CI-Beam-105

Lattice Design and Computational Dynamics III

Dr Öznur Apsimon
The University of Manchester
The Cockcroft Institute of Accelerator Science and Technology
and Dr Rob Apsimon
Lancaster University
The Cockcroft Institute of Accelerator Science and Technology
Contact
oznur.apsimon@manchester.ac.uk
r.apsimon@lancaster.ac.uk

Scope of MADX in this lecture

- Description of the basic concepts and the language
- How to compute optical functions
- Perform “matching”:
  - Beam dimensions
  - Tune, chromaticity
- Machine with imperfections and their correction
- Design of insertions
  - e.g. Dispersion suppressor
- Advanced optimisation or particle tracking code (PTC)

Why MADX?

- A long line of development
- Used at CERN since more than 20 years for machine design and simulation (PS, SPS, LEP, LHC, CLIC, beam lines…)
- Existing versions MAD8, MAD9, MADX (with PTC)
- Increasing and organised support and website in recent years: http://mad.web.cern.ch/mad/
- Running on all systems
- Source is free and easy to extend
- Input easy to understand

Typical Input

- Description of the machine
  - ➡Definition of each machine element
  - ➡Attributes of elements
  - ➡Position of elements
- Description of the beam(s)
- Commands regarding the desired process.

How does it work?

- MADX is an interpreter
  - accepts and executes statements
  - statement can be assignments or expressions
  - can be used interactively or in batch mode
- MADX has many features of a programming language (loops, if conditions, macros, subroutines ...)

MADX Input Language

- Strong resemblance to C/C++
- All statements must be terminated with **;**
- Lines can be commented out with **** or **!**
- Arithmetic expressions, including basic functions (**exp, log, sin, cosh, ...**)
- Built-in random number generators for various distributions
- Deferred expressions (**:=**)
- Predefined constants (**clight, e, pi, mp, me, ...**)

MADX Conventions

- Not case sensitive
- Elements are placed along the reference orbit (variable **s**).
- Horizontal (assumed bending plane) and vertical variables are **x** and **y**
- Describes a local coordinate system moving along s
    - x=y=0 follows the curvilinear system
- MADX variables are floating point numbers (double precision)
- Variables can be used in expressions
    - ANGLE = 2*PI/NBEND
- The assignment symbols = and := (deferred assignment) have very different behaviour
    - DX = GAUSS()*1.5E-3;

      The value is calculated **once** and kept in DX
    - DX := GAUSS()*1.5E-3;

      The value is calculated **every time** DX is used.

Let’s Try

```
x: ==> angle = 2*pi/1232;
x: ==> value, angle;
x: ==> value, sin(1,0)*2;
x: ==> dx = gauss()*2.0;
x: ==> value, dx;
x: ==> value, dx;
x: ==> dx := gauss()*2.0;
x: ==> value, dx;
x: ==> value, dx;
```

Let's Try

Batch mode:

```
> madx

X: ==> call, file=my_file.madx;
```

```
> ./madx < my_file.madx    (unix)

>.\madx < my_file.madx     (Windows)
```

MADX Input Statements

- Typical assignments
  - Properties of machine elements
  - Setting up a lattice
  - Definition of beam properties (particle type, energy, emittance etc.)
  - Assignment of errors and imperfections
- Typical actions
  - Compute lattice functions
  - Correct machine errors
  - Matching of subsections

Definition of machine elements

- All machine elements have to be described
- They can be described individually or
- as a family **("class")** of elements (i.e. all elements with the same properties)
- All elements can have unique names (not necessarily)
- MADX "keywords" are used to define the type of an element
- General format:

name:keyword, attributes

Example: Definition of machine elements

- Dipole (bending) magnet

```
MBL: SBEND, L=10.0, ANGLE = 0.0145444;
```

- Quadrupole (focusing) magnet

```
MQ: QUADRUPOLE, L=3.3, K1 = 1.23E-02;
```

- Sextupole magnet

```
ksf = 0.00156;
MSF: SEXTUPOLE, K2 := ksf, L=1.0;
```

Example: Definition of strength of the elements

- **Dipole (bending) magnet**

$$k_0 = \frac{1}{p/c} B_y[T] \quad \left(= \tfrac{1}{\rho} = \tfrac{angle}{\ell}[rad/m]\right)$$

```
DIP01: SBEND, L=10.0, ANGLE = angle, K0=k0;
DIP02: MBL;     ! belongs to MBL family
DIP03: MBL;     ! an instance of MBL class
```

- **Quadrupole (focusing) magnet**

$$k_1 = \frac{1}{p/c} \frac{\partial B_y}{\partial x} [T/m] \quad (= \tfrac{1}{\ell . f})$$

```
MQA: QUADRUPOLE, L=3.3, K1 =k1;
```

Example: Definition of strength of the elements

- **Sextupole magnet**

$$k_2 = \frac{1}{p/c}\frac{\partial^2 B_y}{\partial x^2}[T/m^2]$$

```
KLSF = k2;
MSXF: SEXTUPOLE, L=1.1, K2 = KLSF;
```

- **Octupole magnet**

$$k_3 = \frac{1}{p/c}\frac{\partial^3 B_y}{\partial x^3}[T/m^3]$$

```
KLOF = k3;
MOF: OCTUPOLE, L=1.1, K3 = KLOF;
```

Example: Definition of machine elements

- **LHC dipole magnet**

```
length = 14.3;
B = 8.33;
PTOP = 7.0E12;
ANGLHC = B*clight*length/PTOP;
MBLHC: SBEND, L=length, ANGLE = anglhc;
```

```
ANGLHC = 2*pi/1232;
MBLHC: SBEND, L=length, ANGLE = anglhc;
```

Let's Try

```
      > madx
X: ==> length = 14.3;
X: ==> B = 8.33;
X: ==> PTOP = 7.0E12;
X: ==> ANGLHC = B*clight*length/PTOP;
X: ==> MBLHC: SBEND, L = LENGTH, ANGLE = ANGLHC;
X: ==> value, mblhc->angle;
```

Thick and Thin Elements

- **Thick elements:** So far examples were thick elements (or lenses)
- Specify length and strength
    - +More precise, path lengths and fringe fields
    - –Not symplectic in tracking (energy and emittance is not exactly conserved).

Thick and Thin Elements

- **Thin elements:** Specified as elements with zero length
- Specified field integration (k_0.L, k_1.L, k_2.L, ...):
    + Easy to use
    + Uses amplitude dependent kicks -> always "symplectic"
    + Used for tracking
    – Path lengths are not precise
    – Fringe fields are not precise
    – Maybe problematic for small machines

Thick and Thin Elements

Special MADX element: multipoles

- **Multipole:** General element of zero length, can be used one or more components of any order:

multip: multipole, knl := {k_{n0}L, k_{n1}L, k_{n2}L, k_{n3}L, ...};

---> knl = k_n . L (components of nth order)

- **Very simple to use**

mul1: multipole, knl := {0, k_1L, 0, 0, ...};
(is equivalent of definition of a quadrupole)

mul0: multipole, knl := {angle, 0, 0, ...};
(is equivalent of definition of a dipole)

Thick and Thin Elements

- All exercises in this course will be with thin lenses

```
my_dipol: multipole, knl := {angle, 0, 0, ...};
my_quad: multipole, knl := {0, k1L, 0, ...};
```

Definition of Sequence

- Positions of the elements are defined in a **"sequence"** file with their names
- A position can be defined at the **centre**, **exit** or **entrance** of an element
- can be defined as absolute or relative numbers

```
ciwinter_sps: SEQUENCE, REFER=CENTRE,
L=6912;

...

...

ENDSEQUENCE;
```

position of all elements in the sequence are defined here.

Definition of Sequence

```
cassps: SEQUENCE, refer=centre, l=6912; ...
...
MBL01: MBLA, at = 102.7484;

MBL02: MBLB, at = 112.7484;
MQ01: MQA, at = 119.3984;
BPM01: BPM, at = 1.75, from MQ01;
COR01: MCV01, at = LMCV/2 + LBPM/2

MBL03: MBLA, at = 126.3484;
MBL04: MBLB, at = 136.3484;
MQ02: MQB, at = 142.9984;
BPM02: BPM, at = 1.75, from MQ02;
COR02: MCV02, at = LMCV/2 + LBPM/2, from BPM02; ...

...

ENDSEQUENCE;
```

Definition of Sequence

A snippet from the SPS sequence

```
SPS : SEQUENCE,          L = 6911.5038;

 BEGI.10010       : STARTSPS    , AT = 0;
 QF.10010         : QF          , AT = 1.5425      , SLOT_ID = 2361953;
 MBA.10030        : MBA         , AT = 6.575       , SLOT_ID = 2361954;
 MBA.10050        : MBA         , AT = 13.235      , SLOT_ID = 2361955;
 MBB.10070        : MBB         , AT = 19.885      , SLOT_ID = 2361956;
 MBB.10090        : MBB         , AT = 26.525      , SLOT_ID = 2361957;
 VVSA.10101       : VVSA        , AT = 29.9385     , SLOT_ID = 2361958;

 ...

 ...

 MBA.63570        : MBA         , AT = 6899.3611   , SLOT_ID = 2363841;
 MBA.63590        : MBA         , AT = 6906.0211   , SLOT_ID = 2363842;
 LOE.63602        : LOE         , AT = 6909.8401   , SLOT_ID = 2363843;
 LSF.63605        : LSF         , AT = 6910.5088   , SLOT_ID = 2363844;
 MDH.63607        : MDH         , AT = 6910.9838   , SLOT_ID = 2363845;
 BPH.63608        : BPH         , AT = 6911.2713   , SLOT_ID = 2363846;
 END.10010        : ENDOFSPS    , AT = 6911.5038;
ENDSEQUENCE;
```

Definition of Sequence

```
circum = 6912;

// bending magnets as thin lenses
mbsps: multipole,knl={0.007272205};

// quadrupoles and sextupoles
qf: quadrupole,l=3.085,k1 =  0.0146315;
qd: quadrupole,l=3.085,k1 = -0.0146434;
lsf: sextupole,l=1.0,  k2 =  1.9518486E-02;
lsd: sextupole,l=1.0,  k2 = -3.7618842E-02;

// monitors and orbit correctors
bpm: monitor,l=0.1;
ch:  hkicker,l=0.1;
cv:  vkicker,l=0.1;
```

Definition of Sequence

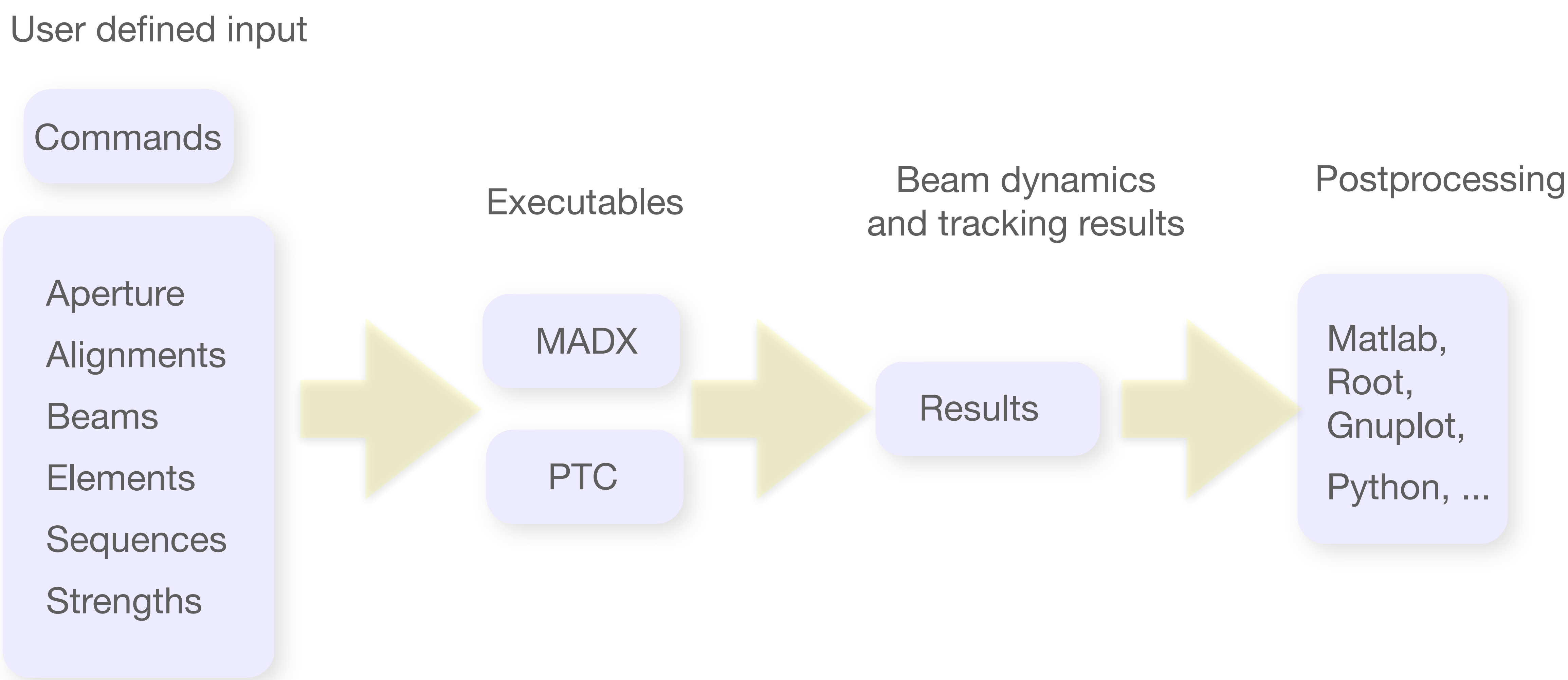
```
cassps: sequence, l = circum;
start_machine: marker, at = 0;
qf,  at = 1.5425;
lsf, at = 3.6425;
ch,  at = 4.2425;
bpm, at = 4.3425;
mbsps, at = 5.0425;
mbsps, at = 11.4425;
mbsps, at = 23.6425;
mbsps, at = 30.0425;
qd, at = 33.5425;
lsd, at = 35.6425;
cv,  at = 36.2425;
bpm, at = 36.3425;
...

...

mbsps, at = 6903.6425;
mbsps, at = 6910.0425;
end_machine: marker, at = 6912;
endsequence;
```

Definition of Sequence

```
circum=6912.0; // define the total length
ncell = 108; // define number of cells
lcell = circum/ncell;
// all magnets as multipoles
mbsps: multipole, knl={2.0*pi/(2*ncell)};
qfsps: multipole, knl={0.0, 4.36588E-02};
qdsps: multipole, knl={0.0,-4.36952E-02};
// sequence declaration;
cassps: sequence, refer=centre, l=circum;
   n = 1;
   while (n <= ncell) {
   qfsps: qfsps,   at=(n-1)*lcell;
   mbsps: mbsps,   at=(n-1)*lcell+16.0;
   qdsps: qdsps,   at=(n-1)*lcell+32.00;
   mbsps: mbsps,   at=(n-1)*lcell+48.00;
   n = n + 1;
}
endsequence;
```

SPS sequence, elements in a loop

MADX and post-processing flow chart



Example case

```
TITLE, 'MAD-X Test';

// Read the lattice from the sequence file.
call file="sps.seq";
option,-echo,-thin_foc;

// Define the beam.
Beam, particle = proton, sequence=ci_sps, energy = 450.0;

// Read the sequence called ci_sps.
use, sequence=ci_sps;

// Select the parameters to be calculated.
select,flag=twiss,column=name,s,betx,bety;

// Run TWISS command to calculate the Twiss parameters.
// Calculate parameters in the centre of the elements
// and write the results in twiss.out file.
twiss,save,centre,file=twiss.out;

// Plot the horizontal and vertical dispersion
// between 10th and 16th defocusing quadrupole.

plot, haxis=s, vaxis=betx, bety,colour=100,range=qd[10]/qd[16];
plot, haxis=s, vaxis=dx, colour=100,range=qd[10]/qd[16];

survey,file=survey.out;

stop;
```

Example case

```
TITLE, 'MAD-X Test';

// Read the lattice from the sequence file.
call file="sps.seq";
option,-echo,-thin_foc;

// Define the beam.
Beam, particle = proton, sequence=hpfbu_sps, energy = 450.0;

// Read the sequence called ci_sps.
use, sequence=ci_sps;

// Select
select,fl

// Run Tw
// Calcul
// and wr
twiss,sav

// Plot t
// betwee

plot, hax
plot, hax

survey,file=survey.out;

stop;
```

- **Call the sequence file defining the machine**
  - call, file="sps.seq";

```
TITLE, 'MAD-X Test';

// Read the lattice from the sequence file.
call file="sps.seq";
option,-echo,-thin_foc;

// Define the beam.
Beam, particle = proton, sequence=ci_sps, energy = 450.0;

// Read the sequence called ci_sps.
use, sequence=ci_sps;

// Select
select,fla

// Run TWI
// Calcula
// and wri
twiss,save

// Plot th
// between 10th and 16th defocusing quadrupole.

plot, haxis=s, vaxis=betx, bety,colour=100,range=qd[10]/qd[16];
plot, haxis=s, vaxis=dx, colour=100,range=qd[10]/qd[16];

survey,file=survey.out;

stop;
```

- **Define beam type and properties**

```
TITLE, 'MAD-X Test';

// Read the lattice from the sequence file.
call file="sps.seq";
option,-echo,-thin_foc;

// Define the beam.
Beam, particle = proton, sequence=ci_sps, energy = 450.0;

// Read the sequence called ci_sps.
use, sequence=ci_sps;

// Select the parameters to be calculated.
select,flag=twiss,column=name,s,betx,bety;

// Run TWI
// Calcula
// and wri
twiss,save

// Plot th
// between

plot, haxi
plot, haxi

survey,file

stop;
```

- **Activate your machine by "using" the sequence**
  - USE, sequence = ci_sps;
  - There can be other sequences in "sps.seq"
  - USE command activates the sequence named

Example case

```
TITLE, 'MAD-X Test';

// Read the lattice from the sequence file.
call file="sps.seq";
option,-echo,-thin_foc;

// Define the beam.
Beam, particle = proton, sequence=ci_sps, energy = 450.0;

// Read the sequence called ci_sps.
use, sequence=ci_sps;

// Select the parameters to be calculated.
select,flag=twiss,column=name,s,betx,bety;

// Run TWISS command to calculate the Twiss parameters.
// Calculate parameters
// and write the results
twiss,save,centre,file=t

// Plot the horizontal a
// between 10th and 16th

plot, haxis=s, vaxis=bet
plot, haxis=s, vaxis=dx,

survey,file=survey.out;

stop;
```

- **SELECT the parameters to work with for**
  - Calculating of the Twiss parameter
  - Saving the lattice functions
  - Plotting
  - …

```
TITLE, 'MAD-X Test';

// Read the lattice from the sequence file.
call file="sps.seq";
option,-echo,-thin_foc;

// Define the beam.
Beam, particle = proton, sequence=ci_sps, energy = 450.0;

// Read the sequence called ci_sps.
use, sequence=ci_sps;

// Select the parameters to be calculated.
select,flag=twiss,column=name,s,betx,bety;

// Run TWISS command to calculate the Twiss parameters.
// Calculate parameters in the centre of the elements
// and write the results in twiss.out file.
twiss,save,centre,file=twiss.out;

// Plot the horizontal and vertical dispers
// between 10th and 16th defocusing quadru

plot, haxis=s, vaxis=betx, bety,colour=100,
plot, haxis=s, vaxis=dx, colour=100,range=

survey,file=survey.out;

stop;
```

- **Start the calculation**
  - **twiss;** or
  - **twiss, file=output;** or
  - **twiss, sequence=hpfbu_sps;**

Example case

```
TITLE, 'MAD-X Test';

// Read the lattice from the sequence file.
call file="sps.seq";
option,-echo,-thin_foc;

// Define the beam.
Beam, particle = proton, sequence=ci_sps, energy = 450.0;

// Read the sequence called ci_sps.
use, sequence=ci_sps;

// Select the parameters to be calculated.
select,flag=twiss,column=name,s,betx,bety;

// Run TWISS command to ca
// Calculate parameters in
// and write the results i
twiss,save,centre,file=twi

// Plot the horizontal and vertical dispersion
// between 10th and 16th defocusing quadrupole.

plot, haxis=s, vaxis=betx, bety,colour=100,range=qd[10]/qd[16];
plot, haxis=s, vaxis=dx, colour=100,range=qd[10]/qd[16];

survey,file=survey.out;

stop;
```

- **Plot beta and dispersion functions**

```
TITLE, 'MAD-X Test';

// Read the lattice from the sequence file.
call file="sps.seq";
option,-echo,-thin_foc;

// Define the beam.
Beam, particle = proton, sequence=ci_sps, energy = 450.0;

// Read the sequence called ci_sps.
use, sequence=ci_sps;

// Select the parameters to be calculated.
select,flag=twiss,column=name,s,betx,bety;

// Run TWISS command to calculate the Twiss parameters.
// Calculate parameters in the centre of the elements
// and write the results in twiss.out file.
twiss,save,centre,file=twi

// Plot the horizontal anc
// between 10th and 16th c

plot, haxis=s, vaxis=betx, bety,colour=100,range=qd[10]/qd[16];
plot, haxis=s, vaxis=dx, colour=100,range=qd[10]/qd[16];

survey,file=survey.out;

stop;
```

- **Survey the geometry of the orbit. Is it closed?**

MADX results on your command line

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

       length          orbit5             alfa          gammatr
         6912              -0   0.001504942753      25.77745337

           q1             dq1          betxmax            dxmax
  26.57999204     -1.67838253      108.7763569       2.44661758

        dxrms          xcomax           xcorms               q2
  1.830638952               0                0      26.62004577

          dq2         betymax            dymax            dyrms
 -1.680294089     108.7331749                0                0

       ycomax          ycorms           deltap          synch_1
            0               0                0                0

      synch_2         synch_3          synch_4          synch_5
            0               0                0                0
```

Example case

Header part of an example output file

```
@ NAME             %05s "TWISS"
@ TYPE             %05s "TWISS"
@ SEQUENCE         %09s "HPFBU_SPS"
@ PARTICLE         %06s "PROTON"
@ MASS             %le        0.938272013
@ CHARGE           %le                  1
@ ENERGY           %le                450
@ PC               %le        449.9990218
@ GAMMA            %le        479.6050546
@ KBUNCH           %le                  1
@ BCURRENT         %le                  0
@ SIGE             %le                  0
@ SIGT             %le                  0
@ NPART            %le                  0
@ EX               %le                  1
@ EY               %le                  1
@ ET               %le                  1
@ LENGTH           %le               6912
@ ALFA             %le     0.001504942753
@ ORBIT5           %le                 -0
@ GAMMATR          %le        25.77745337
@ Q1               %le        26.57999204
@ Q2               %le        26.62004577
@ DQ1              %le        -1.67838253
@ DQ2              %le       -1.680294089
@ DXMAX            %le         2.44661758
@ DYMAX            %le                  0
@ XCOMAX           %le                  0
@ YCOMAX           %le                  0
@ BETXMAX          %le        108.7763569
```

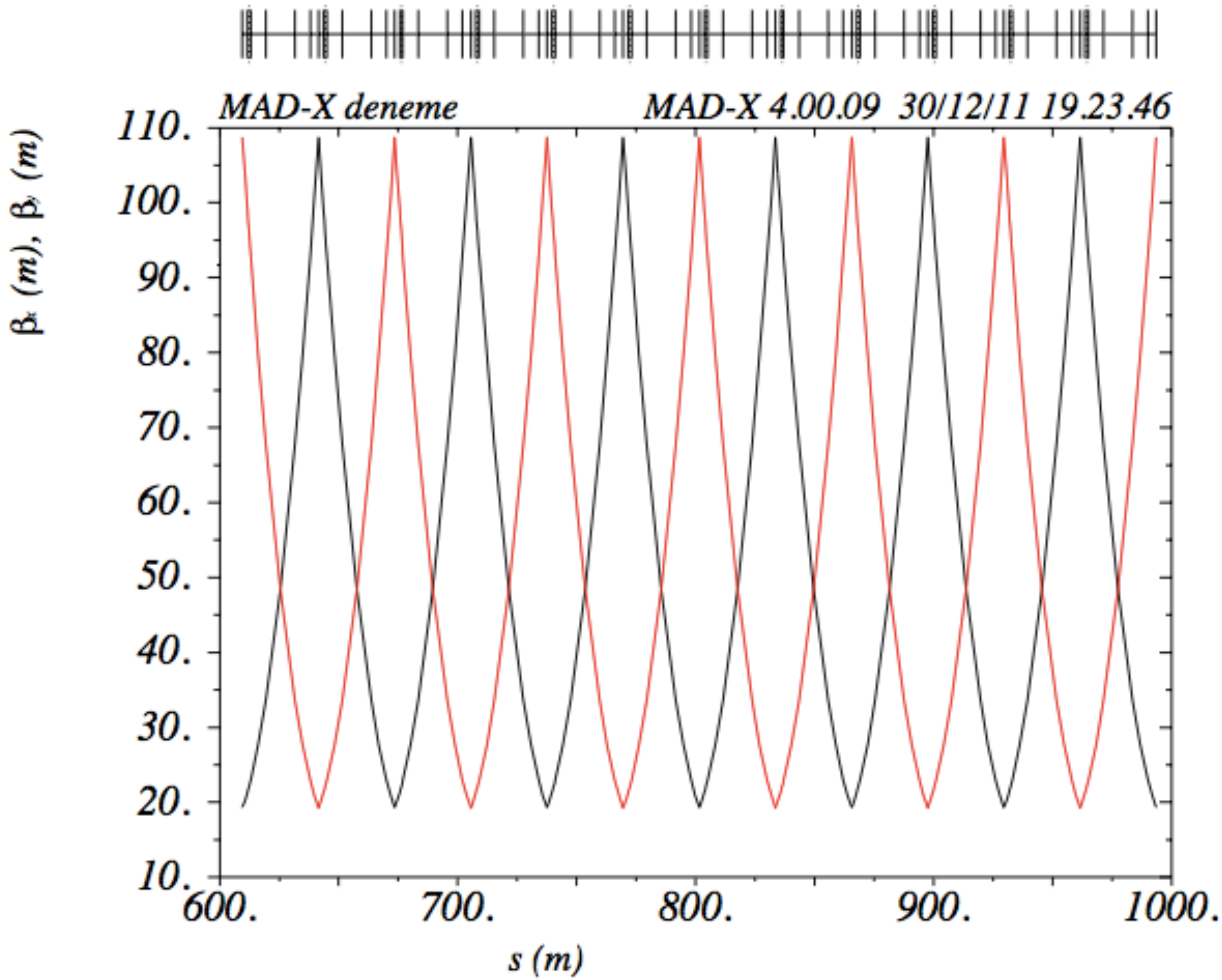
- twiss.out output file is created after running the input file with "madx" extension.

1

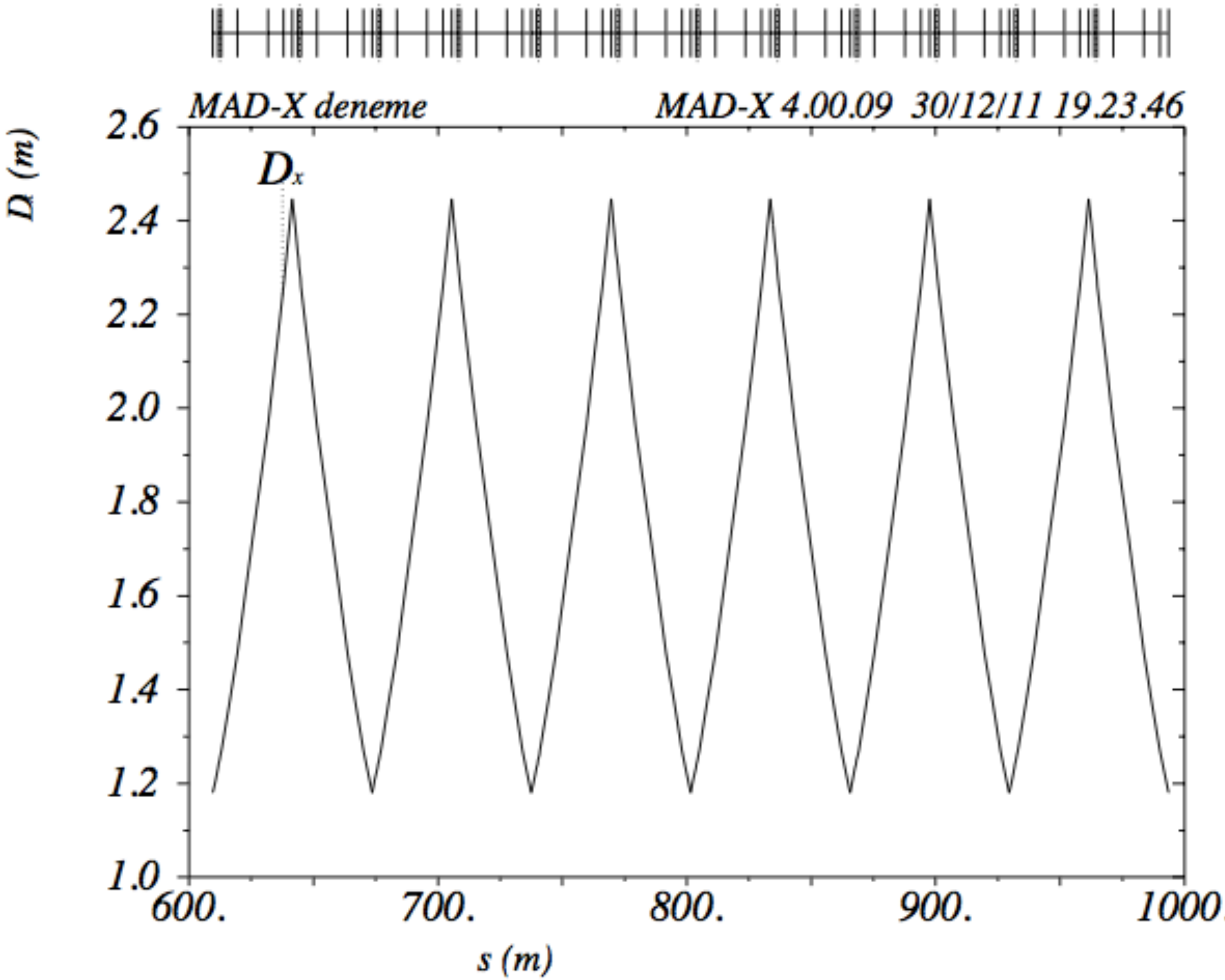
```
* NAME                              S            BETX            BETY
$ %s                              %le             %le             %le
 "HPFBU_SPS$START"                  0     101.5961579     20.70328425
 "START_MACHINE"                    0     101.5961579     20.70328425
 "DRIFT_0"                    0.77125     105.1499566     19.94571028
 "QF"                          1.5425     108.7763569     19.26082066
 "DRIFT_1"                     2.5925     103.8571423     20.21112973
 "LSF"                         3.6425     99.07249356     21.29615787
 "DRIFT_2"                  3.9424975     97.73017837      21.6309074
 "CH"                          4.2425     96.39882586     21.97666007
 "DRIFT_3"                     4.2925     96.17800362     22.03535424
 "BPM"                         4.3425     95.95748651      22.0943539
 "DRIFT_4"                  4.6925025      94.4223997     22.51590816
 "MBSPS"                       5.0425     92.90228648     22.95242507
 "DRIFT_5"                     8.2425     79.69728195     27.63752778
 "MBSPS"                      11.4425     67.74212222      33.5738988
 "DRIFT_6"                    17.5425     48.41469349     48.35614376
 "MBSPS"                      23.6425      33.6289371     67.68523387
 "DRIFT_5"                    26.8425     27.68865546      79.6433337
 "MBSPS"                      30.0425     22.99821861     92.85270185
 "DRIFT_7"                    31.7925     20.96178735     100.6058286
 "QD"                         33.5425     19.29915001     108.7331749
 "DRIFT_1"                    34.5925     20.25187715     103.8118608
```

2

MADX graphical output: Beta functions

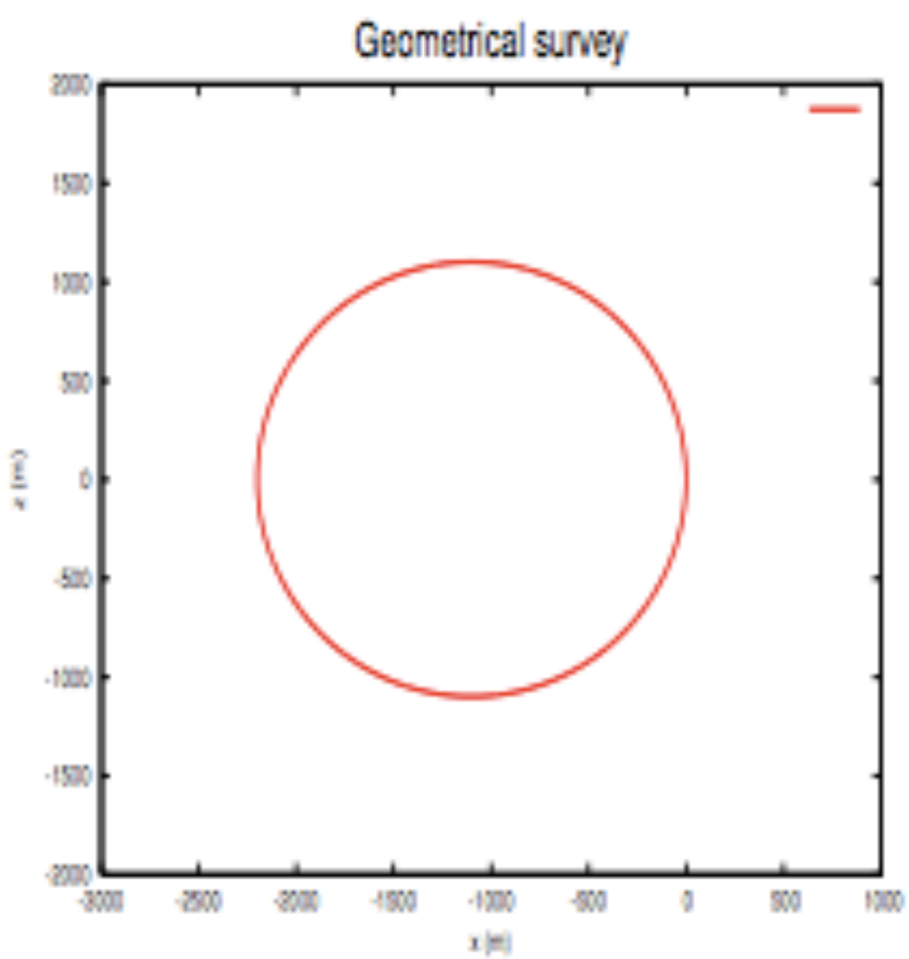


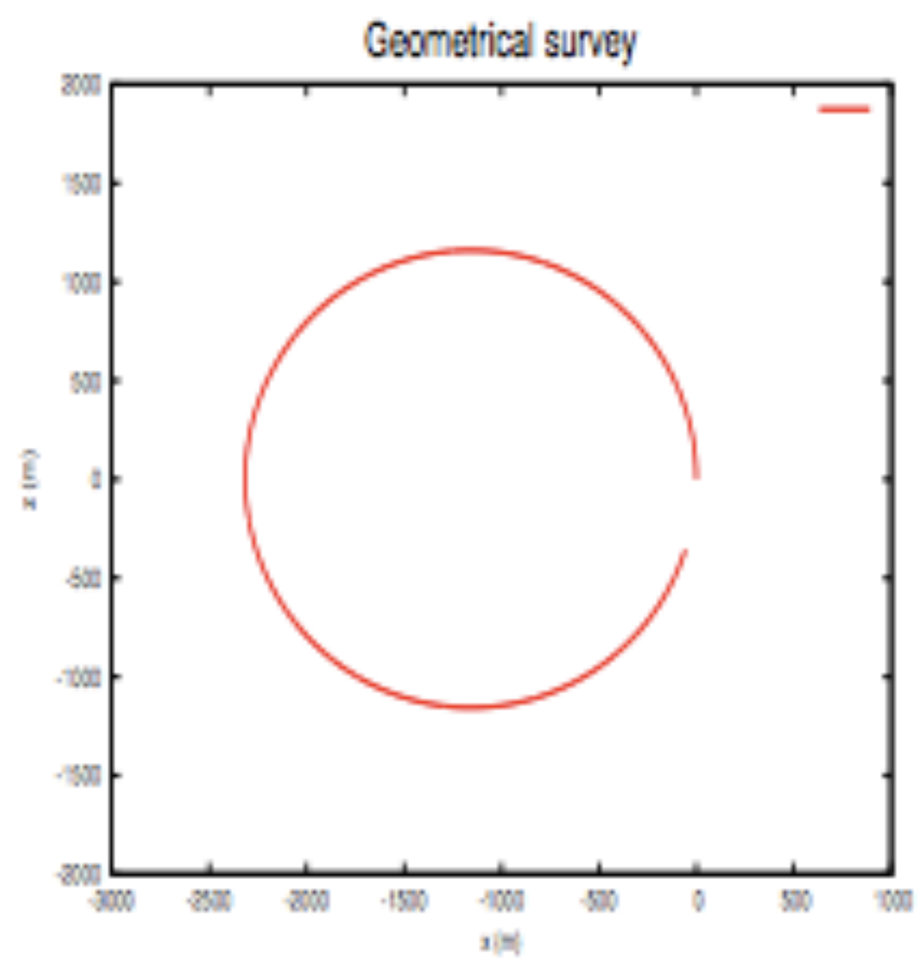
MADX graphical output: Dispersion function



MADX graphical output: Survey

- Output gives x, y, z and theta
- Plot x as a function of z to survey the close orbit.





Optical Matching

Main applications:

- **Setting global optical parameters:**
  (tune, chromaticity ...)

- **Setting local optical parameters**
  (beta function, dispersion ...)

- **Orbit correction**

Global Matching

- Adjust respective multipole strengths to get desired parameters
- Define the properties required and the elements to vary.
- Examples for global parameters:
  - Q1, Q2: Horizontal and vertical tune.
  - dQ1, dQ2: Horizontal and vertical chromaticity.

Global Matching

- Match horizontal (Q1) and vertical (Q2) tunes.
- Vary the quadrupole strengths (kqf, kqd).

```
match, sequence=hpfbu_sps;
   vary,name=kqf, step=0.00001;             → to vary
   vary,name=kqd, step=0.00001;             → to vary
   global,sequence=hpfbu_sps,Q1=26.58;      → target value
   global,sequence=hpfbu_sps,Q2=26.62;      → target value
   Lmdif, calls=10, tolerance=1.0e-21;
endmatch;
```

sps_matching.madx