EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN – BE DEPARTMENT

Werner.Herr@cern.ch

CAS 2013

Problems and solutions of the exercises in the optics course at the CAS 2013 on Trondheim.

W. Herr, BE Department, CERN, 1211-Geneva 23

Abstract

In this document we describe possible solutions for the exercises during the Optics Course at the Advanced Level CERN Accelerator School 2013 on Trondheim.

Geneva, Switzerland

1st October 2013

1 Introduction

At the CERN Accelerator School 2013 Trondheim, a course was held on optics design. This course was intended as an introduction for accelerator physicists who want to start the design of accelerator optics as well as for people from other fields who would like to get a basic knowledge of the principles of accelerator design.

The aim was that the participants design a realistic accelerator optics, following plenary lectures on lattice cells [3], insertions [4] and imperfections [5, 6]. This should be done following a series of steps in form of exercises the participants had to solve and implement in an accelerator design program (MAD-X).

After an introduction to MAD-X [7, 8], the participants had to design a periodic machine with well defined properties. In the second part, correction schemes for chromaticity, closed orbit etc. had to be defined and implemented. In the third part, a straight section with a dispersion suppressor and a low β-insertion was inserted into the originally periodic structure. The tuneability of this lattice and further improvements to increase the flexibility were studied and implemented where required.

In this paper we discuss the proposed solutions and show some of the techniques used in the design of accelerator lattices. Some references in this note correspond to example files and the solutions which will be made available on the web [10] and on a CD-ROM [11].

The proposed solutions are not always the most compact or simplest, rather they have been chosen to demonstrate the main issues and design concepts.

2 Exercise 1

2.1 Problem:

Design a machine for protons at a momentum of 20 GeV/c with the following basic parameters:

- Circumference = 1000 m,
- Quadrupole length L_{quad} = 3.0 m,
- 8 FODO cells.
- Dipole length is 5 m, maximum field is 3 T

Apply the knowledge from previous lectures at this school and define a lattice cell according to the boundary conditions (position of dipole magnets and quadrupoles) and find the optics (strength of dipoles and quadrupoles) so that $\beta_{max} \equiv \hat{\beta}$ is around 300m. Implement it into MAD format using thin lenses for all elements and verify the calculations.

2.2 Solution:

Set up a FODO cell of length L like in Fig.1 [3]. The cell length is L=1000 m/8 = 125 m. For simplicity

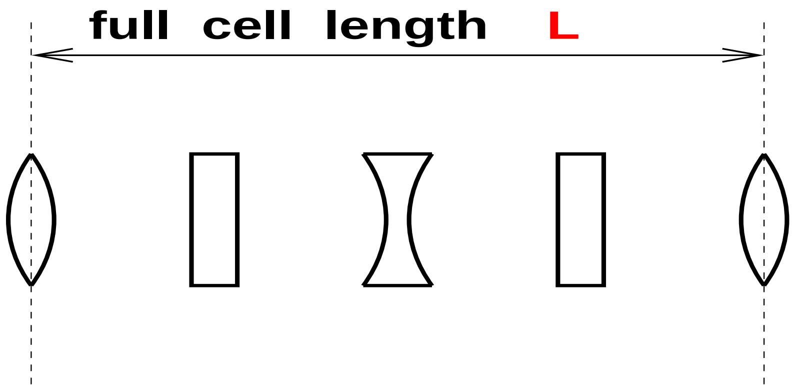

Figure 1: Schematic view of a symmetric FODO cell.

the elements can be distributed equally spaced inside the cell.

The maximum integrated field of a dipole is 15 Tm. At 20 GeV/c this corresponds to a deflection angle of 0.225 [3]. To get a total angle of 2π, we need 2π/0.225 = 27.9 $\rightarrow$ 28 dipoles. For 8 cells, we therefore need 32 dipoles. We need 8 focusing and 8 defocusing quadrupoles and 32 bending dipoles, i.e. 4 dipoles per cell. The bending angle of the dipoles must be $2\pi/32$ = 0.196349541. Part of the lattice is shown in Fig.2 using MAD convention, i.e. upward and downward boxes indicate focusing and defocusing quadrupoles respectively. Vertical lines show dipole magnets. For a FODO cell we get

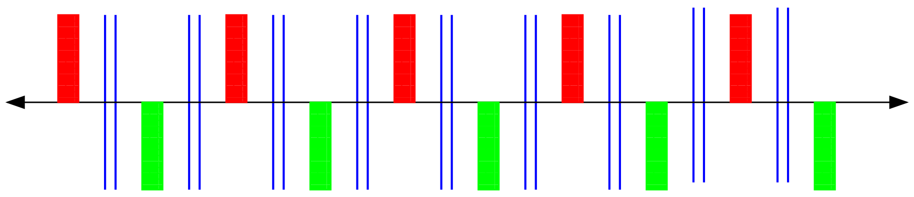

Figure 2: Schematic layout for regular FODO lattice. Symbols indicate quadrupoles and dipoles.

the expression for the phase advance $\phi/2$ per half cell:

$$sin(\phi/2) \;=\; \frac{L_{1/2}}{f_{1/2}} \tag{1}$$

where $L_{1/2}$ is the length of the half cell (62.5 m) and $f_{1/2}$ the focal length of the half cell. the expressions for the maximum and minimum betatron function β are written:

$$\hat{\beta} \;=\; f_{1/2} \cdot \frac{1 + sin(\phi/2)}{cos(\phi/2)} \;=\; L_{1/2} \cdot \frac{1 + sin(\phi/2)}{sin(\phi/2)cos(\phi/2)} \tag{2}$$

and

$$\check{\beta} \;=\; f_{1/2} \cdot \frac{1 - sin(\phi/2)}{cos(\phi/2)} \;=\; L_{1/2} \cdot \frac{1 - sin(\phi/2)}{sin(\phi/2)cos(\phi/2)} \tag{3}$$

For $\hat{\beta} \;=\; 300\, m$ and $L_{1/2} \;=\; 62.5\, m$ we get for $\phi/2 \;=\; 16.07066^o$.
Therefore:

$$f_{1/2} \;=\; 62.5/sin(\phi/2) \;\; = \; 225.776\, m \quad \rightarrow \quad f \;=\; 112.888\, m \tag{4}$$

this gives for the focusing strength k_1 (positive for focusing and negative for defocusing quadrupole):

$$k_1 \;=\; \frac{1}{L_{quad} \cdot f} \;=\; \pm\; 0.002952775 \tag{5}$$

2.2.1 Implementation with thick elements

This cell can be implemented in MAD as (EX1/ex1_thick.seq):

```
circum=1000.0;
ncell = 8;
lcell = circum/ncell;
lq = 3.00;
lq2 = lq/2.;

// forces and other constants;
// element definitions;

// define bending magnet as multipole
!mb: multipole, knl={2.0*pi/(4*ncell)};
mb: sbend, l=5.0,  angle=2.0*pi/32.;



mq: quadrupole, l=lq;
qf: mq, k1:=kqf;
qd: mq, k1:=kqd;
kqf =  .295278e-2;
kqd = -.295278e-2;

// sequence declaration;
cascell2: sequence, refer=centre, l=circum;
start_machine: marker, at = 0;
!
   n = 1;
   while (n < ncell+1) {
   qf: qf,   at=(n-1)*lcell + lq2;
   mb: mb,   at=(n-1)*lcell+0.15*lcell + lq2;
   mb: mb,   at=(n-1)*lcell+0.35*lcell + lq2;
   qd: qd,   at=(n-1)*lcell+0.50*lcell + lq2;
   mb: mb,   at=(n-1)*lcell+0.65*lcell + lq2;
   mb: mb,   at=(n-1)*lcell+0.85*lcell + lq2;
!
   n = n + 1;
}
end_machine: marker at=circum;
endsequence;
```

In this cell, the cell starts at the entry of the focusing quadrupole and ends at the entry of the focusing quadrupole of the next cell. It is repeated $ncell$ times to make the ring of 1000 m circumference. The precise relative position of the elements inside a cell can be chosen to fit the space requirements. The values in the proposed example are somewhat arbitrary.

The bending magnets are defined as multipoles and their first component is the bending angle.

2.2.2 Implementation with thin elements

An alternative, probably simpler implementation would be using only thin elements like in (EX1/ex1_thin.seq).

In later examples, in particular the more demanding exercises with insertions, we shall use the thin lens approximation.

```
circum=1000.0;
ncell =  8; // Number of cells
lcell = circum/ncell;
lq = 3.00; // Length of a quadrupole

// element definitions;

// define bending magnet as multipole
// we have 4 bending magnets per cell
mb: multipole,knl={2.0*pi/(4*ncell)};

// define quadrupoles as multipoles
qf: multipole,knl={0,0.295278e-2*lq};
qd: multipole,knl={0,-0.295278e-2*lq};

// sequence declaration;
cascell1: sequence, refer=centre, l=circum;
start_machine: marker, at = 0;
!
   n = 1;
   while (n < ncell+1) {
   qf: qf,   at=(n-1)*lcell;
   mb: mb,   at=(n-1)*lcell+0.15*lcell;
   mb: mb,   at=(n-1)*lcell+0.35*lcell;
   qd: qd,   at=(n-1)*lcell+0.50*lcell;
   mb: mb,   at=(n-1)*lcell+0.65*lcell;
   mb: mb,   at=(n-1)*lcell+0.85*lcell;
!
   n = n + 1;
}
```

```
end_machine: marker at=circum;
endsequence;
```

Now the cells start and end in the *centres* of the focusing quadrupoles. Both declaration can be executed with the following MAD commands (EX1/ex1_thin.madx):

```
TITLE, 'CAS2013 first exercise';
call file="ex1_thin.seq";

\\ define the beam and its properties
Beam, particle = proton, sequence=cascell1, energy = 20.0,
          NPART=1.05E11, sige=      4.5e-4 ;

\\ define the desired output
use, sequence=cascell1;
select,flag=twiss,column=name,s,x,y,mux,betx,
                         muy,bety,dx,dy;

\\ execute the TWISS command
twiss,save,centre,file=twiss.out;
plot, haxis=s, vaxis=betx, bety, colour=100;

stop;
```

The results are printed to a file *twiss.out* and the graphical output is found in *madx.ps*. Below one finds the Twiss summary printed at the end of the Twiss calculations. The alternative definition of the quadrupoles was used for this run and will be used in all later examples. MAD gives a maximum β ($\hat{\beta}_x$ and $\hat{\beta}_y$) in both planes of 300 m, in good agreement with the desired parameters.

```
++++++ table: summm

        length             orbit5               alfa            gammatr
          1000                 -0        1.989271492       0.7090109959

            q1                dq1            betxmax              dxmax
  0.7142528897      -0.7344010503        299.9996549        365.0168476

         dxrms             xcomax             xcorms                 q2
   321.4860068                  0                  0       0.7142528897

           dq2            betymax              dymax              dyrms
 -0.7344010503        299.9996549                  0                  0

        ycomax             ycorms             deltap            synch_1
             0                  0                  0                  0

       synch_2            synch_3            synch_4            synch_5
             0                  0                  0                  0
```

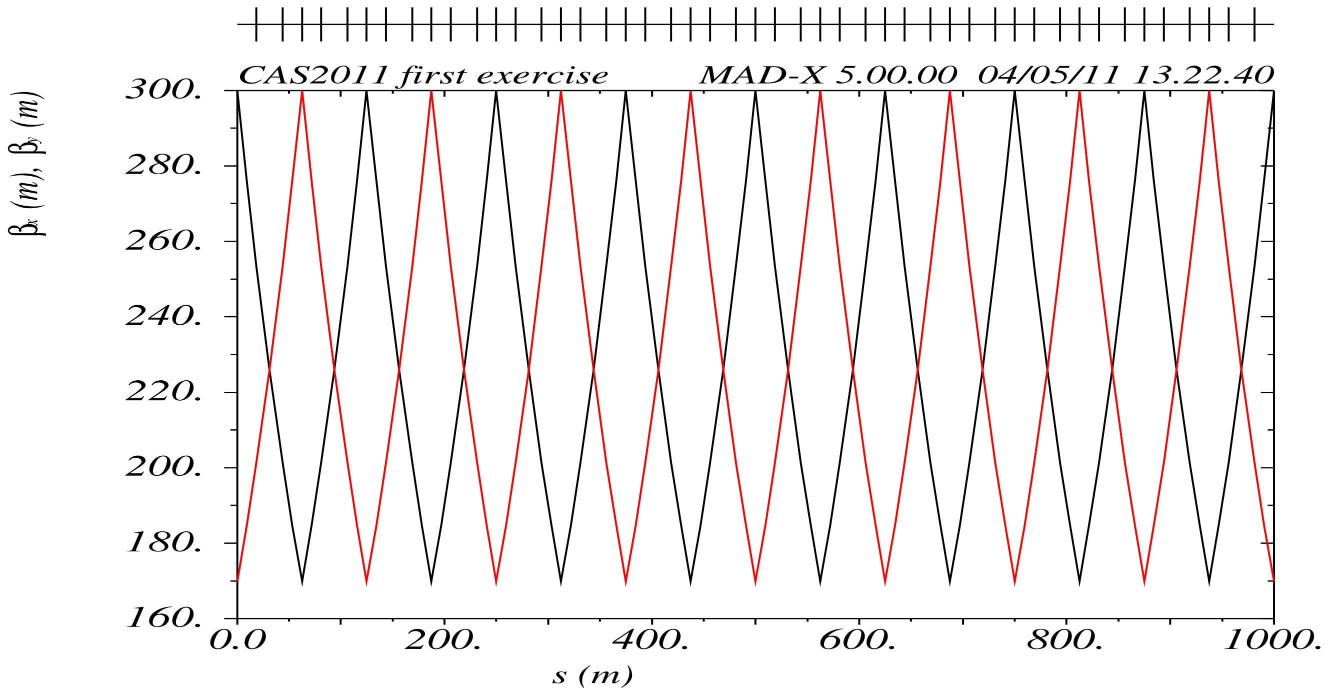

Figure 3: β-functions for symmetric FODO cell. Maximum β is 300 m.

2.2.3 Checking the closure of the ring

Whether the ring is closed or not, can be checked with the **survey** command of the type:

Survey, file=survey.cas;

following all other commands. This will compute the geometry of the machine. For a closed machine the absolute coordinates at the beginning and the end of the ring should be identical, the variable **theta** in the output file should be equal to 2π. The geometries of a closed and an unclosed ring are shown in Fig.4. These figures can be obtained from the generated file $survey.cas$ by plotting the variables x

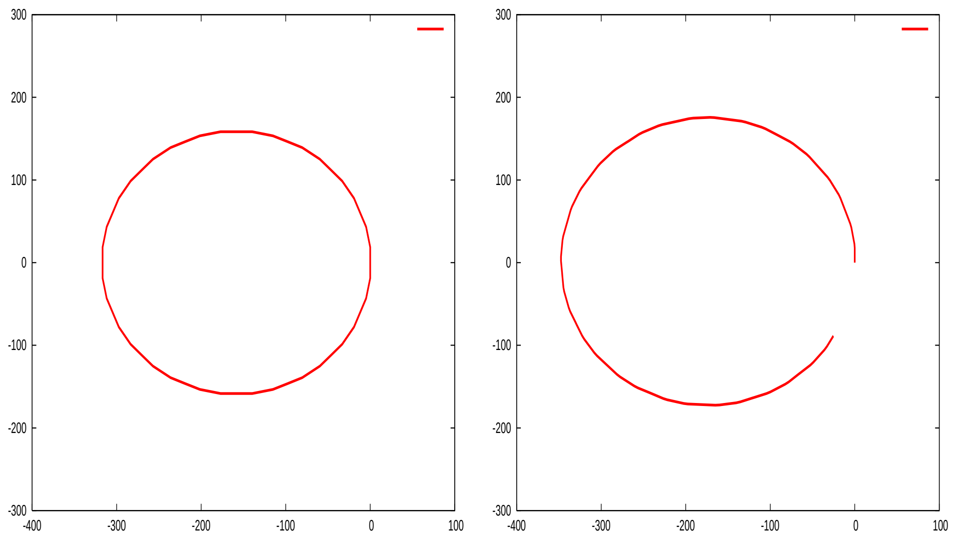

Figure 4: Geometry of a closed and an unclosed ring.

versus z.

3 Exercise 2

3.1 Problem:

Start with the lattice from exercise 1 and modify it so the maximum betatron function $\hat{\beta}$ is around 100 m. The circumference must not be changed.

3.2 Solution:

For the maximum of the β-function we had:

$$\hat{\beta} \;=\; f_{1/2} \cdot \frac{1 + sin(\phi/2)}{cos(\phi/2)} \;=\; L_{1/2} \cdot \frac{1 + sin(\phi/2)}{sin(\phi/2)cos(\phi/2)} \tag{6}$$

For a given half cell length $L_{1/2}$ the smallest $\hat{\beta}$ can be found be differentiating the equation and one obtains:

$$sin^3(\phi/2) + 2sin^2(\phi/2) - 1 = 0 \tag{7}$$

The solution is $\phi/2 \;=\; 38.17^o$ and therefore the minimum $\hat{\beta}$ is 3.3302 $\cdot$ $L_{1/2}$. For the present cell length this gives: $\hat{\beta}_{min}$ = 206.375 m, i.e. we cannot reach this value with the present lattice.

The only way to decrease $\hat{\beta}_{min}$ is to reduce the cell length, i.e. increase the number of cells for a given circumference.

Starting from the solution of the previous exercise, we increase the number of cells to 20 and get $L_{1/2}$ = 25. With the same procedure as before we get for the phase advance per half cell: $\phi/2 \;=\; 21.56^o$. With this phase advance we need $f_{1/2}$ = 68.0 m and f = 34.0 m. This gives focusing strengths of $k_1 \;=\; \pm$ 0.0098.

After modifying the previous sequence with ncell=20 and the quadrupole strength, we get the desired $\hat{\beta}_{min}$ = 100 m.

Using the previous MAD input and with the necessary modifications (ncell = 20, $k_1 \;=\; \pm$ 0.0098, EX2/ex2_thin.seq and EX2/ex2_thin.madx) we get the output below.

```
++++++ table: summ

          length            orbit5              alfa           gammatr
            1000                -0      0.1789932276       2.363642013

              q1               dq1           betxmax             dxmax
     2.395724244      -2.518384148       100.0267292       34.45748961

           dxrms            xcomax            xcorms                q2
      29.0343083                 0                 0       2.395724244

             dq2           betymax             dymax             dyrms
    -2.518384148       100.0267292                 0                 0

          ycomax            ycorms            deltap           synch_1
               0                 0                 0                 0

         synch_2           synch_3           synch_4           synch_5
               0                 0                 0                 0
```

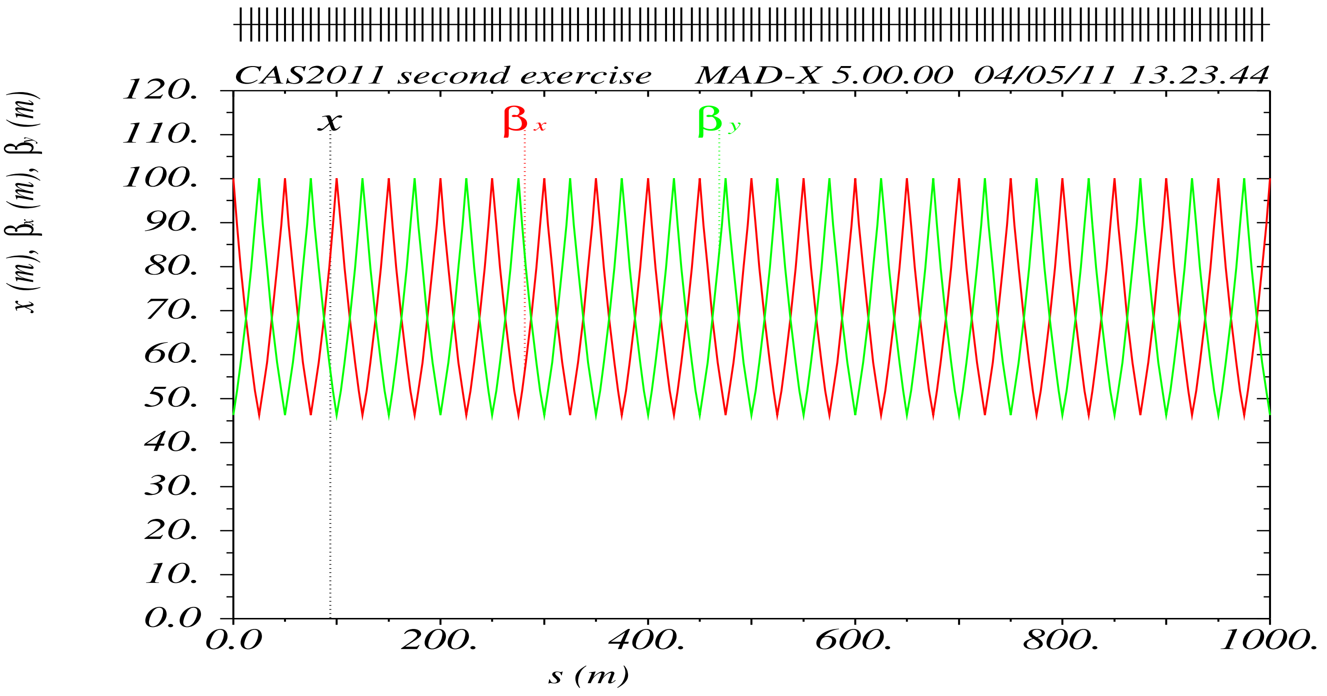

Figure 5: β-functions for symmetric FODO cell. Maximum β is 100 m.

4 Exercise 3

4.1 Problem:

Start with the lattice from exercise 2 and modify it so you can correct the chromaticity for both planes to zero. Try first to calculate approximately the required strengths. Implement your correction scheme in your previous MAD files and verify your calculation. Use MAD to compute the exact strengths required by matching the global parameters Q'_x and Q'_y (in MAD names: DQ1 and DQ2). Compare the results with your calculations.

4.2 Solution:

From exercise 2 we know that the horizontal and vertical tunes are $\phi \cdot ncell$ = 21.56$^o \cdot 2 \cdot ncell$ = 2.396. The natural chromaticity is [3]:

$$Q' \;=\; -\frac{1}{\pi} tan(\phi/2) \cdot ncell \;=\; -2.515$$

The precise numbers we can get from the MAD summary table.

For the correction we need an element where the $focusing\ length\ f$ is amplitude dependent, e.g. a sextupole where the focusing length (i.e. the derivative of the force) depends linearly on the amplitude. Furthermore, we have to $sort$ the particles in amplitude according to their momentum. This can be done with horizontal dispersion. Therefore we put sextupoles in regions $with$ horizontal dispersion. (Remark: sextupoles in regions $without$ dispersion can be used to deliberately control third order resonances without affecting the chromaticity).

In order to correct both, the horizontal and vertical chromaticities to zero, we need two independent groups of sextupoles. A natural choice is next to the quadrupoles. We shall call the two groups SF and SD, when they are next to the focusing or defocusing quadrupoles. We show this schematically by

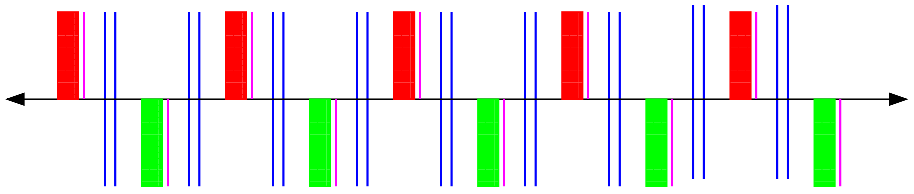

Figure 6: Schematic layout for regular FODO lattice with sextupoles for chromaticity corrections.

modifying Fig.2 by inserting symbols for sextupoles and get Fig.6. The tune change due to a horizontal displacement x in a sextupole is easily written:

$$\Delta Q \;=\; -\frac{1}{4\pi}\int \beta k_2 \cdot x\, ds$$

When the horizontal displacement x is due to a momentum deviation $\frac{\delta p}{p}$ at a place with dispersion D_x we use:

$$x \;=\; D_x \frac{\delta p}{p}$$

to get:

$$\Delta Q \;=\; -\frac{1}{4\pi}\int \beta k_2 \cdot D_x \frac{\delta p}{p}\, ds \;=\; -\frac{1}{4\pi}\int \beta k_2 D_x ds \cdot \frac{\delta p}{p} \;=\; Q' \cdot \frac{\delta p}{p}$$

For thin lenses the chromaticity change for two groups of sextupoles is in the horizontal plane:

$$\Delta Q'_x \;=\; -\frac{1}{4\pi}\sum_{F\ sextupole} k_2^F l_s D_x^F \beta_x^F + \frac{1}{4\pi}\sum_{D\ sextupole} k_2^D l_s D_x^D \beta_x^D$$

or for a periodic structure

$$\Delta Q'_x \;=\; -\,\frac{1}{4\pi}\cdot ncell\cdot(k_2^F l_s D_x^F \beta_x^F - k_2^D l_s D_x^D \beta_x^D)$$

For $\Delta Q'_y$ the horizontal and vertical β-functions have to be exchanged in the formulae.

$$\Delta Q'_y \;=\; -\,\frac{1}{4\pi}\cdot ncell\cdot(-k_2^F l_s D_x^F \beta_y^F + k_2^D l_s D_x^D \beta_y^D)$$

These two equations form a system of equation with the two unknowns k_2^F and k_2^D when the desired $\Delta Q'$ changes are given.

From the MAD output for the previous exercise we know the Twiss parameters at the sextupoles (Tab.1). It is clear that the dispersion function at the position of the sextupoles should be large to minimize the required strengths. Furthermore, at the position of the two groups, the horizontal and vertical β-functions should be as different as possible. Therefore our choice to place the sextupoles next to the quadrupoles was a good choice.

	D_x	β_x	β_y
at QF:	34.457	100.027	46.265
at QD:	23.760	46.265	100.027

Table 1: Twiss parameters at lattice sextupoles.

Since we do not consider collective effects in this exercise, we want to correct the chromaticities to zero, therefore we take:

$$\Delta Q'_x \;=\; -\,Q'_x \quad\text{and}\quad \Delta Q'_y \;=\; -\,Q'_y$$

We solve the above system of equations with the numbers from the Tab.1 and we find:
$k_2^F l_s \;=\; 0.017041/ncell$ and $k_2^D l_s \;=\; -\,0.024714/ncell.$

To our MAD description from the previous exercise we add the following lines (EX3/ex3.seq):

```
// define the sextupoles as multipole
lsex = 0.00001; // dummy length, only used in the sequence;
ksf :=  +0.017041/20.0;
ksd :=  -0.024714/20.0;

// ATTENTION: must use knl:=  and NOT knl= to match later !
msf: multipole, knl:={0,0,ksf};
msd: multipole, knl:={0,0,ksd};
```

and change the definition of our cell to add the sextupoles:

```
    n = 1;
    while (n < ncell+1) {
    qf: qf,      at=(n-1)*lcell;
    msf: msf,    at=(n-1)*lcell + lsex/2.0;
    mb: mb,      at=(n-1)*lcell+0.15*lcell;
```

```
   mb: mb,      at=(n-1)*lcell+0.35*lcell;
   qd: qd,      at=(n-1)*lcell+0.50*lcell;
   msd: msd,    at=(n-1)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,      at=(n-1)*lcell+0.65*lcell;
   mb: mb,      at=(n-1)*lcell+0.85*lcell;
!
   n = n + 1;
}
```

This gives the following global parameters:

```
++++++ table: summm

          length             orbit5               alfa            gammatr
            1000                 -0       0.1789932276        2.363642013

              q1                dq1            betxmax              dxmax
     2.395724244    -0.006308569499        100.0267292        34.45748961

           dxrms             xcomax             xcorms                 q2
     29.17402399                  0                  0        2.395724244

             dq2            betymax              dymax              dyrms
 -0.006122732322        100.0267292                  0                  0

          ycomax             ycorms             deltap            synch_1
               0                  0                  0                  0

         synch_2            synch_3            synch_4            synch_5
               0                  0                  0                  0
```

The chromaticities are corrected to values smaller than 0.01 for both planes. We get for Q'_x (dq1) = -6.3 $\times$ 10^{-3} and Q'_y (dq2) = -6.1 $\times$ 10^{-3}.

To get the exact values for the sextupole strengths, MAD can be used to MATCH these values. An example is given below (EX3/ex3.madx):

```
TITLE, s='MAD-X test';
call file="ex3.seq";
option,-echo;
// option,debug,-echo;

Beam, particle = proton, sequence=cascell3, energy = 20.0,
          NPART=1.05E11, sige=      4.5e-4 ;

use, period=cascell3;
```

```
match, sequence=cascell3;
   vary,name=ksf, step=0.00001;
   vary,name=ksd, step=0.00001;
   global,sequence=cascell3,DQ1=0.0;
   global,sequence=cascell3,DQ2=0.0;
   Lmdif, calls=10, tolerance=1.0e-21;
endmatch;

select,flag=twiss,column=name,s,betx,muy,bety,dx,dy;
twiss,save,centre,deltap=0.0,file=twiss2.out;
plot, haxis=s, vaxis=x, y;
stop;
```

The values found from the matching procedure are:

ksf = 8.54170E-04 (= 0.0170834/20) and
ksd = -1.23874E-03 (= -0.0247748/20)

We find that our approximate values are already very close ! On the level of the magnet and power converter precision as well as possible chromaticity measurement they are equivalent.

The output produced with these values (Twiss summary) is:

```
++++++ table: summ

          length            orbit5              alfa           gammatr
            1000                -0      0.1789932276       2.363642013

              q1               dq1           betxmax             dxmax
     2.395724244  5.974929442e-15       100.0267292       34.45748961

           dxrms            xcomax            xcorms                q2
     29.17402399                 0                 0       2.395724244

             dq2           betymax             dymax             dyrms
-1.177815379e-14       100.0267292                 0                 0

          ycomax            ycorms            deltap           synch_1
               0                 0                 0                 0

         synch_2           synch_3           synch_4           synch_5
               0                 0                 0                 0
```

It can be seen that the chromaticity is now practically zero.

5 Exercise 4

5.1 Problem:

Start with the lattice from the previous exercise and assume random misalignments of the quadrupoles of r.m.s. 0.1 mm in the horizontal and 0.2 mm in the vertical plane. Calculate the expected r.m.s. orbit and verify with MAD.

5.2 Solution:

Assuming the thin lens approximation, one can compute the r.m.s. orbit distortion. The deflection $\delta x'_i$ due to a (here horizontal) displacement Δx_i of a quadrupole is:

$$\delta x'_i = k_1 \cdot l_{quad} \cdot \Delta x$$

The orbit response x_j at a position j due to a single kick δx_i at position i is [5]:

$$x_j = \frac{\sqrt{\beta_i \beta_j}}{2\ sin(\pi Q_x)} cos(\pi Q_x - (\mu_i - \mu_j)) \cdot \delta x'_i = \frac{\sqrt{\beta_j}}{2\ sin(\pi Q_x)} cos(\pi Q_x - (\mu_i - \mu_j)) \cdot \sqrt{\beta_i} \delta x'_i$$

Here Q is the horizontal tune and μ the phase function around the machine. Taking the average of the squared displacements at all positions, and using an average $\beta = \bar{\beta}$:

$$< x^2 > = (\frac{\sqrt{\bar{\beta}}}{2\ sin(\pi Q_x)})^2 \cdot \frac{1}{2} \sum \bar{\beta} \delta x'^2_i$$

where we obtained a factor $\frac{1}{2}$ from averaging $\cos^2(\mu)$ and we write for x_{rms}:

$$\sqrt{< x^2 >} = x_{rms} = \frac{\sqrt{\bar{\beta}}}{2\sqrt{2}\ sin(\pi Q_x)} \sqrt{\sum \bar{\beta} \delta x'^2_i}$$

Assuming N identical cells with $N_d = 2N$ quadrupoles we can re-write the sum:

$$x_{rms} = \frac{\sqrt{\bar{\beta}}}{2\sqrt{2}\ sin(\pi Q_x)} \cdot \sqrt{N_d} \sqrt{\bar{\beta}} \delta x'_{rms}$$

$$x_{rms} = \frac{\sqrt{N_d}\ \bar{\beta}}{2\sqrt{2}\ sin(\pi Q_x)} \cdot \delta x'_{rms}$$

We use the expressions for thin lenses [3]:

$$\bar{\beta} = \frac{L_{cell}}{sin(\phi)}$$

and:

$$k_1 l_{quad} = \frac{4 sin(\phi/2)}{L_{cell}}$$

and obtain after some mathematics:

$$x_{rms} = \frac{\sqrt{N_d}}{\sqrt{2}\ sin(\pi Q_x) cos(\phi/2)} \cdot \Delta x'_{rms}$$

For a r.m.s. quadrupole displacement of 0.1 mm we can use the above expression and get as an estimate for $x_{rms} \approx$ 0.5 mm and $y_{rms} \approx$ 1.0 mm.

This can be verified with MAD by adding the lines:

```
eoption,add=false,seed=22021955;
select,flag=error,pattern="q.*";
ealign dx:=tgauss(3.0)*0.1e-3,dy:=tgauss(3.0)*0.2e-3;
```

This will assign random errors in the quadrupoles (all elements starting with 'q' are selected for the error assignment) following Gaussian distributions with r.m.s. values of 0.1 mm in the horizontal and 0.2 mm in the vertical plane. The SEED option allows to specify the starting point for the random number generator.

The MAD summary is given below and shows good agreement with the expectation. The x$_{rms}$ (xcorms) is 0.48 mm and y$_{rms}$ (ycorms) is 1.14 mm.

```
++++++ table: summ

          length            orbit5              alfa           gammatr
            1000                -0      0.1789930603       2.363643115

              q1               dq1           betxmax             dxmax
     2.395693776 1.299715695e-06       100.0394482       34.45831771

           dxrms            xcomax            xcorms                q2
     29.25808458  0.001235416833 0.0004802625523       2.395755835

             dq2           betymax             dymax             dyrms
-3.506407129e-06       100.0401998    0.002702833035     0.00114043919

          ycomax            ycorms            deltap           synch_1
  0.002699910864    0.001139215535                 0                 0

         synch_2           synch_3           synch_4           synch_5
               0                 0                 0                 0
```

A graphical output of the distorted orbits is given in Fig.7.

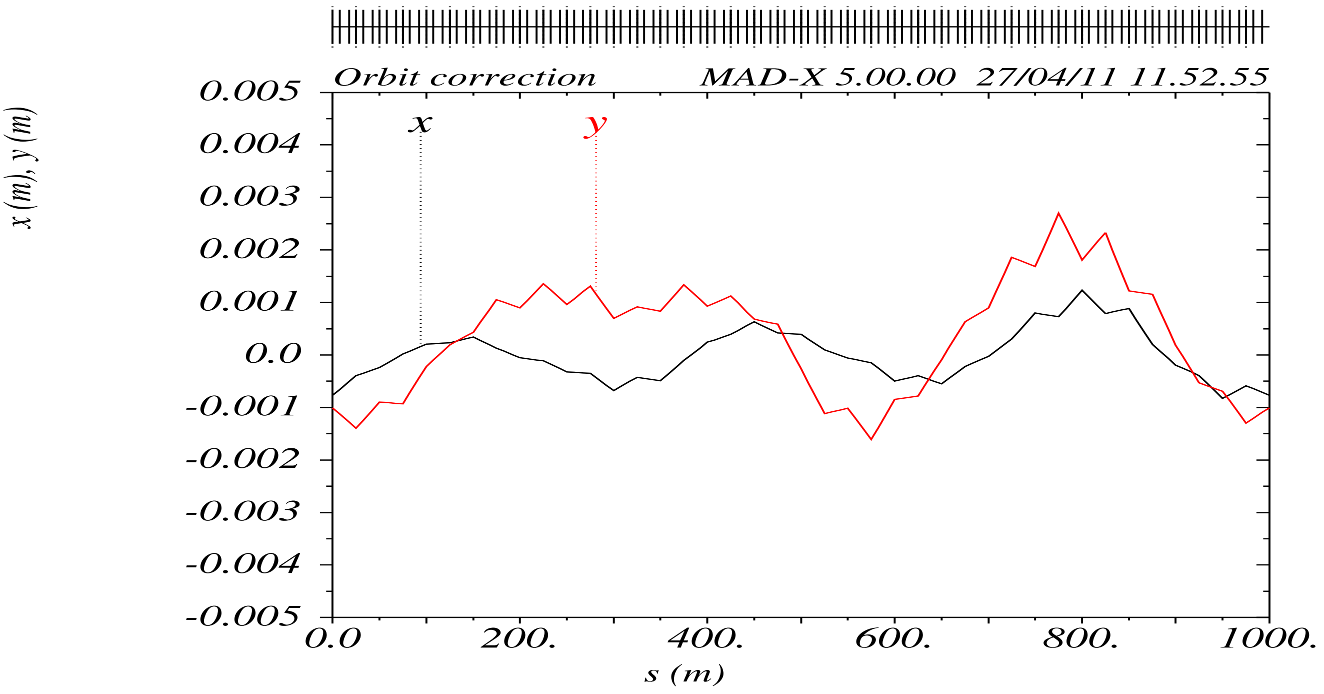

Figure 7: Horizontal and vertical orbit distortion, without correction.

6 Exercise 5

6.1 Problem:

Start with the lattice from the previous exercise and add the necessary equipment to be able to correct the closed orbit in both planes, (using MAD). Estimate first the maximum necessary strength of the orbit correctors assuming a maximum quadrupole displacement of 1 mm.

6.2 Solution:

The estimate of the necessary corrector strength is straightforward. The angular deflection of the beam passing through a quadrupole at an offset Δx is:

$$\delta x' \; = \; k_1 \cdot l \cdot \Delta x$$

Assuming the correctors next to the quadrupole and a maximum offset of 1 mm, we get about 30 μrad. To allow for some additional errors of monitors (scaling and positioning) or field errors from dipoles and a reserve we define the required strength as 150 μrad, which is a rather safe value. For the energy of 20 GeV of our beam this requires corrector magnets with an integrated field of $\approx$ 0.010 Tm, which can easily be provided (LHC orbit correctors provide 1.89 Tm).

To correct the orbit (with MAD or other orbit correction programs), we have to add orbit monitors and orbit correctors to the lattice. We decide to add one monitor and one corrector at each quadrupole. Monitors and orbit correctors in MAD can be defined to act only in one or in both planes (see MAD-X manual). While we want our monitors to measure the orbit in both planes, we define horizontal correctors at the focusing and vertical orbit correctors at defocusing quadrupoles. We have added them

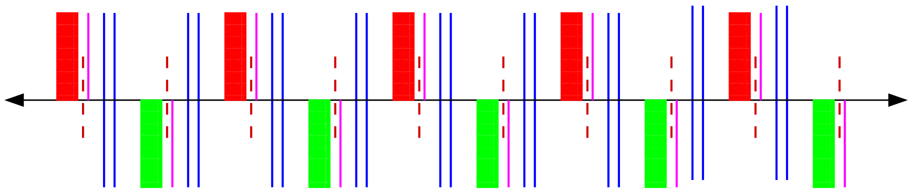

Figure 8: Schematic layout for regular FODO lattice with chromaticity sextupoles and closed orbit dipole correctors (dashed vertical lines).

in our schematic picture in Fig.8. This needs an extra (reserve) contribution to the corrector strength since not all imperfections can be corrected locally. We suggest the following sequence as one possibility (EX5/ex5.seq):

```
circum=1000.0;
ncell = 20;
lcell = circum/ncell;
lq = 3.00;
lsex  = 0.0001;
lbpm  = 0.0001;
lbpm2 = lbpm/2.;
lcor  = 0.0001;
lcor2 = lcor/2.;

// forces and other constants;
// element definitions;

// define orbit monitors and correctors
```

```
bpm: monitor, l = lbpm;
ch : hkicker, l = lcor;
cv : vkicker, l = lcor;

// define bending magnet as multipole
mb: multipole, knl={2.0*pi/(4*ncell)};

// define the quadrupoles as multipole
kqf =  .980000e-2;
kqd = -.980000e-2;
qf: multipole, knl={0,lq*kqf};
qd: multipole, knl={0,lq*kqd};

// define the sextupoles as multipole
// values from previous exercise
ksf :=   8.54169767E-04;
ksd :=  -1.23874055E-03;


// ATTENTION: must use knl:=  not knl= to match !
msf: multipole, knl:={0,0,ksf};
msd: multipole, knl:={0,0,ksd};

// sequence declaration;
cascell4: sequence, refer=centre, l=circum;
start_machine: marker, at = 0;
!
   n = 1;
   while (n < ncell+1) {
   qf: qf,      at=(n-1)*lcell;
   msf: msf,    at=(n-1)*lcell + lsex/2.0;
   bpm: bpm,    at=(n-1)*lcell + lsex + lbpm2;
   ch : ch,     at=(n-1)*lcell + lsex + lbpm + lcor2;
   mb: mb,      at=(n-1)*lcell+0.15*lcell;
   mb: mb,      at=(n-1)*lcell+0.35*lcell;
   qd: qd,      at=(n-1)*lcell+0.50*lcell;
   msd: msd,    at=(n-1)*lcell+0.50*lcell + lsex/2.0;
   bpm: bpm,    at=(n-1)*lcell+0.50*lcell + lsex + lbpm2;
   cv:  cv,     at=(n-1)*lcell+0.50*lcell + lsex + lbpm + lcor2;
   mb: mb,      at=(n-1)*lcell+0.65*lcell;
   mb: mb,      at=(n-1)*lcell+0.85*lcell;
!
   n = n + 1;
}
end_machine: marker at=circum;
endsequence;
```

To correct an orbit with MAD, one single command for each plane is required (EX5/ex5.madx):

```
correct,flag=ring,mode=micado,error=1.0e-7,ncorr=20,
        plane=x,clist="c.tab",mlist="m.tab";
```

```
correct,flag=ring,mode=micado,error=1.0e-7,ncorr=20,
        plane=y,clist="c.tab",mlist="m.tab";
```

The desired algorithms (here MICADO), the plane, the number of correctors (20) and some output files are passed as parameters.

A following execution of a Twiss command will make use of the computed correction and give the corrected orbit. A graphical output of the corrected orbits is given in Fig.9.

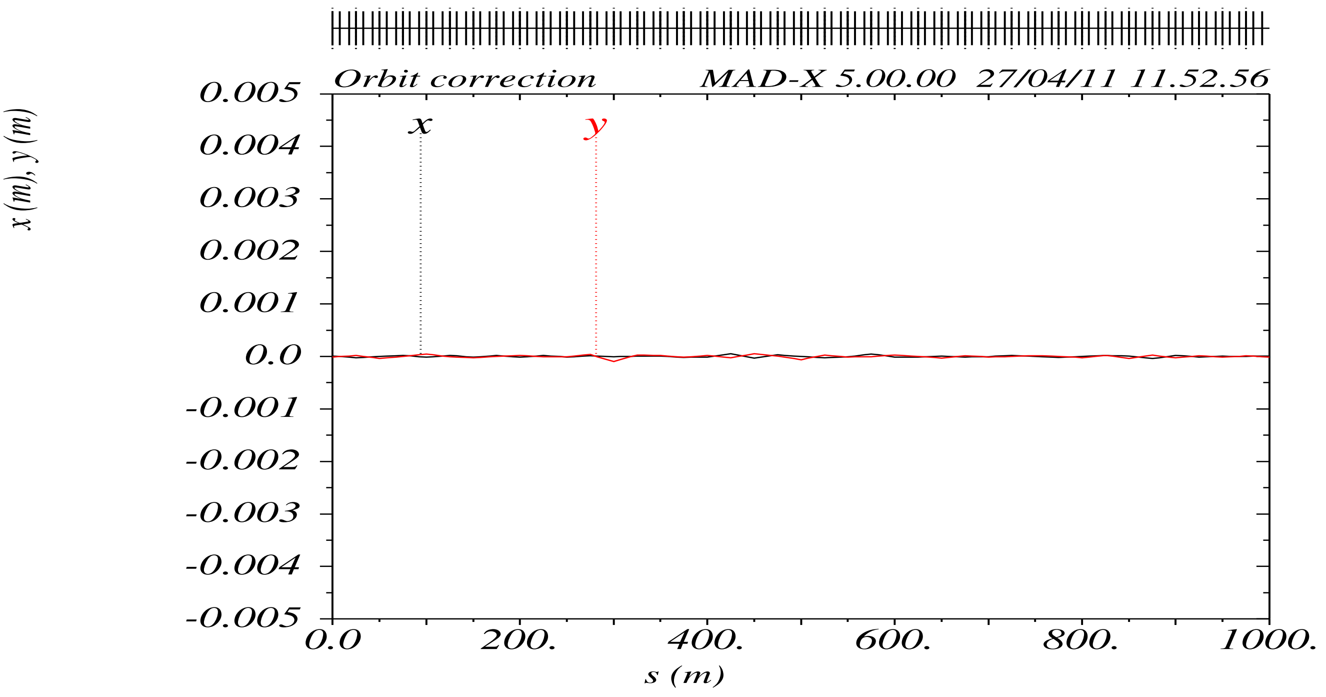

Figure 9: Horizontal and vertical orbit distortion, after orbit correction.

Although in this course we treat only circular machines, MAD-X can be used to compute the optical functions of beam lines and linear accelerators. Further it is possible to correct trajectories, i.e. single pass 'orbits', changing the flag above from $flag = ring$ to $flag = line$.

7 Exercise 6

7.1 Problem:

The purpose of this exercise is to insert dispersion suppressors into the existing regular lattice. Start with the lattice from the previous exercise and first double the circumference to 2000 m. Change the phase advance to $\phi = 60^o$ per cell. Insert two straight sections (each of 2 cells): i.e. cells without bending magnets but keep the same focusing of the quadrupoles. Insert the two straight sections opposite in azimuth in the ring. Modify now the lattice to keep the horizontal dispersion function small ($<$ 1-2 m) along this straight section, i.e. set up a dispersion suppressor. You can do this by adding or removing bending magnets or changing the bending radius of some or all the bending magnets.
At this stage do not change the focusing properties in any of the cells. Such straight sections with very small dispersion are very useful for the installation of RF equipment, wigglers, undulators, beam instrumentation, collimation systems etc., or to house an experiment.
As a second exercise, replace all bending magnets presently implemented as thin lenses, by thick lenses, i.e. define them as SBEND with the same angle. Look at the result and explain what you observe.

7.2 Solution:

We increase the circumference by changing its value in the lattice definition (sequence file) to 2000.0 m and the number of cells to 40. We define two sections without dipoles and let them be the cells number 7 and 8. These define our straight section. Before 7 and behind 8 we have to modify the lattice to make a $dispersion\ suppressor$. Do the same manipulation on the opposite side of the ring, e.g. for cells with the original numbers 27 and 28. Please note, the choice of the cell numbers for the dispersion suppressors is arbitrary. Two types of suppressors we know: missing magnet dispersion suppressors and half-field dispersion suppressors [4].

To change the phase advance, we insert the following lines into the MAD input (EX6/HF.madx):

```
match, sequence=cascell6;
   vary,name=kqf, step=0.00001;
   vary,name=kqd, step=0.00001;
   global,sequence=cascell6,Q1=6.700;
   global,sequence=cascell6,Q2=6.650;
   Lmdif, calls=10, tolerance=1.0e-21;
endmatch;
```

Our phase advance per cell is now close to 60^{o}[1]. This allows us to use both types of dispersion suppressors [4].

Please note: because of the changed phase advance it will be necessary to recompute the chromaticity correction, e.g. by matching DQ1 and DQ2 to the desired values. Furthermore, sextupoles should only be positioned in the part with regular dispersion, i.e. the arc outside the dispersion suppressor.

We reserve the cells 3 to 6 and 9 to 12 for setting up the dispersion suppressors. Equivalent for the opposite side of the ring (23 to 26 and 29 to 32). To modify them easily, we have taken their description out of the loop in the sequence. The remaining cells we do not modify or change. We do not change the strengths in the quadrupoles, i.e. we keep the focusing properties of the FODO cell.

[1]A phase advance of exactly 60^o would require to adjust the tune onto the 3rd order resonance (6.6666), which must be avoided. We have therefore chosen values slightly above and below this value.

7.2.1 Missing magnet dispersion suppressor

From the conditions for a missing magnet dispersion suppressor with 60^o phase advance we find that we need 2 cells on each side of the straight section. The outmost cells without bending magnets (cells 5 and 10) and the inner cell with the normal bending strength (cells 6 and 9) [4]. The cells 3, 4, 11, 12 can remain unchanged here. The steps are schematically shown in Fig.10. In the MAD description this

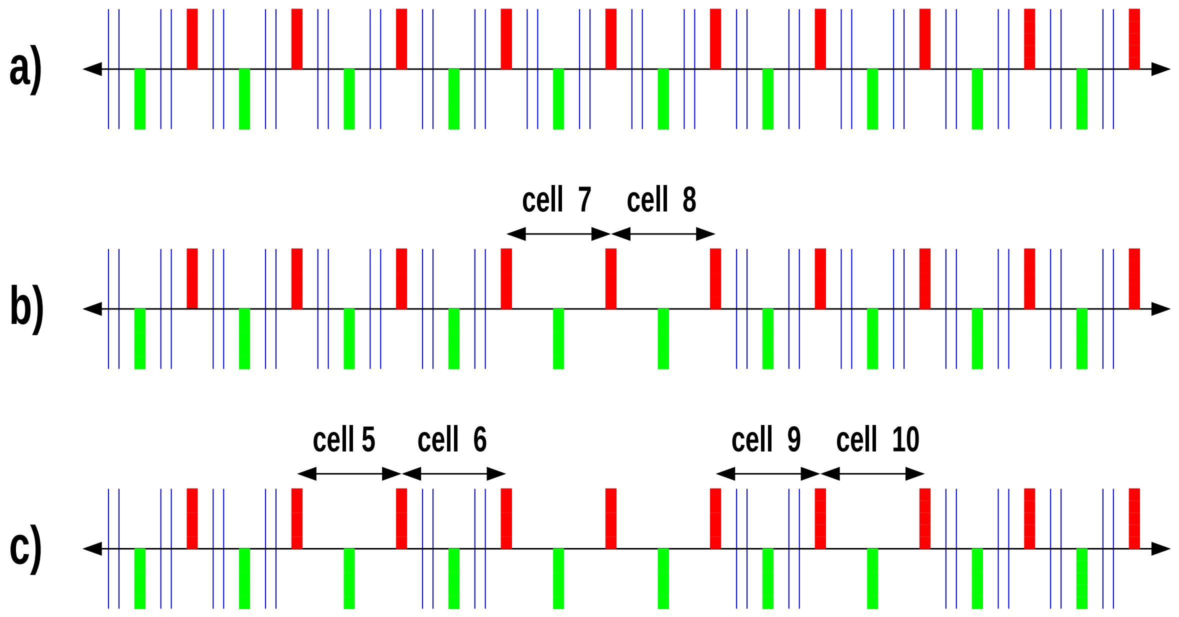

Figure 10: Schematic layout for missing magnet dispersion suppressor. a) Regular lattice layout. b) Straight section inserted. c) Missing magnet sections inserted.

is done easily by restricting the loop over cells to the regular arc cells and adding the tuning and straight section cells explicitly.

We have to be careful with the definition of the strength of the bending magnets because their number is now not 4*ncell. In the case of the missing magnet case a simple counting is enough, for the reduced field suppressor the fields have to be integrated to give 2π. A variable $nnorm$ is introduced to this purpose to compute the required deflection angle of the bending magnets. Please note: for an incorrect bending field the machine does not close.
Note that we have no sextupoles in the dispersion suppressors and the straight section (EX6/casMM.seq).

```
// defines a missing magnet dispersion suppressor
circum=2000.0;
ncell = 40;
lcell = circum/ncell;
lq = 3.00;
lsex = .0001;

nnorm = 128;

// forces and other constants;
// element definitions;

// define bending magnet as multipole
mb: multipole, knl={2.0*pi/nnorm};
```

```
// define the quadrupoles as multipole
kqf =  .980000e-2;
kqd = -.980000e-2;
qf: multipole, knl:={0,lq*kqf};
qd: multipole, knl:={0,lq*kqd};

// define the sextupoles as multipole
ksf :=  +0.017041/20.0;
ksd :=  -0.024714/20.0;

// ATTENTION: must use knl:=  not knl= to match !
msf: multipole, knl:={0,0,ksf};
msd: multipole, knl:={0,0,ksd};

// sequence declaration;
cascell6: sequence, refer=centre, l=circum;
start_machine: marker, at = 0;
!
   n = 0;
   while (n < 4) {
   qf: qf,     at=(n)*lcell;
   msf: msf,   at=(n)*lcell + lsex/2.0;
   mb: mb,     at=(n)*lcell+0.15*lcell;
   mb: mb,     at=(n)*lcell+0.35*lcell;
   qd: qd,     at=(n)*lcell+0.50*lcell;
   msd: msd,   at=(n)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,     at=(n)*lcell+0.65*lcell;
   mb: mb,     at=(n)*lcell+0.85*lcell;
!
   n = n + 1;
}

//   start dispersion suppressor
   qf: qf,     at=(n)*lcell;
   qd: qd,     at=(n)*lcell+0.50*lcell;

   qf: qf,     at=(n+1)*lcell;
   mb: mb,     at=(n+1)*lcell+0.15*lcell;
   mb: mb,     at=(n+1)*lcell+0.35*lcell;
   qd: qd,     at=(n+1)*lcell+0.50*lcell;
   mb: mb,     at=(n+1)*lcell+0.65*lcell;
   mb: mb,     at=(n+1)*lcell+0.85*lcell;
//   end dispersion suppressor

//   begin straight section
   qf: qf,     at=(n+2)*lcell;
   qd: qd,     at=(n+2)*lcell+0.50*lcell;

   qf: qf,     at=(n+3)*lcell;
   qd: qd,     at=(n+3)*lcell+0.50*lcell;
```

```
//   end straight section

//   start dispersion suppressor
   qf: qf,      at=(n+4)*lcell;
   mb: mb,      at=(n+4)*lcell+0.15*lcell;
   mb: mb,      at=(n+4)*lcell+0.35*lcell;
   qd: qd,      at=(n+4)*lcell+0.50*lcell;
   mb: mb,      at=(n+4)*lcell+0.65*lcell;
   mb: mb,      at=(n+4)*lcell+0.85*lcell;

   qf: qf,      at=(n+5)*lcell;
   qd: qd,      at=(n+5)*lcell+0.50*lcell;
//   end dispersion suppressor

   n = n + 6;
   while (n < 24) {
   qf: qf,      at=(n)*lcell;
   msf: msf,    at=(n)*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.15*lcell;
   mb: mb,      at=(n)*lcell+0.35*lcell;
   qd: qd,      at=(n)*lcell+0.50*lcell;
   msd: msd,    at=(n)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.65*lcell;
   mb: mb,      at=(n)*lcell+0.85*lcell;
!
   n = n + 1;
}
//   start dispersion suppressor
   qf: qf,      at=(n)*lcell;
   qd: qd,      at=(n)*lcell+0.50*lcell;

   qf: qf,      at=(n+1)*lcell;
   mb: mb,      at=(n+1)*lcell+0.15*lcell;
   mb: mb,      at=(n+1)*lcell+0.35*lcell;
   qd: qd,      at=(n+1)*lcell+0.50*lcell;
   mb: mb,      at=(n+1)*lcell+0.65*lcell;
   mb: mb,      at=(n+1)*lcell+0.85*lcell;
//   end dispersion suppressor

//   begin straight section
   qf: qf,      at=(n+2)*lcell;
   qd: qd,      at=(n+2)*lcell+0.50*lcell;

   qf: qf,      at=(n+3)*lcell;
   qd: qd,      at=(n+3)*lcell+0.50*lcell;
//   end straight section

//   start dispersion suppressor
   qf: qf,      at=(n+4)*lcell;
   mb: mb,      at=(n+4)*lcell+0.15*lcell;
   mb: mb,      at=(n+4)*lcell+0.35*lcell;
   qd: qd,      at=(n+4)*lcell+0.50*lcell;
   mb: mb,      at=(n+4)*lcell+0.65*lcell;
   mb: mb,      at=(n+4)*lcell+0.85*lcell;

   qf: qf,      at=(n+5)*lcell;
   qd: qd,      at=(n+5)*lcell+0.50*lcell;
//   end dispersion suppressor

 n = n + 6;
   while (n < ncell) {
   qf: qf,      at=(n)*lcell;
   msf: msf,    at=(n)*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.15*lcell;
   mb: mb,      at=(n)*lcell+0.35*lcell;
   qd: qd,      at=(n)*lcell+0.50*lcell;
   msd: msd,    at=(n)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.65*lcell;
   mb: mb,      at=(n)*lcell+0.85*lcell;
!
   n = n + 1;
}
end_machine: marker at=circum;

endsequence;
```

7.2.2 Half-field dispersion suppressor

The half-field dispersion suppressor is conceptually even simpler and for a phase advance of 60^o per cell we need 3 cells on each side of the straight section with half the bending strength (Fig.11). We follow

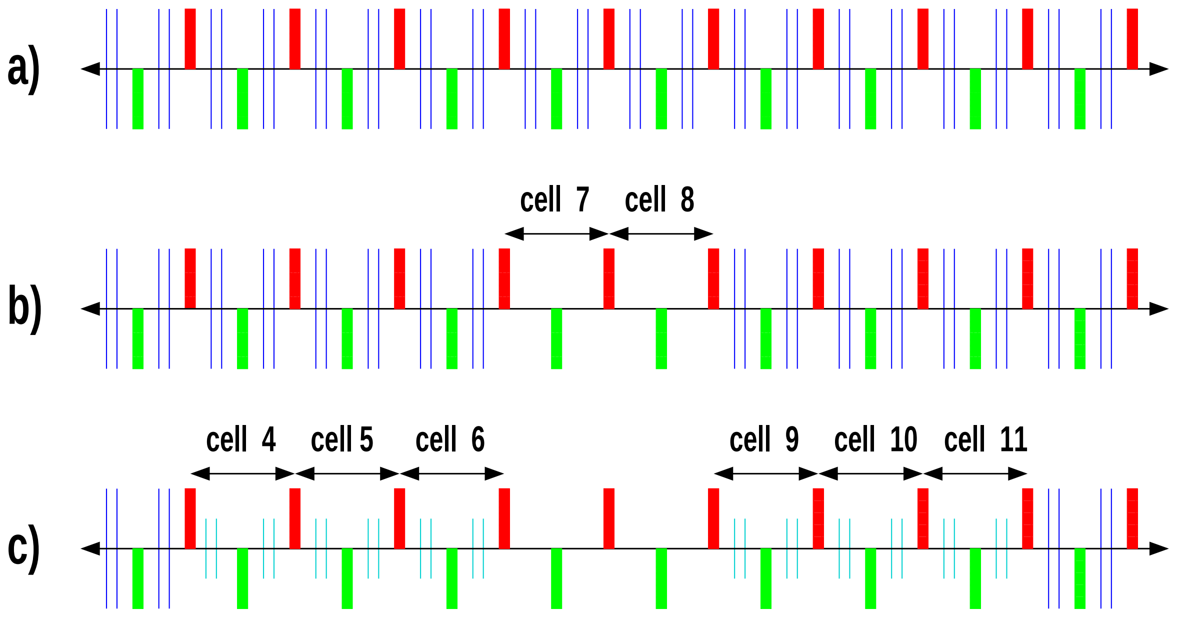

Figure 11: Schematic layout for half field dispersion suppressor. a) Regular lattice layout. b) Straight section inserted. c) Half field magnet sections inserted.

the same strategy as before and use the following lattice description (EX6/casHF.seq):

```
// defines a missing magnet dispersion suppressor
circum=2000.0;
ncell = 40;
lcell = circum/ncell;
lq = 3.00;
lsex = .0001;

nnorm = 120;

// forces and other constants;
// element definitions;

// define bending magnet as multipole
mb: multipole, knl={2.0*pi/nnorm};
mb2: multipole, knl={1.0*pi/nnorm};

// define the quadrupoles as multipole
kqf =  .980000e-2;
kqd = -.980000e-2;
qf: multipole, knl:={0,lq*kqf};
qd: multipole, knl:={0,lq*kqd};

// define the sextupoles as multipole
ksf :=  +0.017041/20.0;
ksd :=  -0.024714/20.0;

// ATTENTION: must use knl:=  not knl= to match !
msf: multipole, knl:={0,0,ksf};
msd: multipole, knl:={0,0,ksd};

// sequence declaration;
cascell6: sequence, refer=centre, l=circum;
start_machine: marker, at = 0;
!
   n = 0;
   qf: qf,     at=(n)*lcell;
   m1: marker,     at=(n)*lcell;
   msf: msf,   at=(n)*lcell + lsex/2.0;
   mb: mb,     at=(n)*lcell+0.15*lcell;
   mb: mb,     at=(n)*lcell+0.35*lcell;
   qd: qd,     at=(n)*lcell+0.50*lcell;
   msd: msd,   at=(n)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,     at=(n)*lcell+0.65*lcell;
   mb: mb,     at=(n)*lcell+0.85*lcell;
!
 n = 1;
   while (n < 3) {
   qf: qf,     at=(n)*lcell;
   msf: msf,    at=(n)*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.15*lcell;
   mb: mb,      at=(n)*lcell+0.35*lcell;
   qd: qd,      at=(n)*lcell+0.50*lcell;
   msd: msd,    at=(n)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.65*lcell;
   mb: mb,      at=(n)*lcell+0.85*lcell;
!
   n = n + 1;
}

//   start dispersion suppressor

   qf: qf,      at=(n)*lcell;
   mb2: mb2,      at=(n)*lcell+0.15*lcell;
   mb2: mb2,      at=(n)*lcell+0.35*lcell;
   qd: qd,      at=(n)*lcell+0.50*lcell;
   mb2: mb2,      at=(n)*lcell+0.65*lcell;
   mb2: mb2,      at=(n)*lcell+0.85*lcell;

   qf: qf,      at=(n+1)*lcell;
   mb2: mb2,      at=(n+1)*lcell+0.15*lcell;
   mb2: mb2,      at=(n+1)*lcell+0.35*lcell;
   qd: qd,      at=(n+1)*lcell+0.50*lcell;
   mb2: mb2,      at=(n+1)*lcell+0.65*lcell;
   mb2: mb2,      at=(n+1)*lcell+0.85*lcell;

   qf: qf,      at=(n+2)*lcell;
   mb2: mb2,      at=(n+2)*lcell+0.15*lcell;
   mb2: mb2,      at=(n+2)*lcell+0.35*lcell;
   qd: qd,        at=(n+2)*lcell+0.50*lcell;
   left: marker, at=(n+2)*lcell+0.50*lcell;
   mb2: mb2,      at=(n+2)*lcell+0.65*lcell;
   mb2: mb2,      at=(n+2)*lcell+0.85*lcell;
//   end dispersion suppressor

//   begin straight section
   qf: qf,      at=(n+3)*lcell;
   qd: qd,      at=(n+3)*lcell+0.50*lcell;

   qf: qf,      at=(n+4)*lcell;

   qd: qd,         at=(n+4)*lcell+0.50*lcell;
   right: marker, at=(n+4)*lcell+0.50*lcell;
//   end straight section

//   start dispersion suppressor
   qf: qf,      at=(n+5)*lcell;
   mb2: mb2,      at=(n+5)*lcell+0.15*lcell;
   mb2: mb2,      at=(n+5)*lcell+0.35*lcell;
   qd: qd,      at=(n+5)*lcell+0.50*lcell;
   mb2: mb2,      at=(n+5)*lcell+0.65*lcell;
   mb2: mb2,      at=(n+5)*lcell+0.85*lcell;

   qf: qf,      at=(n+6)*lcell;
   mb2: mb2,      at=(n+6)*lcell+0.15*lcell;
   mb2: mb2,      at=(n+6)*lcell+0.35*lcell;
   qd: qd,      at=(n+6)*lcell+0.50*lcell;
   mb2: mb2,      at=(n+6)*lcell+0.65*lcell;
   mb2: mb2,      at=(n+6)*lcell+0.85*lcell;

   qf: qf,      at=(n+7)*lcell;
   mb2: mb2,      at=(n+7)*lcell+0.15*lcell;
   mb2: mb2,      at=(n+7)*lcell+0.35*lcell;
   qd: qd,      at=(n+7)*lcell+0.50*lcell;
   mb2: mb2,      at=(n+7)*lcell+0.65*lcell;
   mb2: mb2,      at=(n+7)*lcell+0.85*lcell;

//   end dispersion suppressor

   n = n + 8;
   while (n < 23) {
   qf: qf,      at=(n)*lcell;
   msf: msf,    at=(n)*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.15*lcell;
   mb: mb,      at=(n)*lcell+0.35*lcell;
   qd: qd,      at=(n)*lcell+0.50*lcell;
   msd: msd,    at=(n)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.65*lcell;
   mb: mb,      at=(n)*lcell+0.85*lcell;
!
   n = n + 1;
}
//   start dispersion suppressor

   qf: qf,      at=(n)*lcell;
   mb2: mb2,      at=(n)*lcell+0.15*lcell;
   mb2: mb2,      at=(n)*lcell+0.35*lcell;
   qd: qd,      at=(n)*lcell+0.50*lcell;
   mb2: mb2,      at=(n)*lcell+0.65*lcell;
   mb2: mb2,      at=(n)*lcell+0.85*lcell;

   qf: qf,      at=(n+1)*lcell;
   mb2: mb2,      at=(n+1)*lcell+0.15*lcell;
   mb2: mb2,      at=(n+1)*lcell+0.35*lcell;
   qd: qd,      at=(n+1)*lcell+0.50*lcell;
   mb2: mb2,      at=(n+1)*lcell+0.65*lcell;
   mb2: mb2,      at=(n+1)*lcell+0.85*lcell;

   qf: qf,      at=(n+2)*lcell;
   mb2: mb2,     at=(n+2)*lcell+0.15*lcell;
   mb2: mb2,     at=(n+2)*lcell+0.35*lcell;
   qd: qd,     at=(n+2)*lcell+0.50*lcell;
   mb2: mb2,     at=(n+2)*lcell+0.65*lcell;
   mb2: mb2,     at=(n+2)*lcell+0.85*lcell;
//   end dispersion suppressor

//   begin straight section
   qf: qf,     at=(n+3)*lcell;
   qd: qd,     at=(n+3)*lcell+0.50*lcell;

   qf: qf,     at=(n+4)*lcell;
   qd: qd,     at=(n+4)*lcell+0.50*lcell;
//   end straight section

//   start dispersion suppressor
   qf: qf,     at=(n+5)*lcell;
   mb2: mb2,     at=(n+5)*lcell+0.15*lcell;
   mb2: mb2,     at=(n+5)*lcell+0.35*lcell;
   qd: qd,     at=(n+5)*lcell+0.50*lcell;
   mb2: mb2,     at=(n+5)*lcell+0.65*lcell;
   mb2: mb2,     at=(n+5)*lcell+0.85*lcell;

   qf: qf,     at=(n+6)*lcell;
   mb2: mb2,     at=(n+6)*lcell+0.15*lcell;
   mb2: mb2,     at=(n+6)*lcell+0.35*lcell;
   qd: qd,     at=(n+6)*lcell+0.50*lcell;
   mb2: mb2,     at=(n+6)*lcell+0.65*lcell;
   mb2: mb2,     at=(n+6)*lcell+0.85*lcell;

   qf: qf,     at=(n+7)*lcell;
   mb2: mb2,     at=(n+7)*lcell+0.15*lcell;
   mb2: mb2,     at=(n+7)*lcell+0.35*lcell;
   qd: qd,     at=(n+7)*lcell+0.50*lcell;
   mb2: mb2,     at=(n+7)*lcell+0.65*lcell;
   mb2: mb2,     at=(n+7)*lcell+0.85*lcell;

//   end dispersion suppressor

 n = n + 8;
   while (n < ncell) {
   qf: qf,     at=(n)*lcell;
   msf: msf,   at=(n)*lcell + lsex/2.0;
   mb: mb,     at=(n)*lcell+0.15*lcell;
   mb: mb,     at=(n)*lcell+0.35*lcell;
   qd: qd,     at=(n)*lcell+0.50*lcell;
   msd: msd,   at=(n)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,     at=(n)*lcell+0.65*lcell;
   mb: mb,     at=(n)*lcell+0.85*lcell;
!
```

```
    n = n + 1;
}
end_machine: marker at=circum;

endsequence;
```

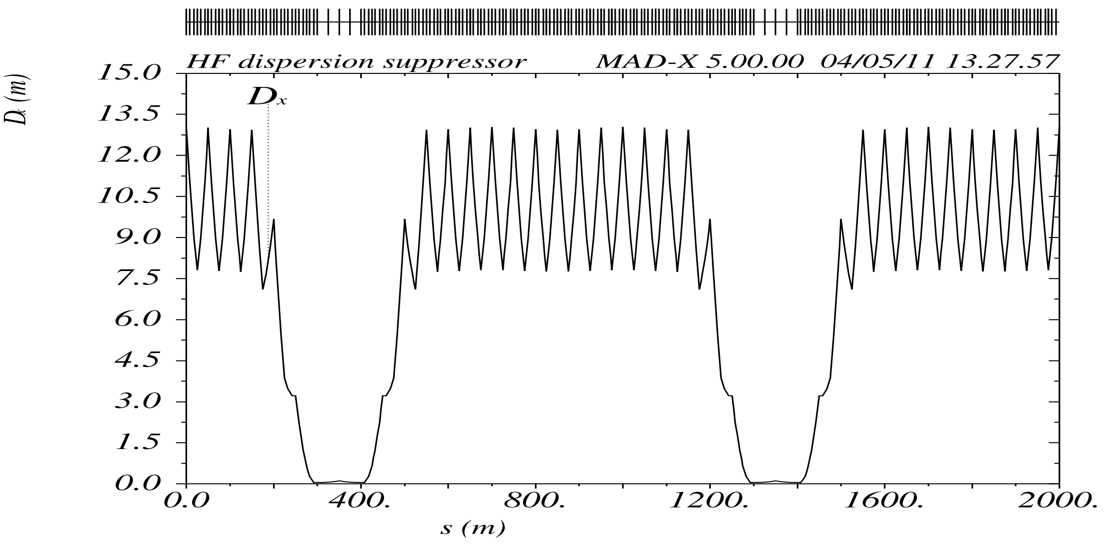

Figure 12: Horizontal dispersion with half-field dispersion suppressor.

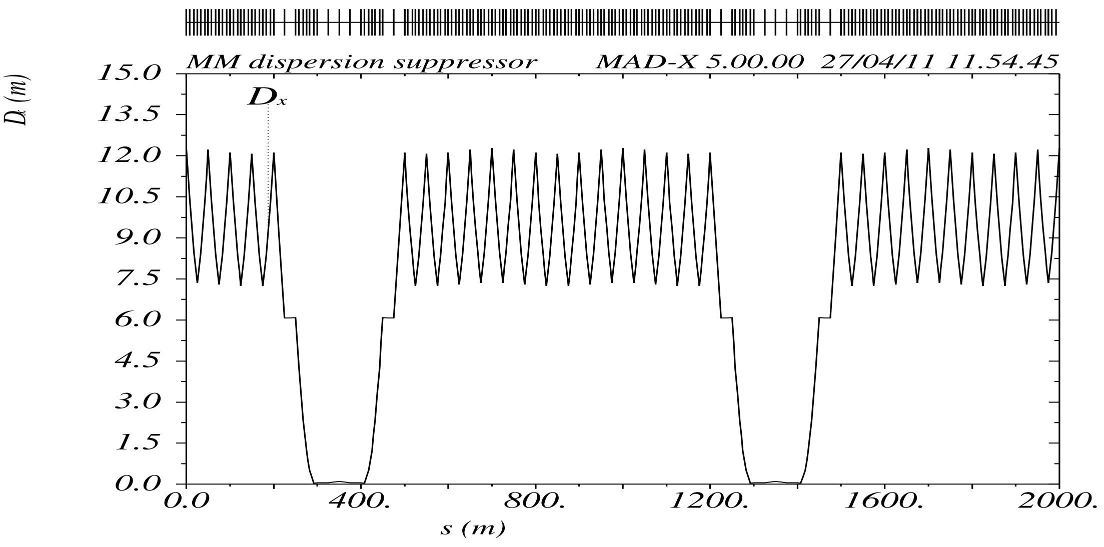

Figure 13: Horizontal dispersion with missing magnet dispersion suppressor.

We can see from Figs.12 and 13 that the dispersion D_x is reduced as wanted. In a second step one would use individual control of the quadrupoles in the straight section and dispersions suppressors to get the desired properties and to make the matching perfect. In Fig.14 we show a comparison of the geometrical layout of two circular machines with the same circumference, but with and without straight sections. Please note that with straight sections the remaining dipoles require a stronger bending field since the closure must be established with a smaller number of dipoles.
In Figs.15 and 16 we show the integrated bending angle along the circumference of the machine (full turn is -2π). The change of bending angle at the missing magnet dispersion suppressor (Fig.15) and the half field suppressor (Fig.16) are clearly visible and indicated by the arrows.

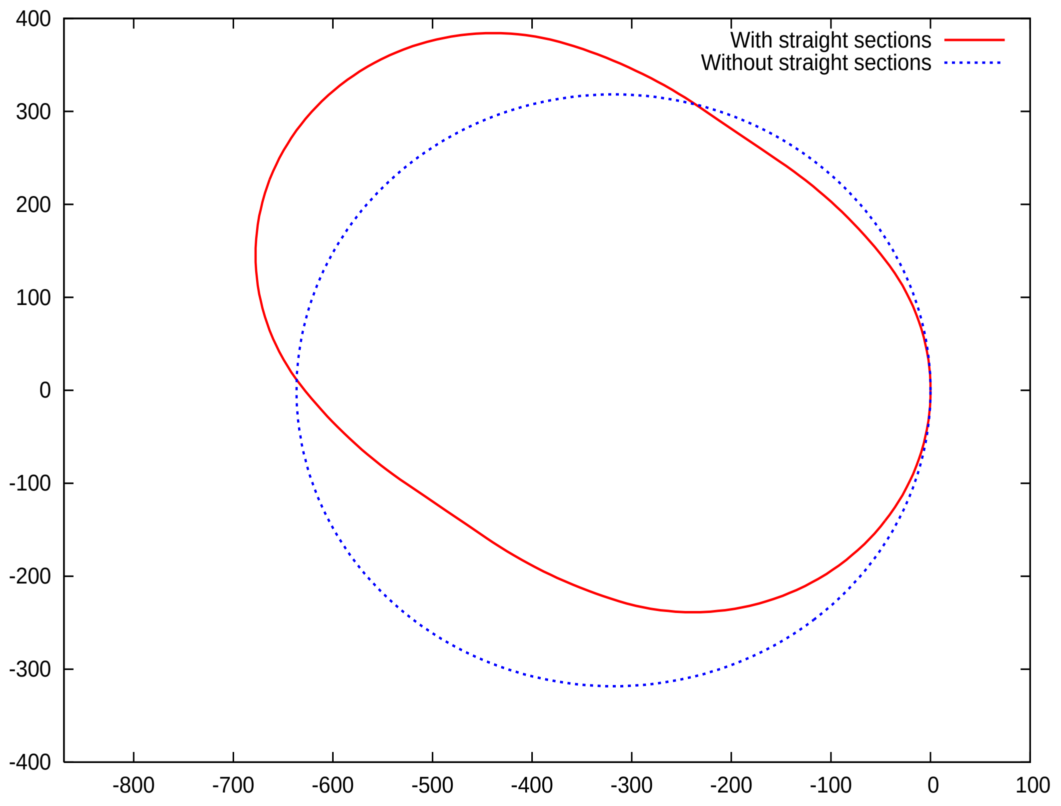

Figure 14: Circular machines of 2000 m circumference with and without straight sections.

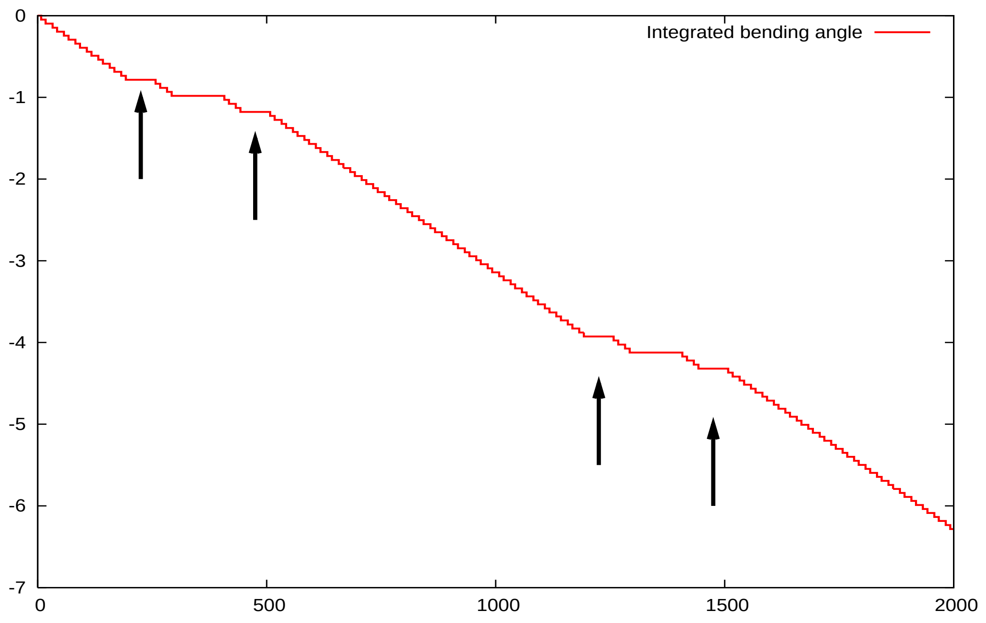

Figure 15: Integrated bending angle, missing magnet dispersion suppressor. Cells without bending clearly visible.

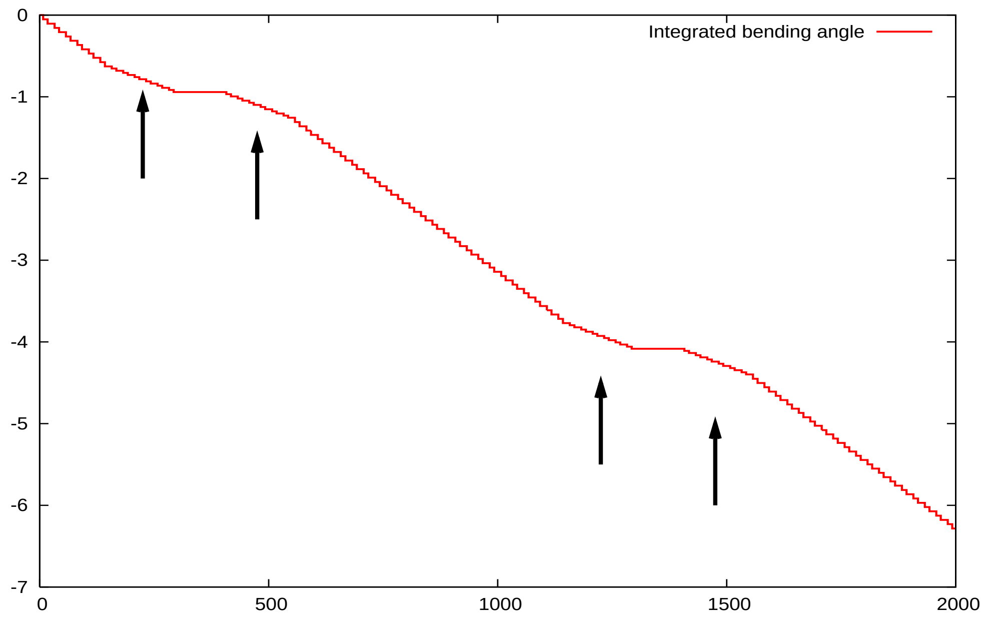

Figure 16: Integrated bending angle, half field dispersion suppressor. Cells with reduced bending clearly visible.

8 Exercise 7

8.1 Problem:

Start from the previous lattice and design a symmetric insertion with a low-β section in a dispersion free region. The β should be small at least in one plane and should have a waist at an "interaction point". Two options:

- Try to design the insertion yourself
- Use an already prepared example sequence and try to match

8.2 Solution:

To simplify the design, we should like to keep the geometry of the arc without modification. To that purpose we increase the length of already existing straight sections by 300m each (see Fig.17) by adding three additional, identical cells on each side of the existing straight section. The layout of the dispersion suppressor is not changed by this procedure, it is just moved to another position in the ring. The advantage of this procedure is that it provides more space and quadrupoles to accommodate the insertion.

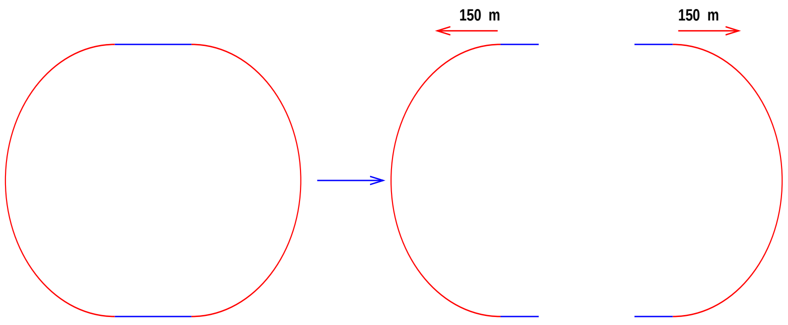

Figure 17: Provide space for insertion without changing the geometry of the arc.

This increases the circumference from 2000 m to 2600 m and the total number of cells from 40 to 52. Schematically this is shown in Fig.18 for one of the two foreseen insertions. The corresponding

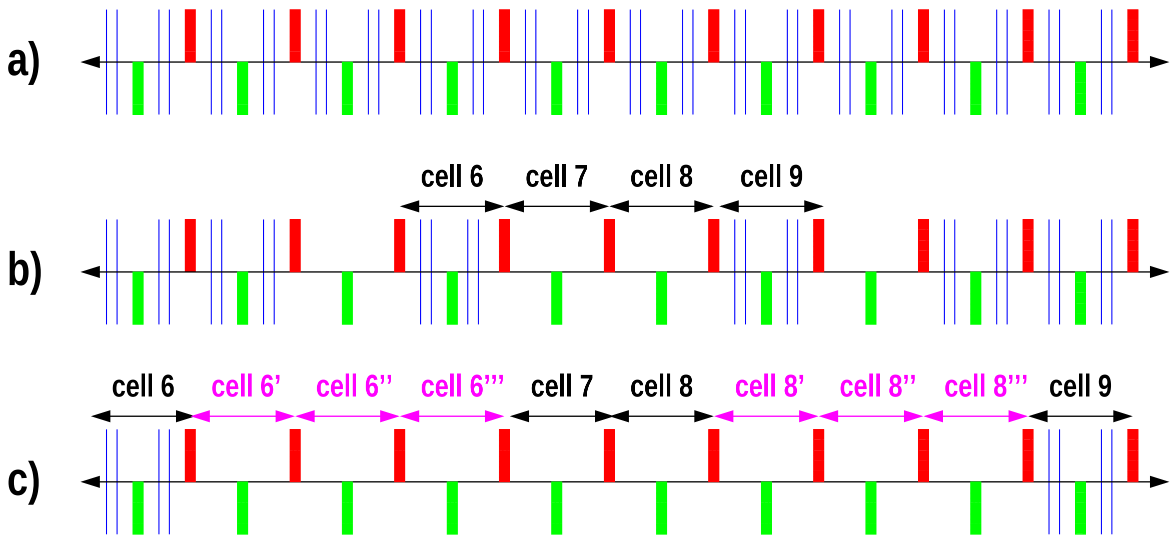

Figure 18: Schematic layout for missing magnet dispersion suppressor with additional straight sections for insertion. a) Regular lattice layout. b) Short straight section inserted. c) Long straight section inserted.

sequence is shown below.

```
// defines a missing magnet dispersion suppressor
circum=2600.0;
ncell = 52;
lcell = circum/ncell;
lq = 3.00;
lsex = .0001;

// constants for insertion
la = 0.0;

nnorm = 128;

// forces and other constants;
// element definitions;

// define bending magnet as multipole
mb: multipole, knl={2.0*pi/nnorm};

// define the quadrupoles as multipole
kqf =  1.33625e-2;
kqd = -1.33225e-2;

qf: multipole, knl:={0,lq*kqf};
qd: multipole, knl:={0,lq*kqd};

// define the sextupoles as multipole
ksf :=  +6.11394E-03;
ksd :=  -1.01706E-02;

// ATTENTION: must use knl:=  not knl= to match !
msf: multipole, knl:={0,0,ksf};
msd: multipole, knl:={0,0,ksd};

// sequence declaration;
cascell6: sequence, refer=centre, l=circum;
start_machine: marker, at = 0;
!
   n = 0;
   while (n < 4) {
   qf: qf,      at=(n)*lcell;
   msf: msf,    at=(n)*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.15*lcell;
   mb: mb,      at=(n)*lcell+0.35*lcell;
   qd: qd,      at=(n)*lcell+0.50*lcell;
   msd: msd,    at=(n)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.65*lcell;
   mb: mb,      at=(n)*lcell+0.85*lcell;
!
   n = n + 1;
}
```

```
//   start dispersion suppressor
   qf: qf,     at=(n)*lcell;
   qd: qd,     at=(n)*lcell+0.50*lcell;

   qf: qf,     at=(n+1)*lcell;
   mb: mb,     at=(n+1)*lcell+0.15*lcell;
   mb: mb,     at=(n+1)*lcell+0.35*lcell;
   qd: qd,     at=(n+1)*lcell+0.50*lcell;
   mb: mb,     at=(n+1)*lcell+0.65*lcell;
   mb: mb,     at=(n+1)*lcell+0.85*lcell;
//   end dispersion suppressor

//   begin straight section
   qf: qf,     at=(n+2)*lcell;
   qd: qd,     at=(n+2)*lcell+0.50*lcell;

   qf: qf,     at=(n+3)*lcell;
   qd: qd,     at=(n+3)*lcell+0.50*lcell;

   qf: qf,     at=(n+4)*lcell;
   qd: qd,     at=(n+4)*lcell+0.50*lcell;

   qf: qf,     at=(n+5)*lcell;
   qd: qd,     at=(n+5)*lcell+0.50*lcell;

   qf: qf,     at=(n+6)*lcell;
   qd: qd,     at=(n+6)*lcell+0.50*lcell;

   qf: qf,     at=(n+7)*lcell;
   qd: qd,     at=(n+7)*lcell+0.50*lcell;

   qf: qf,     at=(n+8)*lcell;
   qd: qd,     at=(n+8)*lcell+0.50*lcell;

   qf: qf,     at=(n+9)*lcell;
   qd: qd,     at=(n+9)*lcell+0.50*lcell;
//   end straight section

//   start dispersion suppressor
   qf: qf,     at=(n+10)*lcell;
   mb: mb,     at=(n+10)*lcell+0.15*lcell;
   mb: mb,     at=(n+10)*lcell+0.35*lcell;
   qd: qd,     at=(n+10)*lcell+0.50*lcell;
   mb: mb,     at=(n+10)*lcell+0.65*lcell;
   mb: mb,     at=(n+10)*lcell+0.85*lcell;

   qf: qf,     at=(n+11)*lcell;
   qd: qd,     at=(n+11)*lcell+0.50*lcell;
//   end dispersion suppressor
```

```
   n = n + 12;
   while (n < 30) {
   qf: qf,      at=(n)*lcell;
   msf: msf,    at=(n)*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.15*lcell;
   mb: mb,      at=(n)*lcell+0.35*lcell;
   qd: qd,      at=(n)*lcell+0.50*lcell;
   msd: msd,    at=(n)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.65*lcell;
   mb: mb,      at=(n)*lcell+0.85*lcell;
!
   n = n + 1;
}
//   start dispersion suppressor
   qf: qf,      at=(n)*lcell;
   qd: qd,      at=(n)*lcell+0.50*lcell;

   qf: qf,      at=(n+1)*lcell;
   mb: mb,      at=(n+1)*lcell+0.15*lcell;
   mb: mb,      at=(n+1)*lcell+0.35*lcell;
   qd: qd,      at=(n+1)*lcell+0.50*lcell;
   mb: mb,      at=(n+1)*lcell+0.65*lcell;
   mb: mb,      at=(n+1)*lcell+0.85*lcell;
//   end dispersion suppressor

//   begin straight section
   qf: qf,      at=(n+2)*lcell;
   qd: qd,      at=(n+2)*lcell+0.50*lcell;

   qf: qf,      at=(n+3)*lcell;
   qd: qd,      at=(n+3)*lcell+0.50*lcell;

   qf: qf,      at=(n+4)*lcell;
   qd: qd,      at=(n+4)*lcell+0.50*lcell;

   qf: qf,      at=(n+5)*lcell;
   qd: qd,      at=(n+5)*lcell+0.50*lcell;

   qf: qf,      at=(n+6)*lcell;
   qd: qd,      at=(n+6)*lcell+0.50*lcell;

   qf: qf,      at=(n+7)*lcell;
   qd: qd,      at=(n+7)*lcell+0.50*lcell;

   qf: qf,      at=(n+8)*lcell;
   qd: qd,      at=(n+8)*lcell+0.50*lcell;

   qf: qf,      at=(n+9)*lcell;
   qd: qd,      at=(n+9)*lcell+0.50*lcell;
```

```
//   end straight section

//   start dispersion suppressor
   qf: qf,     at=(n+10)*lcell;
   mb: mb,     at=(n+10)*lcell+0.15*lcell;
   mb: mb,     at=(n+10)*lcell+0.35*lcell;
   qd: qd,     at=(n+10)*lcell+0.50*lcell;
   mb: mb,     at=(n+10)*lcell+0.65*lcell;
   mb: mb,     at=(n+10)*lcell+0.85*lcell;

   qf: qf,     at=(n+11)*lcell;
   qd: qd,     at=(n+11)*lcell+0.50*lcell;
//   end dispersion suppressor

 n = n + 12;
   while (n < ncell) {
   qf: qf,     at=(n)*lcell;
   msf: msf,   at=(n)*lcell + lsex/2.0;
   mb: mb,     at=(n)*lcell+0.15*lcell;
   mb: mb,     at=(n)*lcell+0.35*lcell;
   qd: qd,     at=(n)*lcell+0.50*lcell;
   msd: msd,   at=(n)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,     at=(n)*lcell+0.65*lcell;
   mb: mb,     at=(n)*lcell+0.85*lcell;
!
   n = n + 1;
}
end_machine: marker at=circum;

endsequence;
```

8.2.1 Initial parameter settings

In the next step we redefine the strengths of the quadrupoles in a straight section to allow individual strengths. Since the insertion should be symmetric, the strengths of the equivalent quadrupoles left and right of the IP must be the same. We choose the straight section starting at cell 4 for our insertion, the second straight section remains unchanged. Initially the quadrupoles are defined with their original strengths.
In the centre of the long straight section we define an ”interaction point” IP as a marker to serve as a reference.
Since we want a symmetric β-function around the IP (minimum at IP and symmetric waist), the neighbouring quadrupoles have to be of the same type. In the final solution the IP therefore replaces a defocusing quadrupole.
The optical functions outside the insertion must remain unchanged and for the matching procedure we choose two defocusing quadrupoles as reference. These quadrupoles should have the optical functions (here: $\beta_{x,y}$ and $\alpha_{x,y}$) of the regular lattice quadrupoles, but their choice is not unique: any quadrupole in the regular lattice would serve this purpose. To that purpose we have installed markers as reference points next to these quadrupoles (called **left** and **right**).
The quadrupole next to the interaction point we leave in place at this stage, but give it an individual strength and name (**qd1**). The reason for this procedure will become clear in the next section. These

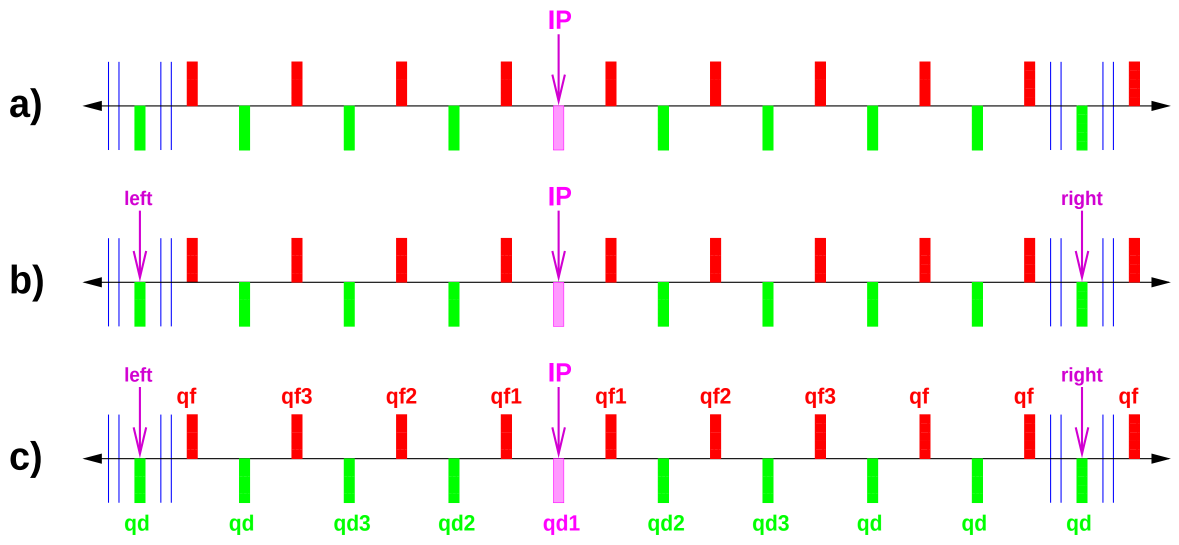

Figure 19: Schematic layout for insertion. a) Long stragiht section with IP defined. b) Markers (**left**, **right**) inserted at regular quadrupole (reference point) inserted. c) Independent (but symmetric) quadrupoles in insertion.

steps are illustrated in Fig.19.
The resulting sequence is then:

```
// defines a missing magnet dispersion suppressor
circum=2600.0;
ncell = 52;
lcell = circum/ncell;
lq = 3.00;
lsex = .0001;

// constants for insertion
la = 0.0;

nnorm = 128;

// forces and other constants;
// element definitions;

// define bending magnet as multipole
mb: multipole, knl={2.0*pi/nnorm};

// define the quadrupoles as multipole
kqf =  1.33625e-2;
kqd = -1.33225e-2;

qf: multipole, knl:={0,lq*kqf};
qd: multipole, knl:={0,lq*kqd};

kqf1 =  1.33625e-2;
kqd1 = -1.33275e-2;
kqf2 =  1.33625e-2;
```

```
kqd2 = -1.33275e-2;
kqf3 =  1.33625e-2;
kqd3 = -1.33275e-2;

qf1: multipole, knl:={0,lq*kqf1};
qd1: multipole, knl:={0,lq*kqd1};
qf2: multipole, knl:={0,lq*kqf2};
qd2: multipole, knl:={0,lq*kqd2};
qf3: multipole, knl:={0,lq*kqf3};
qd3: multipole, knl:={0,lq*kqd3};

// define the sextupoles as multipole
ksf :=  +6.11394E-03;
ksd :=  -1.01706E-02;

// ATTENTION: must use knl:=  not knl= to match !
msf: multipole, knl:={0,0,ksf};
msd: multipole, knl:={0,0,ksd};

// sequence declaration;
cascell6: sequence, refer=centre, l=circum;
start_machine: marker, at = 0;
!
   n = 0;
   while (n < 4) {
   qf: qf,     at=(n)*lcell;
   msf: msf,   at=(n)*lcell + lsex/2.0;
   mb: mb,     at=(n)*lcell+0.15*lcell;
   mb: mb,     at=(n)*lcell+0.35*lcell;
   qd: qd,     at=(n)*lcell+0.50*lcell;
   msd: msd,   at=(n)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,     at=(n)*lcell+0.65*lcell;
   mb: mb,     at=(n)*lcell+0.85*lcell;
!
   n = n + 1;
}

//   start dispersion suppressor
   qf: qf,     at=(n)*lcell;
   qd: qd,     at=(n)*lcell+0.50*lcell;

   qf: qf,     at=(n+1)*lcell;
   mb: mb,     at=(n+1)*lcell+0.15*lcell;
   mb: mb,     at=(n+1)*lcell+0.35*lcell;
   qd: qd,     at=(n+1)*lcell+0.50*lcell;
   left: marker,     at=(n+1)*lcell+0.50*lcell;
   mb: mb,     at=(n+1)*lcell+0.65*lcell;
   mb: mb,     at=(n+1)*lcell+0.85*lcell;
//   end dispersion suppressor
//   begin straight section
   qf: qf,     at=(n+2)*lcell;
   qd: qd,     at=(n+2)*lcell+0.50*lcell;

   qf3: qf3,     at=(n+3)*lcell;
   qd3: qd3,     at=(n+3)*lcell+0.50*lcell;

   qf2: qf2,     at=(n+4)*lcell;
   qd2: qd2,     at=(n+4)*lcell+0.50*lcell;

   qf1: qf1,     at=(n+5)*lcell;
   qd1: qd1,     at=(n+5)*lcell+0.50*lcell;
   IP: marker,   at=(n+5)*lcell+0.50*lcell;

   qf1: qf1,     at=(n+6)*lcell;
   qd2: qd2,     at=(n+6)*lcell+0.50*lcell;

   qf2: qf2,     at=(n+7)*lcell;
   qd3: qd3,     at=(n+7)*lcell+0.50*lcell;

   qf3: qf3,     at=(n+8)*lcell;
   qd: qd,     at=(n+8)*lcell+0.50*lcell;

   qf: qf,     at=(n+9)*lcell;
   qd: qd,     at=(n+9)*lcell+0.50*lcell;
//   end straight section

//   start dispersion suppressor
   qf: qf,     at=(n+10)*lcell;
   mb: mb,     at=(n+10)*lcell+0.15*lcell;
   mb: mb,     at=(n+10)*lcell+0.35*lcell;
   qd: qd,     at=(n+10)*lcell+0.50*lcell;
   right: marker,     at=(n+10)*lcell+0.50*lcell;
   mb: mb,     at=(n+10)*lcell+0.65*lcell;
   mb: mb,     at=(n+10)*lcell+0.85*lcell;

   qf: qf,     at=(n+11)*lcell;
   qd: qd,     at=(n+11)*lcell+0.50*lcell;
//   end dispersion suppressor

   n = n + 12;
   while (n < 30) {
   qf: qf,     at=(n)*lcell;
   msf: msf,   at=(n)*lcell + lsex/2.0;
   mb: mb,     at=(n)*lcell+0.15*lcell;
   mb: mb,     at=(n)*lcell+0.35*lcell;
   qd: qd,     at=(n)*lcell+0.50*lcell;
   msd: msd,   at=(n)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,     at=(n)*lcell+0.65*lcell;
   mb: mb,     at=(n)*lcell+0.85*lcell;
!
   n = n + 1;
}
//   start dispersion suppressor
   qf: qf,      at=(n)*lcell;
   qd: qd,      at=(n)*lcell+0.50*lcell;

   qf: qf,      at=(n+1)*lcell;
   mb: mb,      at=(n+1)*lcell+0.15*lcell;
   mb: mb,      at=(n+1)*lcell+0.35*lcell;
   qd: qd,      at=(n+1)*lcell+0.50*lcell;
   mb: mb,      at=(n+1)*lcell+0.65*lcell;
   mb: mb,      at=(n+1)*lcell+0.85*lcell;
//   end dispersion suppressor

//   begin straight section
   qf: qf,      at=(n+2)*lcell;
   qd: qd,      at=(n+2)*lcell+0.50*lcell;

   qf: qf,      at=(n+3)*lcell;
   qd: qd,      at=(n+3)*lcell+0.50*lcell;

   qf: qf,      at=(n+4)*lcell;
   qd: qd,      at=(n+4)*lcell+0.50*lcell;

   qf: qf,      at=(n+5)*lcell;
   qd: qd,      at=(n+5)*lcell+0.50*lcell;

   qf: qf,      at=(n+6)*lcell;
   qd: qd,      at=(n+6)*lcell+0.50*lcell;

   qf: qf,      at=(n+7)*lcell;
   qd: qd,      at=(n+7)*lcell+0.50*lcell;

   qf: qf,      at=(n+8)*lcell;
   qd: qd,      at=(n+8)*lcell+0.50*lcell;

   qf: qf,      at=(n+9)*lcell;
   qd: qd,      at=(n+9)*lcell+0.50*lcell;
//   end straight section

//   start dispersion suppressor
   qf: qf,      at=(n+10)*lcell;
   mb: mb,      at=(n+10)*lcell+0.15*lcell;
   mb: mb,      at=(n+10)*lcell+0.35*lcell;
   qd: qd,      at=(n+10)*lcell+0.50*lcell;
   mb: mb,      at=(n+10)*lcell+0.65*lcell;
   mb: mb,      at=(n+10)*lcell+0.85*lcell;

   qf: qf,      at=(n+11)*lcell;
   qd: qd,      at=(n+11)*lcell+0.50*lcell;
//   end dispersion suppressor

 n = n + 12;
   while (n < ncell) {
   qf: qf,      at=(n)*lcell;
   msf: msf,    at=(n)*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.15*lcell;
   mb: mb,      at=(n)*lcell+0.35*lcell;
   qd: qd,      at=(n)*lcell+0.50*lcell;
   msd: msd,    at=(n)*lcell+0.50*lcell + lsex/2.0;
   mb: mb,      at=(n)*lcell+0.65*lcell;
   mb: mb,      at=(n)*lcell+0.85*lcell;
!
   n = n + 1;
}
end_machine: marker at=circum;

endsequence;
```

8.2.2 Matching β functions

Computing the optical functions of the previously defined lattice, the functions are still fully periodic since the added lattice cells have the same focusing properties of the regular machine. In the next step we have to modify (match) the strengths of these cells to satisfy the requirements.
The matching of the desired insertion must fulfill three basic constraints:

- Desired value of β^* at the interaction point (IP)
- Optical parameters at position **left** must be the same as in the unchanged lattice
- Optical parameters at position **right** must be the same as in the unchanged lattice

The matching strategy is divided into separate simple steps for pedagogical reasons:
Step 1:
Run TWISS command on regular lattice with original quadrupole strengths and store optical parameters at the markers **left** and **right**.
Step 2:
Compute and plot the optical functions in the insertion (i.e. from **left** to **right**) for the original lattice. They serve as a reference during the matching.
Step 3:
Set the defocusing quadrupole (**qd1**) near IP to zero strength to make symmetric insertion around interaction point. Computing the optical functions now would give strongly distorted functions (probably the machine is not stable). Use the constraints of the inital optical functions **left** and the desired value at the IP for matching, allowing the independent quadrupoles of the insertion to change their values. Please note: only one half of the insertion is adjusted, implying full symmetry around the IP.
Step 4:
Recompute and plot the optical functions in the insertion after the matching.
Step 5:
Recompute and plot the optical functions for the full machine after the matching.

Using the previously defined sequence we use the following MAD input commands:

```
TITLE, 'Low beta insertion';
call file="casMMb.seq";
option,-echo;
// option,debug,-echo;

Beam, particle = proton, sequence=cascell6, energy = 20.0,
          NPART=1.05E11, sige=      4.5e-4 ;

use, period=cascell6;

savebeta, label = leftb0, place = left;
savebeta, label = rightb0, place = right;
select,flag=twiss,column=name,s,mux,muy,betx,alfx,bety,alfy,dx;
twiss,centre,file=twiss.out;
plot, haxis=s, vaxis=betx;
plot, haxis=s, vaxis=bety;
plot, haxis=s, vaxis=dx,vmin=0.0,vmax=15.0;
plot, haxis=s, vaxis=x;

use, sequence=cascell6, range=left/right;
twiss, beta0=leftb0,sequence=cascell6,file=twiss1.out;
plot, haxis=s, vaxis=betx;
plot, haxis=s, vaxis=bety;
plot, haxis=s, vaxis=dx,vmin=0.0,vmax=15.0;
plot, haxis=s, vaxis=x;

kqd1 = 0.0;
use, sequence=cascell6, range=left/ip;
match, sequence=cascell6,beta0=leftb0;
   vary,name=kqf3, step=0.00001;
   vary,name=kqd3, step=0.00001;
   vary,name=kqf2, step=0.00001;
   vary,name=kqd2, step=0.00001;
   vary,name=kqf1, step=0.00001;
!  vary,name=kqd1, step=0.00001;
   constraint,range=ip,sequence=cascell6,betx=10.0,bety=10.0,alfx=0.0,
                                        alfy=0.0,dx=0.0,dpx=0.0;
   Lmdif, calls=100, tolerance=1.0e-21;
endmatch;

use, sequence=cascell6, range=left/right;
twiss, beta0=leftb0,sequence=cascell6,file=twiss2.out;
plot, haxis=s, vaxis=betx;
plot, haxis=s, vaxis=bety;
plot, haxis=s, vaxis=dx,vmin=0.0,vmax=15.0;
plot, haxis=s, vaxis=x;


use, sequence=cascell6;
twiss, sequence=cascell6,file=twiss3.out;
```

```
plot, haxis=s, vaxis=betx;
plot, haxis=s, vaxis=bety;
plot, haxis=s, vaxis=dx,vmin=0.0,vmax=15.0;
plot, haxis=s, vaxis=x;

survey,file=survey.cas;


stop;
```

The results of the matching procedure are shown in Fig.20 and 21. Finally, in Fig.22 we show the

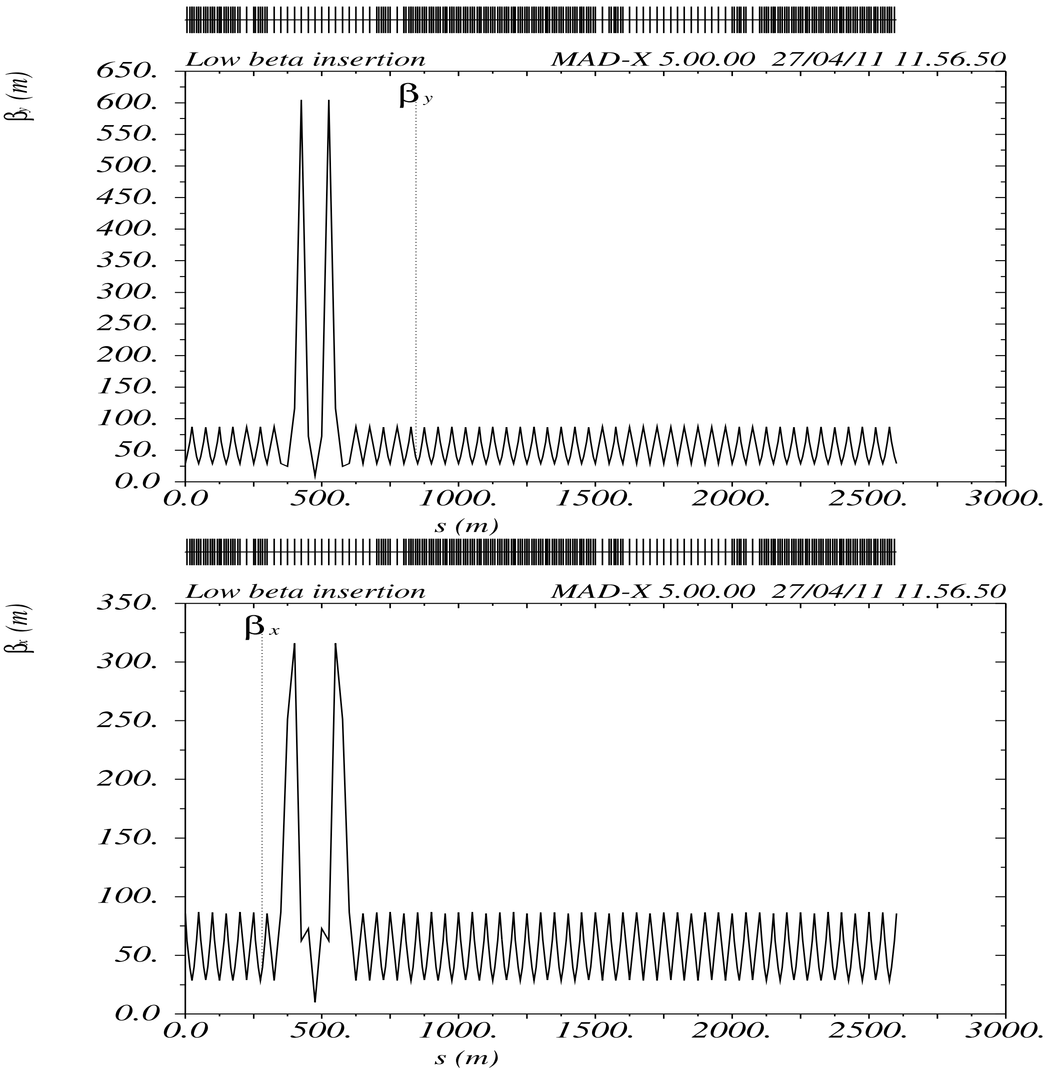

Figure 20: Matched β^* section. We show the horizontal and vertical β-function around the machine. Horizontal and vertical β-functions are 10 m.

geometry of the machine produced with the **survey** command. Clearly visible are the straight sections and the two arcs. Also shown as a demonstration is a case where the machine is not closed.

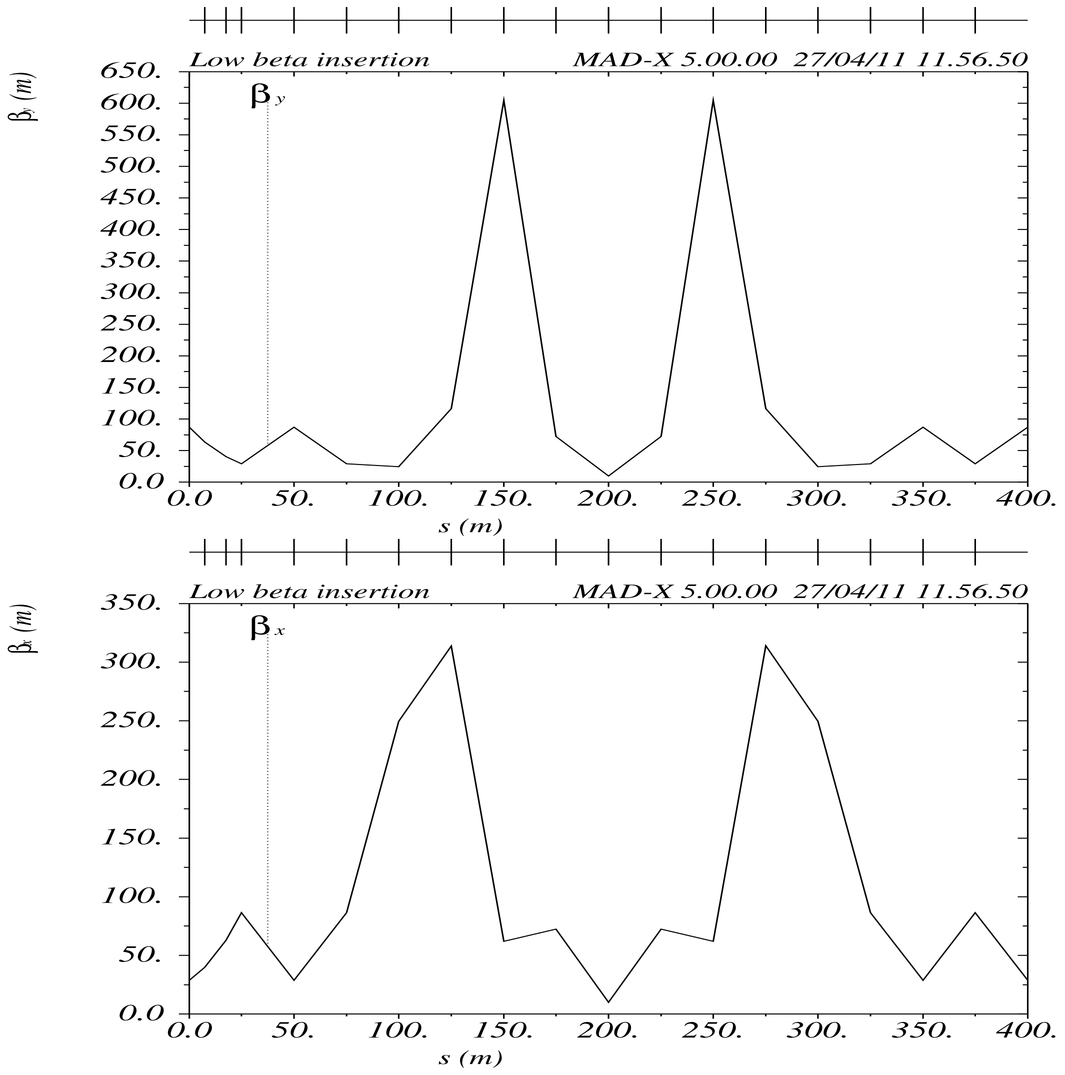

Figure 21: Matched β^* section. We show the horizontal and vertical β-function around the insertion. Horizontal and vertical β-functions are 10 m.

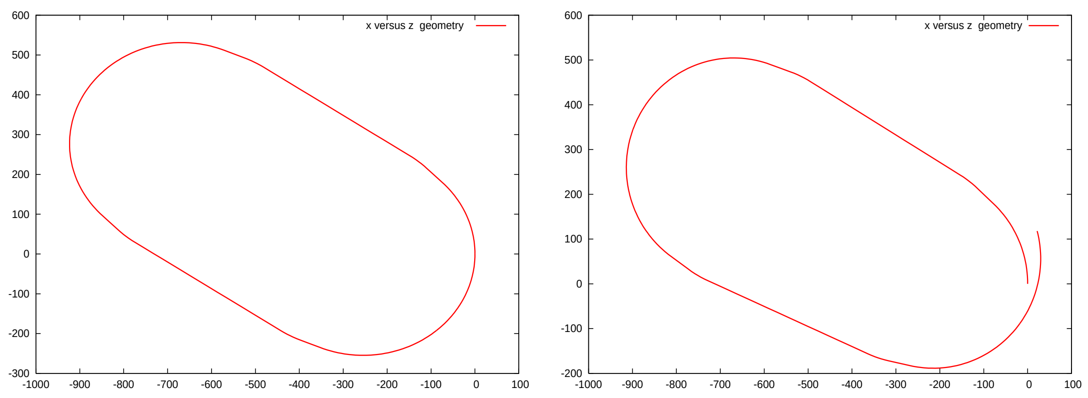

Figure 22: Geometries of a closed and a non-closed ring.

9 Exercise 8

9.1 Problem:

Use the lattice from exercise 6 and set up a single particle tracking to study the stability of the beams. Use the thin lens version for tracking with madx.

- Select appropriate particle amplitudes
- Change to tune and sextupole strengths to observe the effect
- Try to track with a thick lens lattice with PTC

9.2 Solution:

9.2.1 Tracking with MAD and normalized coordinates

```
TITLE, 'MAD-X thin lens tracking';
call file="casMM.seq";
option,-echo;
// option,debug,-echo;

Beam, particle = proton, sequence=cascell6, energy = 20.0,
          ex = 1.0e-4, ey = 1.0e-4,
          NPART=1.05E11, sige=      4.5e-4 ;

use, period=cascell6;


match, sequence=cascell6;
   vary,name=kqf, step=0.00001;
   vary,name=kqd, step=0.00001;
   global,sequence=cascell6,Q1=6.700;
   global,sequence=cascell6,Q2=6.650;
   Lmdif, calls=10, tolerance=1.0e-21;
endmatch;

match, sequence=cascell6;
   vary,name=ksf, step=0.00001;
   vary,name=ksd, step=0.00001;
   global,sequence=cascell6,DQ1=0.0;
   global,sequence=cascell6,DQ2=0.0;
   Lmdif, calls=10, tolerance=1.0e-21;
endmatch;


select,flag=twiss,column=name,s,mux,betx,dx,muy,bety,dy;
twiss,save,centre,file=twiss.out;
plot, haxis=s, vaxis=betx;
plot, haxis=s, vaxis=bety;
plot, haxis=s, vaxis=dx,vmin=0.0,vmax=15.0;
plot, haxis=s, vaxis=x;

option,trace;
```

```
small=0.10;
!track,dump;
track;

nsigmax=5;
n=1; // sigma multiplier

while (n <= nsigmax)
{
xs=n*1.0;
ys=n*1.0;
value,xs,ys;
start,fx=xs,phix=0.0,fy=ys,phiy=0.0;
n = n + 1;
}
run,turns=1024;

endtrack;

plot, file="MAD_track",table=track,haxis=x,vaxis=px,
      particle=1,2,3,4,5, colour=1000, multiple, symbol=3;
plot, file="MAD_track",table=track,haxis=y,vaxis=py,
      particle=1,2,3,4,5, colour=1000, multiple, symbol=3;

stop;
```

9.2.2 Tracking with MAD and absolute coordinates

```
TITLE, 'MAD-X thin lens tracking';
call file="casMM.seq";
option,-echo;
// option,debug,-echo;

Beam, particle = proton, sequence=cascell6, energy = 20.0,
          ex = 1.0e-4, ey = 1.0e-4,
          NPART=1.05E11, sige=      4.5e-4 ;

use, period=cascell6;


match, sequence=cascell6;
   vary,name=kqf, step=0.00001;
   vary,name=kqd, step=0.00001;
   global,sequence=cascell6,Q1=6.690;
   global,sequence=cascell6,Q2=6.700;
   Lmdif, calls=10, tolerance=1.0e-21;
endmatch;

match, sequence=cascell6;
   vary,name=ksf, step=0.00001;
   vary,name=ksd, step=0.00001;
   global,sequence=cascell6,DQ1=0.0;
   global,sequence=cascell6,DQ2=0.0;
   Lmdif, calls=10, tolerance=1.0e-21;
endmatch;


select,flag=twiss,column=name,s,mux,betx,dx,muy,bety,dy;
twiss,save,centre,file=twiss.out;
plot, haxis=s, vaxis=betx;
plot, haxis=s, vaxis=bety;
plot, haxis=s, vaxis=dx,vmin=0.0,vmax=15.0;
plot, haxis=s, vaxis=x;


track,dump;

start, x= 2e-2, px=0, y= 2e-2, py=0;
start, x= 4e-2, px=0, y= 4e-2, py=0;
start, x= 6e-2, px=0, y= 6e-2, py=0;
start, x= 8e-2, px=0, y= 8e-2, py=0;
start, x=10e-2, px=0, y=10e-2, py=0;
start, x=12e-2, px=0, y=12e-2, py=0;
start, x=14e-2, px=0, y=14e-2, py=0;
start, x=16e-2, px=0, y=16e-2, py=0;
start, x=18e-2, px=0, y=18e-2, py=0;

run,turns=1024;
endtrack;


plot, file="MAD_track",table=track,haxis=x,vaxis=px,
      particle=1,2,3,4,5,6,7,8,9, colour=1000, multiple, symbol=3;
plot, file="MAD_track",table=track,haxis=y,vaxis=py,
      particle=1,2,3,4,5,6,7,8,9, colour=1000, multiple, symbol=3;


stop;
```

9.2.3 Tracking with thick lenses and PTC and absolute coordinates

```
TITLE, 'Thick lens tracking with PTC';
call file="casMMi_thick.seq";
option,-echo;
// option,debug,-echo;

Beam, particle = proton, sequence=cascell6, energy = 20.0,
          NPART=1.05E11, sige=      4.5e-4 ;
```

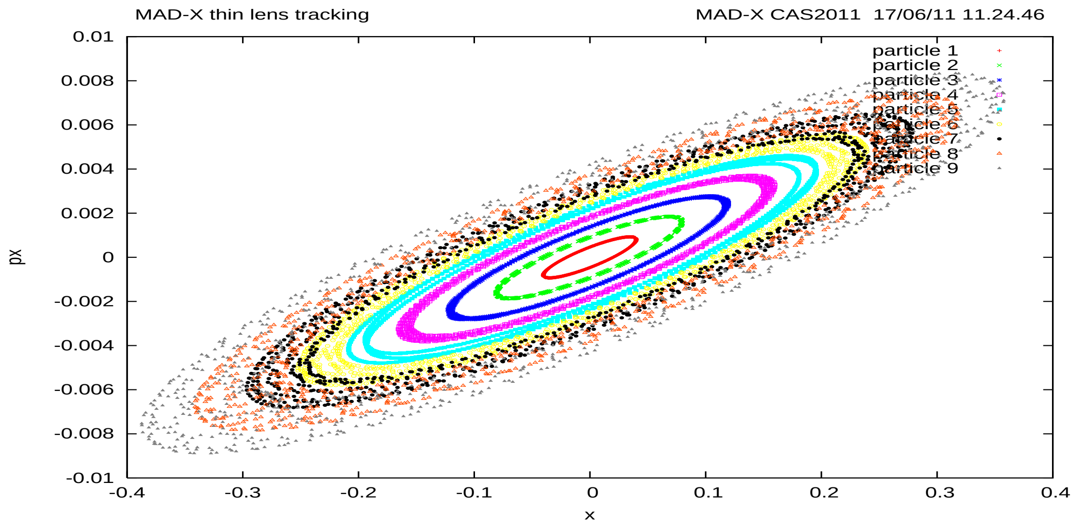

Figure 23: Tracking output - thin lenses with MAD-X. Plot shows phase horizontal space.

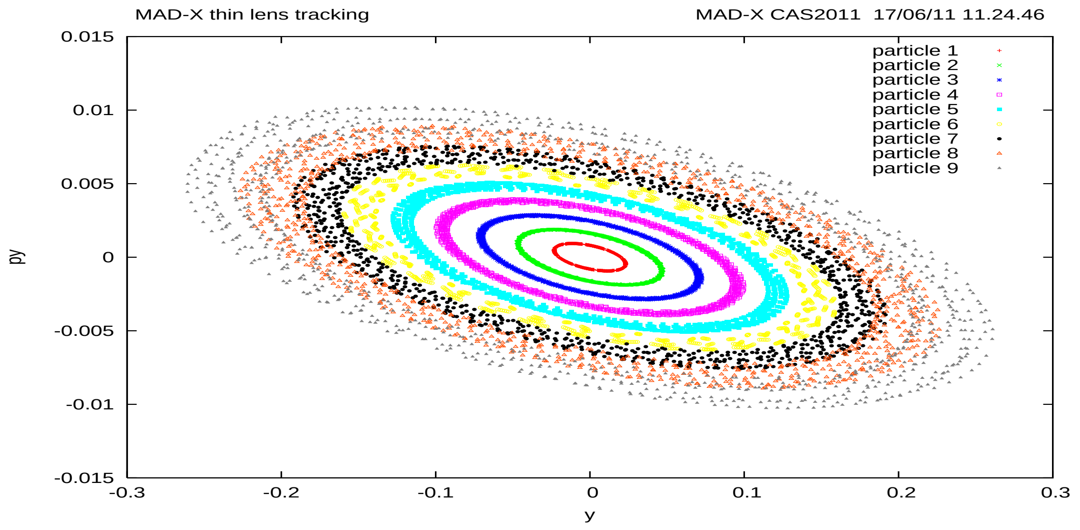

Figure 24: Tracking output - thin lenses with MAD-X. Plot shows vertical phase space.

```
use, period=cascell6;


match, sequence=cascell6;
   vary,name=kqf, step=0.00001;
   vary,name=kqd, step=0.00001;
   global,sequence=cascell6,Q1=6.690;
   global,sequence=cascell6,Q2=6.700;
   Lmdif, calls=10, tolerance=1.0e-21;
endmatch;

match, sequence=cascell6;
   vary,name=ksf, step=0.00001;
   vary,name=ksd, step=0.00001;
   global,sequence=cascell6,DQ1=0.0;
   global,sequence=cascell6,DQ2=0.0;
   Lmdif, calls=10, tolerance=1.0e-21;
```

```
endmatch;


select,flag=twiss,column=name,s,mux,betx,alfx,dx,dpx,muy,bety,alfy,dy;
twiss,save,centre,file=twiss.out;
plot, haxis=s, vaxis=betx;
plot, haxis=s, vaxis=bety;
plot, haxis=s, vaxis=dx,vmin=0.0,vmax=15.0;
plot, haxis=s, vaxis=x;


!ksf = 1.125*ksf;
!ksd = 1.125*ksd;

ptc_create_universe;
ptc_create_layout,model=2,method=6,nst=10,exact;
ptc_align;

select, flag=ptc_twiss, clear;
select, flag=ptc_twiss, column=name,s,x,y,mu1,mu2,mu3,beta11,beta21,beta12,b
ptc_twiss,icase=5, closed_orbit, betx = 106.7496412, bety = 30.3134712 ,
                   alfx = -1.74420269, alfy = 0.5247598117,
                   dx = 8.612042418,
                   dpx = 0.1533776324,
file=twiss.ptc1;


ptc_create_universe;
ptc_create_layout,model=2,method=6,nst=10,exact;

ptc_start, x= 2e-2, px=0, y= 2e-2, py=0;
ptc_start, x= 4e-2, px=0, y= 4e-2, py=0;
ptc_start, x= 6e-2, px=0, y= 6e-2, py=0;
ptc_start, x= 8e-2, px=0, y= 8e-2, py=0;
ptc_start, x=10e-2, px=0, y=10e-2, py=0;
ptc_start, x=12e-2, px=0, y=12e-2, py=0;
ptc_start, x=14e-2, px=0, y=14e-2, py=0;
ptc_start, x=16e-2, px=0, y=16e-2, py=0;
ptc_start, x=18e-2, px=0, y=18e-2, py=0;

ptc_track,icase=4,closed_orbit,dump,
       turns=1000 ,ffile=1; //onetable, turns=1000, norm_no=4; norm_out

plot, file="PTC_track",table=track,haxis=x,vaxis=px,
      particle=1,2,3,4,5,6,7,8,9, colour=1000, multiple, symbol=3;
plot, file="PTC_track",table=track,haxis=y,vaxis=py,
      particle=1,2,3,4,5,6,7,8,9, colour=1000, multiple, symbol=3;

ptc_track_end;
ptc_end;
```

```
stop;
```

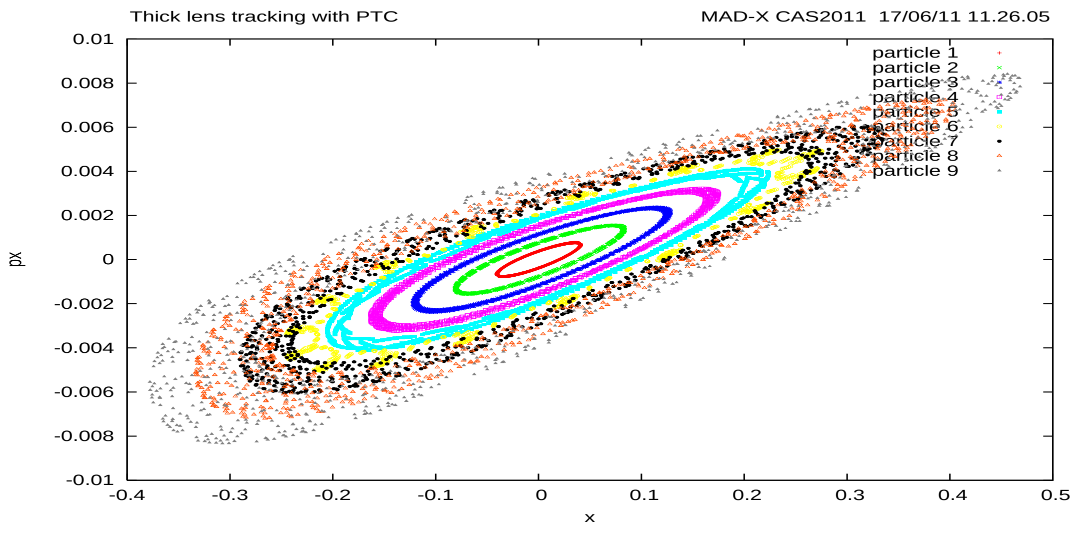

Figure 25: Tracking output- thick lenses with PTC. Plot shows phase horizontal space.

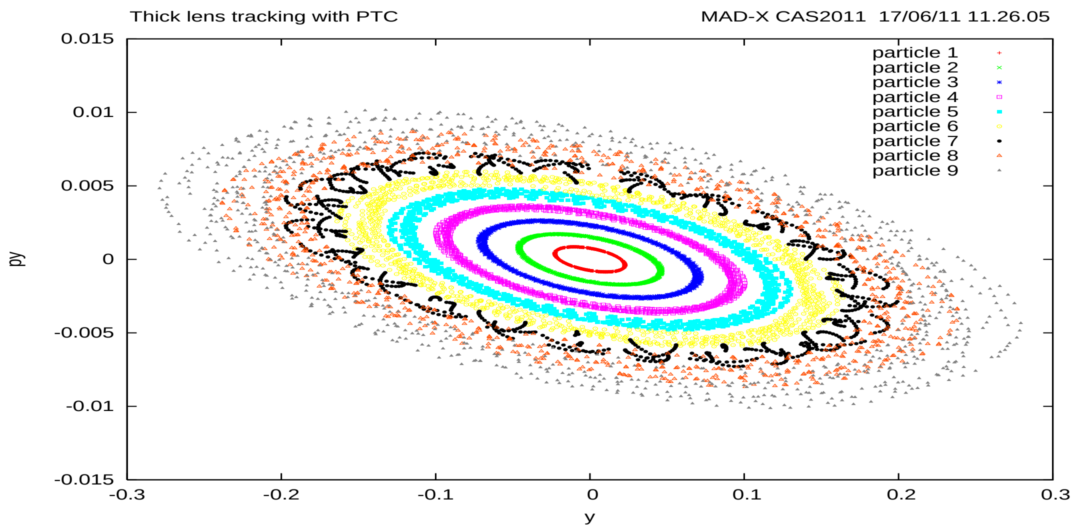

Figure 26: Tracking output- thick lenses with PTC. Plot shows vertical phase space.

References

[1] *The MAD-X Home Page, version January 2013*,
$http : //frs.home.cern.ch/frs/Xdoc/mad - X.html$.

[2] Proceedings, CAS 2003, Zeuthen, CERN-2006-002, ISBN 92-9083-26703, (2006).

[3] B. Holzer, *Lattice cells*, this school, and [2].

[4] B. Holzer, *Insertions*, this school, and [2].

[5] O. Brüning, *Linear imperfections*, this school, and [2].

[6] O. Brüning, *Non-linear imperfections*, this school, and [2].

[7] W. Herr, *MAD-X for pedestrian*, this school, and [2].

[8] W. Herr, *A MAD-X Primer*, CERN-AB-2004-027-ABP, and [2].

[9] T. Pieloni, *Beam-Beam Effects*, this school, and [2].

[10] $http : //cern.ch/Werner.Herr/CAS2013/$

[11] *CAS 2013, Optics Design*, CD-ROM (LINUX, WINDOWS) with the material of the course, CERN, Geneva 2013.