- cavities should be defined using `no_cavity_totalpath`, e.g.:

  ```
  BR.C02: RFCAVITY, VOLT:=0.008, HARMON:=1, L:=l.cav,
  LAG:=0.5, no_cavity_totalpath;
  ```

2.1 Without space charge nodes using PTC_CREATE_LAYOUT

The command for creating the flat file is the following:

```
ptc_create_universe;
ptc_create_layout, model=2, method=2, nst=5,exact=true;
ptc_script, file="thin4.PTC";
ptc_end;
```

with `thin4.PTC` containing:

```
SELECTLAYOUT
1
PRINT FLAT FILE
PTC_FLAT.TXT
return
```

By calling `ptc_create_layout` one creates the layout in PTC and defines the different ptc nodes.

- The flag `model` specifies the model for cutting with `model=1` for drift-kick-drift and `model=2` for matrix-kick-matrix and `model=3` for delta-matrix-kick-matrix (as in SixTrack-code model) (MADX-Index). For PTC-orbit simultion `model=2` should be used. According to Werner `model=2` takes the exact matrix for dipoles and quadrupoles. Frank Schmidt said that it just applies the correct phase advance between two integration nodes. `model=3` is just for benchmarking with sixtrack and should not be used.

- The flag `method` specifies the order of the symplectic integrator, with `method=2` for drift-kick-drift (which is always symplectic), `method=4` for 4th order integrator and `method=6` for Yoshida [1] (pdf), which defines higher order integrators iteratively. See for more details p. 85 et seq. (Symplectic Integration and Splitting) [3] (pdf).
- The flag `nst` specifies the number of integration steps per element.
- The flag `exact` specifies if the exact Hamiltonian is used (`exact=true`) or the expanded Hamiltonian for small momenta (`exact=false`).
- The flag `resplit` applies the PTC resplit procedure. This is meant to create an adaptive setting of the `method` and `nst` attributes according to the strength of the quadrupoles (using the `thin` attribute) and respectively the dipoles (using the `xbend` attribute). Additionally, there is the `even` attribute for even and odd number of splits (MADX-Index). The `thin` flag is the main `resplit` attribute. It is meant for splitting quadrupoles according to their strength. The default of `thin=0.001` has shown in practice to work well without costing too much with respect of performance. The `xbend` attribute is meant for splitting dipoles, e.g. `xbend=0.001`. It is an optional `resplit` attribute and therefore has the default set to -1, which means no splitting. A splitting by `xbend=0.001` may be advisable for dipoles as well.

  `xbend` can only be used in combination with `exact=true` (otherwise there is some sort of inconsistency in the tracking routines).

  Sextupoles are also split up by this algorithm but the number of integration steps cannot be controlled.
- The even switch ensures even number of splits when using the PTC `resplit` procedure. The default is `even=true`. This is particularly useful when one attempts to calculate PTC_TWISS with the CENTER_MAGNETS option, i.e. if one would like to calculate the twiss parameters in the center of the element. Uneven number of splits will be achieved with `even=false`.

The script `thin4.ptc` just dumps the created ptc description of the lattice in a flat file.

2.2 With Space Charge Nodes using the PTC-Orbit algorithm

2.2.1 Syntax

To create a layout taking the later placement of space charge nodes into account the algorithm is defined in the script `thin4.ptc` called by the following command:

```
ptc_create_universe;
ptc_create_layout, model=2, method=2, nst=1,exact=true;
ptc_script, file="thin4.PTC";
ptc_end;
```

with `thin4.PTC` containing:

```
SELECTLAYOUT
1
CUTTING ALGORITHM
```

```
2
LIMIT FOR CUTTING
10000 10000
lmax
1.0d0
fuzzylmax
0.10
THIN LENS
0.1
THIN LENS
-0.01
0.10 -1
PRINT FLAT FILE
PTC_FLAT.TXT
return
```

The attributes `model` and `exact` are defined with the `ptc_create_layout` command. All other attributes are overwritten by the PTC-Orbit algorithm called in `thin4.PTC`. Explanation of the different commands for the PTC-Orbit cutting algorithm called in `thin.PTC`:

- `SELECTLAYOUT` selects the ptc layout
- `CUTTING ALGORITHM` chooses the cutting algorithm for the ptc nodes. `2` stands for cutting all elements, `1` for cutting only drifts and `0` ... .
- `LIMIT FOR CUTTING` sets a limit for the maximum number of integration nodes. This number is set to a high number so that the routine does not jump to a higher order integrator instead of splitting the magnet in more pieces.
- `lmax` defines the maximum distance between the orbit nodes in m. `1.0d0` is e.g. a maximum distance of 1.0 m between the nodes. `d0` stand for double precision number.
- `fuzzylmax` specifies the accuracy of `lmax`, is probably the parameter `fuzzy_split` on p. 85 et seq. (Symplectic Integration and Splitting) [3] (pdf).
- splitting quadrupoles:

  ```
  THIN LENS
  0.1
  ```

  `thin`lense=0.1 > 0 specifies the maximum integrated quadrupole strength. So e.g. with `thinlens=0.1` a quadrupole of 1 m length and $k = 2.2$, would be split up in 22 parts, so that the integrated length of one part would be 0.1 or smaller.
- splitting sextupoles:

  ```
  THIN LENS
  -0.01
  0.10 -1
  ```

  `thinlens=-0.01`< 0 defines the number of thin lenses similar to the case of the quadrupole splitting procedure. `0.10` is the "sexr0" parameter in [m] defined by $kl = |k_s * sexr0| * l$. `-1` is a flag for no effect for `XBEND`.

- `PRINT FLAT FILE` prints the flat file in the specified filename (here `RCS_PTC_FLAT.TXT`)
- `return` gives the command to return to ptc.

The produced flat file looks for example like this for a description of the dipole (SBEND):

```
...
MAGNET CHART MAGNET CHART MAGNET CHART MAGNET CHART ...
 T     2     2     5  EXACT METHOD NST NMUL
1.50   1.497  0.139  TILT=0.0E+0 LD LC B0
0.519   0.854  2.701   0.570  BETA0 GAMMA0I GAMBET P0C
0.104   0.104  EDGES
...
```

The first line:

```
T           2           2           5  EXACT METHOD NST NMUL
```

specifies the integration method. In this case we have `exact=true` (`T` stand for `exact=true` and `F` for false). The `method` is 2, the number of integration steps `nst` is 2. What is `NMUL`?
The second line:

```
1.50   1.497  0.139  TILT=0.0E+0 LD LC B0
```

gives the path length of the element (`LD`, here 1.5), the length of the element? (`LC`, here 1.497), the magnetic dipole field component b_0 (`B0,` here 0.139) and the `TILT`, which is self explanatory.
The third line:

```
0.519   0.854  2.701   0.570  BETA0 GAMMA0I GAMBET P0C
```

gives the relativistic β (`BETA0`, here 0.519), the inverse of the relativistic γ (`GAMMA0I`, here 0.854), $(\frac{1}{\beta\gamma})^2$ (`GAMBET`, here 2.701), and the momentum times the speed of light pc in GeV (`P0C`, here 0.570)
The fourth line:

```
0.104   0.104  EDGES
```

gives the angles of the edges. Here an angle of 0.104 was chosen in the SBEND definition:

```
MB: SBEND,L=1.5, ANGLE=0.209, E1=0.104, E2=0.104;
```

2.2.2 Algorithm

The PTC-Orbit cutting algorithm is called in the file `thin4.PTC`, where

```
CUTTING ALGORITHM
2
```

specifies to use 2nd order integrators for the integration. Again `LMAX` is the distance between the space charge nodes. The algorithm then tries to group the elements of the lattice in nodes of approx `LMAX` length, but always smaller than `LMAX`. Between each ptc node the algorithm then places a space charge node.
All markers should be removed of the sequence in order to get an optimal description of the lattice in respect of precision and cpu time. Markers would introduce an artificial splitting of the element in probably an ineffecient and not optimal way in the sense of symplectic integration.
Running ptc-orbit one can print out (or dump in a file) the nodes by using the command

```
OFstream fio1("Nodes_RNG.out", ios::out);
showNodes(fio1);
```

This only shows a list of the nodes:

```
Node #  Name                    Index   Length        Position      Type
------  ----------------------  ------  ------------  ------------  -----------
1       PTC(0)                  0       0.741869      0.741869      ptc
2       PTC(1)                  10      0.741869      1.48374       ptc
3       PTC(2)                  20      0.8           2.28374       ptc
4       PTC(3)                  30      0.8           3.08374       ptc
5       PTC(4)                  40      0.95          4.03374       ptc
...
```

`Index` is the orbit index. For each ptc-node 10 orbit index are reserved to allow for later insertion of additional elements like foils etc. In this example we chose `LMAX=1.0`. The file `Negative_node.dat`, which is created automatically then shows a detailed description of the different ptc nodes. In our example:

```
  Number of PTC Nodes           181
 ****************************************************
  Node Number           0
  Number of PTC Integration Nodes           8
  0.000000000000000E+000 MYRING\$START                       1          -1
  0.000000000000000E+000 MYRING\$START                       1           1
  0.000000000000000E+000 MYRING\$START                       1           0
  0.000000000000000E+000 MYRING\$START                       1           2
  0.000000000000000E+000 MYRING\$START                       1          -2
  0.000000000000000E+000 DRIFT_0                            2          -1
  0.000000000000000E+000 DRIFT_0                            2           1
  0.000000000000000E+000 DRIFT_0                            2           0
 ****************************************************
  Node Number           1
  Number of PTC Integration Nodes           5
  0.741869260617025      DRIFT_0                            2           0
   1.48373852123405      DRIFT_0                            2           2
   1.48373852123405      DRIFT_0                            2          -2
   1.48373852123405      MB1.C1                             3          -1
   1.48373852123405      MB1.C1                             3           1
 ****************************************************
  Node Number           2
  Number of PTC Integration Nodes           1
   1.48373852123405      MB1.C1                             3           0
 ****************************************************
  Node Number           3
  Number of PTC Integration Nodes           5
   2.28373852123405      MB1.C1                             3           0
   3.08373852123405      MB1.C1                             3           2
   3.08373852123405      MB1.C1                             3          -2
   3.08373852123405      DRIFT_1                            4          -1
   3.08373852123405      DRIFT_1                                    4            1
...
```

Fig. 2.2.1 illustrates the cutting algorithm

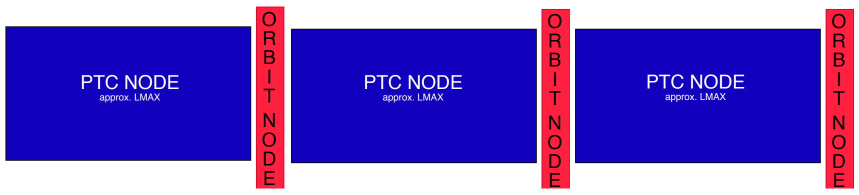


Figure 2.2.1: Placement of PTC-Orbit Nodes

Chapter 3

Defining Time Dependent Elements in PTC-Orbit

Time dependent elements can be defined in PTC-Orbit by reading in tables with the variation of the e.g. quadrupole strength and tables for the RF system.
For the definition the file `FINAL_SETTINGS.TXT` , which contains:

```
  0  1.000000000000000E-003 0 F
  1
  0.000000000000000E+000
```

The first line is from left to right: time - time unit (here `1.000000000000000E-003` for ms) - number of turn - flag (set to false, not yet implemented effect). The other two lines I don't know.

3.1 Definition in PTC-orbit

To include the time dependence, on calls a ptc script, e.g. `PSB_PTC_TOTAL_RAMP_RFTable.ptc`:

```
ReadPTCscript("PSB_PTC_TOTAL_RAMP_RFTable.ptc");
```

Where the `PSB_PTC_TOTAL_RAMP_RFTable.ptc` contains the following:

```
select layout
1
+time
+cavity
SET ORBIT RAMPING
set orbit state
SET ORBIT RESTORE
 f f
 ramp
 BR.KSW2L1  "./Tables/KSW2L1.TXT"       1
 ramp
 QDE  "./Tables/QDE.TXT"       1
initialize cavity
1
```

```
BR.C02,  "./Tables/RF_DOUBLEHARM_NOACC.TXT"
INITIAL TIME IN MY UNITS
energize ORBIT lattice
set orbit acceleration
return
```

Explanation of the different commands:

- `+time` enables the time tracking
- `+cavity` ?
- `SET ORBIT RAMPING` probably enables ramping
- To keep the lattice updated without returning to the original:

  ```
  SET ORBIT RESTORE
   f f
  ```

- for ramping the quadrupoles/dipoles one has to specify the name (as called in madx) and then the table (here `QDE.TXT`) with the ramping parameters. Here the command for a quad:

  ```
   ramp
   QDE  "./Tables/QDE.TXT"      1
  ```

  for the definition of the main cavity on has to call the following command:

  ```
  initialize cavity
  1
  BR.C02,  "./Tables/RF_DOUBLEHARM_NOACC.TXT"
  ```

  where `initialize cavity` obviously initializes the cavity, `1` is the number of cavities (here `1`) and the table for the time dependence is given in `RF_DOUBLEHARM_NOACC.TXT`.

- `INITIAL TIME IN MY UNITS` ??
- to set the right energy for the actual time including cavities one has to call the last command:

  ```
  energize ORBIT lattice
  set orbit acceleration
  return
  ```

  where the commands are self explaining.

3.2 Syntax of the Tables

In the previous section 3.1 we showed how to define time dependent elements. Here we want to explain the tables.

3.2.1 Quadrupoles and Dipoles

Which table is called for which element is defined by for e.g. a quadrupole:

```
 QDE   "./Tables/QDE.TXT"       1
```

or a dipole:

```
 BR.KSW2L1   "./Tables/KSW2L1.TXT"       1
```

The tables have the following syntax. We start with the dipole example `KSW2L1.TXT` :

```
2 1 1
1
0.0     .01125474948    .000000
0.0001     .000000     .000000
```

The different numbers are explained in Fig. 3.2.1: and a typical quadrupole table is e.g. `QDE.txt`:

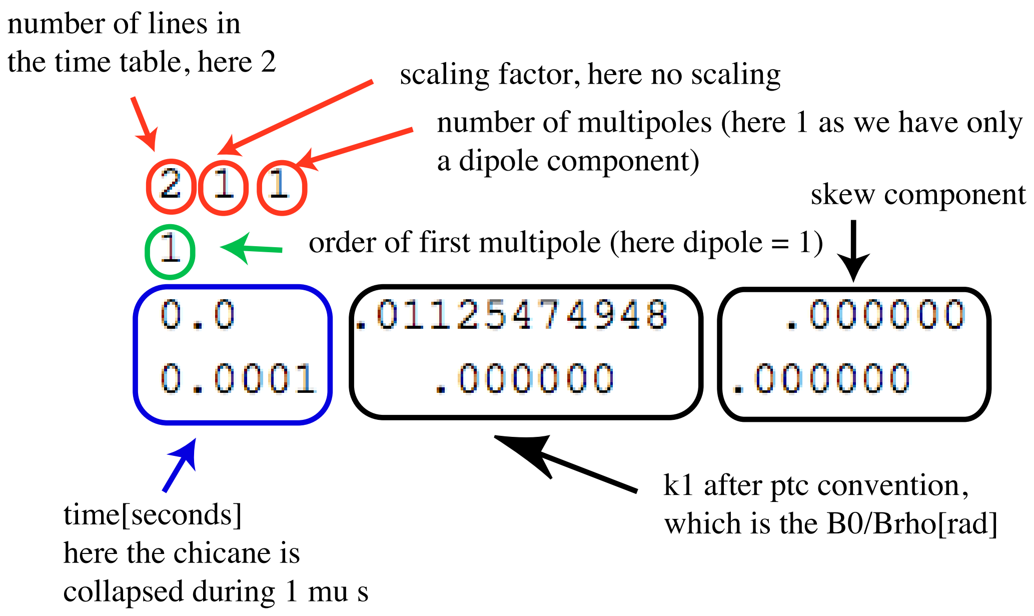


Figure 3.2.1: Table for a dipole

```
2 1 1
2
0.0     -0.7736596834 0.000000
0.0005  0.000000      0.000000
```

The different numbers are explained in Fig. 3.2.2:

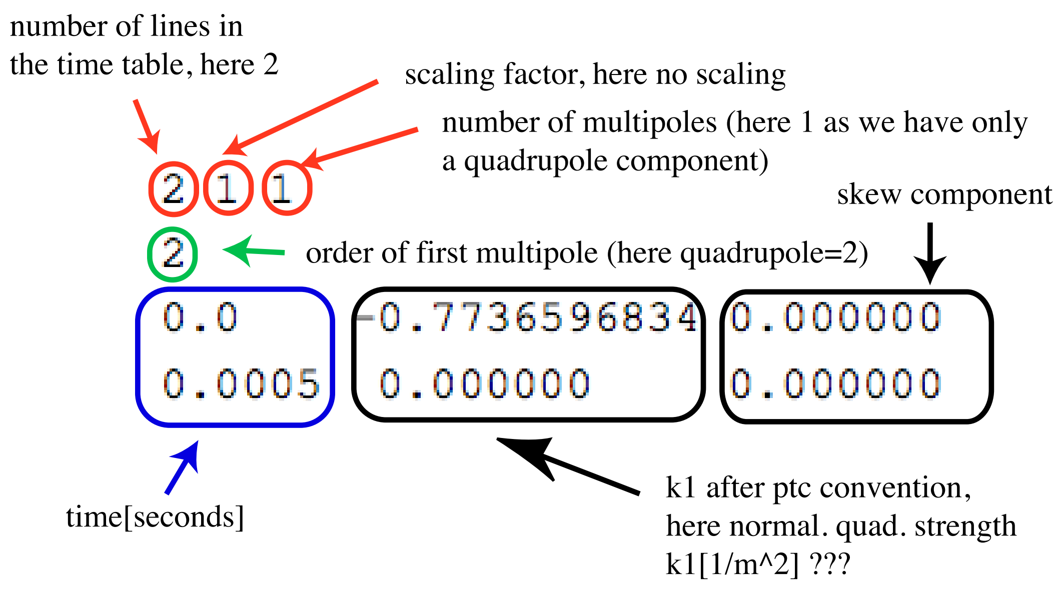

number of lines in the time table, here 2
scaling factor, here no scaling
number of multipoles (here 1 as we have only a quadrupole component)
skew component
2 1 1
2
order of first multipole (here quadrupole=2)
0.0
0.0005
-0.7736596834
0.000000
0.000000
0.000000
time[seconds]
k1 after ptc convention, here normal. quad. strength k1[1/m^2] ???

Figure 3.2.2: Table for a quadrupole.

3.2.2 Cavities

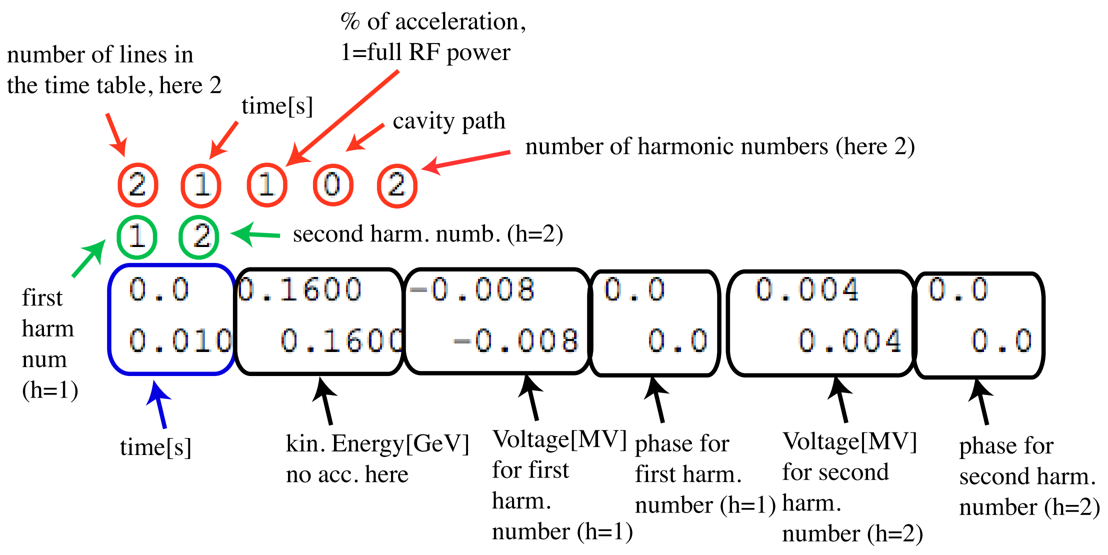
Which table is called for the cavity is defined by:

```
BR.C02,  "./Tables/RF_DOUBLEHARM_NOACC.TXT"
```

`RF_DOUBLEHARM_NOACC.TXT` then contains the time dependence:

```
2  1  1  0  2
1  2
0.0  0.1600  -0.008   0.0    0.004   0.0
0.010  0.1600  -0.008   0.0    0.004   0.0
```

The different numbers are explained in Fig. 3.2.3:



% of acceleration, 1=full RF power
number of lines in the time table, here 2
time[s]
cavity path
number of harmonic numbers (here 2)
2 1 1 0 2
1 2
second harm. numb. (h=2)
first harm num (h=1)
0.0 0.1600 -0.008 0.0 0.004 0.0
0.010 0.1600 -0.008 0.0 0.004 0.0
time[s]
kin. Energy[GeV] no acc. here
Voltage[MV] for first harm. number (h=1)
phase for first harm. number (h=1)
Voltage[MV] for second harm. number (h=2)
phase for second harm. number (h=2)

Figure 3.2.3: Table for a double harmonic RF cavity.

Chapter 4

Space Charge Simulation Methods

The placement of the space charge nodes with PTC-Orbit is explained in chapter 2.2, here we only want to summarize a rule of thumb or starting point for the simulation parameters. Some information about the routines can be found in the Source code of orbit Code, specifically in the files `TSpaceCharge.cc`, `SpaceCharge3D.cc` and `LSpaceCharge.cc`.

4.1 Pairwise Sum:

The PairWise (PW) sum algorithm calculates the field acting on each particle by summing up the field of all other particles treated as point charges. Writing down the equations the sum can be arranged in pairs. The algorithm is very slow but simple. The command syntax is:

```
addPWTransSCSet(...);
```

The computation time goes with N^2, where N is the number of macroparticles.

4.2 Brute Force Model:

The Brute Force (BF) algorithm uses a mesh as illustrated in Fig. 4.3.1 and calculates the field acting on each particle by summing up the field from each mesh point on the particle. The command syntax is:

```
addBFTransSCSet(...);
```

The computation time goes with N^4_{mesh}, where N_{mesh} is the number of meshpoints.

4.3 2 1/2 D Model:

The 2 1/2 D model model first projects the whole distribution in the transverse plane and then calculates the space charge force for each grid point. The kicks are then weighted according to the longitudinal distribution and applied to the macroparticles. That's why its called 2 1/2 D. Two routines exist in orbit for the 2 1/2 D model:

- `addFFTTransSCSet`- "Force based transverse space charge":The grid is adjusted to 2 sigma of the beam size and no boundary conditions are taken into account. If the beam size is relatively small compared to the aperture, this is the routine of your choice. The command is:

  ```
  addFFTTransSCSet(nxBins,nyBins,eps,nMacroTSCMin);
  ```

where `nxbins` is the number of hor. and `nybins` the number of vert.bins, eps the smoothing parameter (good number is 1×10^{-6}) and `nMacroTSCMin` the minimum number of macros to have before using the kick. The field is calculated using a Fast Fourier Transform (FFT). For a most efficient calculation the number of bins (`nbins`) should be a power of 2 (`nxbins`= 2^n and `nybins`= 2^m).

The computation time goes with $2N_{\text{mesh}}$ or $\log(N_{\text{mesh}})$, where N_{mesh} is the number of meshpoints.

- `addPotentialTransSCSet`- "Potential based transverse space charge": The grid is kept constant and is extended over the full beampipe. The dimension of the beampipe is defined over the parameters `BP*` . `BP*` has different meanings for different beam pipe shapes (parameter `Type`). The boundary conditions are taken into account taking the beampipe as boundary. The command syntax is:

```
addPotentialTransSCSet(nxBins, nyBins, eps, Type,BP1,BP2,BP3,
     BP4,bPoints, bModes, gFact,nMacroTSCMin);
```

  where `nxbins` is the number of hor. and `nybins` the number of vert.bins, eps the smoothing parameter (good number is 1×10^{-6} and `nMacroTSCMin` the minimum number of macros to have before using the kick.

The longitudinal space charge nodes are defined by the command `addFFTLSpaceCharge(...)`, which defines the number of longitudinal bins. The command syntax is:

```
addFFTLSpaceCharge(snLong,iLNode, ZImped, b_a, useAvg,
                nMacroLSCMin, useSpaceCharge)
```

where `snLong` is the name for the node, `iLNode` the index order used to place the node (Integer), `ZImped` a vector containing the Z/n wall longitudinal coupling impedance (Complex vector, 0 for no impedance), b_a the wall to beam radius ratio (Real), `useAvg` a flag (Integer) indicating whether kick values interpolated from the phase bins should be used ("1" if an interpolation scheme is used and "0" if a kick is calculated for each macro-particle's explicit phase), `nMacroLSCMin` a minimum number of macro-particles required before the calculation is done (typically setting this number to 100 prevents spurious numerical events).
Now some rules of thumb for the number of space charge nodes valid for `addFFTTransSCSet` and `addPotentialTransSCSet`:

- **Number of longitudinal space charge nodes:** It is mostly sufficient to place only one longitudinal space charge node. The number of bins (`nLongBins`) should be 2^n as the field is again calculated by a Fourier transform.

- **Number of transverse space charge nodes:** Approx 10 space charge nodes per betatron period is sufficient for most space charge calculations,so:

$$N_{\text{SC nodes}} \geq 10/\text{betatron period} \tag{4.3.1}$$

  If the variation of the beta functions is large, more space charge nodes are needed.

  Example: PS with a circumference of 600 m and $Q_{x,y} \approx 6$. Then one betatron period has a length of $L_{\text{betatron period}} = 100$ m. The minimal distance between space charge nodes is then $L_{\text{max, SC}} = L_{\text{betatron period}}/10 = 10$ m

- **Mesh for transverse space charge:** A mesh is put over the distribution with the boundaries at 2σ of the beam size. One should at least have 10 macroparticles per meshpoint. This means that for a mesh of 100×100 one needs at least 100000 macroparticles. To ensure the 10 macroparticles per meshpoint, one should take 500000 macroparticles as not the whole beam lies inside 2σ. The gird is then automatically created by orbit. The number of meshpoints is specified by `nxBins` respectively `nyBins`. The mesh is illustrated in Fig. 4.3.1. The mesh is usually chosen to be quadratic so `nxBins=nyBins`. The number of bins should be 2^n if a FFT method is used.

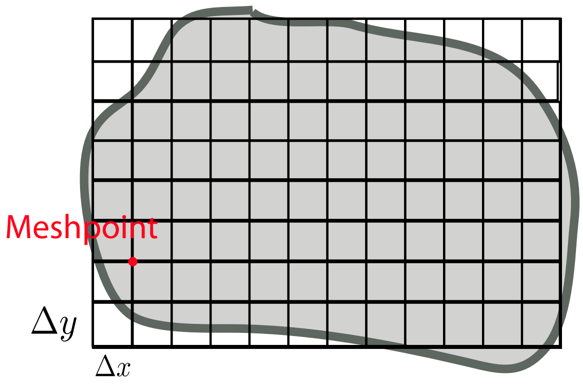


Figure 4.3.1: Mesh for Space Charge Simulations

4.4 3 D Model:

Computationally very heavy as it takes the full 3 D distribution into account. There are two routines `addSpCh3D_BFSet` (where `BF` stands for bunching factor) and `addSpCh3D_FFTSet` .

Bibliography

[1] Haruo Yoshida.
Construction of higher order symplectic integrators.
Physics Letters A, 1990.

[2] F Schmidt, E Forest, and E McIntosh.
Introduction to the polymorphic tracking code: Fibre bundles, polymorphic taylor types and exact tracking.
Technical Report CERN-SL-2002-044-AP. KEK-REPORT-2002-3, CERN, Geneva, Jul 2002.

[3] Dan T Abell.
Ptc library user guide.
Technical report, July 2011.