CAS Course on Optics Design

Trondheim, August 2013

Information at:
http://cern.ch/Werner.Herr/CAS2013

CAS 2013 course on optics design

- **Aims:**
  - **From the lectures to praxis**
  - **Design a realistic machine optics**
- **Not a lecture, but following a series of steps (as exercises) applying what was learned in previous lectures**
- **Done by you in close collaboration with the tutors and your colleagues (this is not an examination !)**

Procedure and basic steps

- Introduction to MAD-X
- Work on 8 exercises:
  - Design of periodic machine with desired properties (1-2)
  - Correction of chromaticity and orbit imperfections (3-5)
  - Design of a dispersion suppressor (6)
  - Design of a β-insertion (low and high β, for experiments, collimation etc.) (7)
  - Particle tracking to study stability (8)

Available tools

- Individual computers and accounts
- LINUX operating system
- You have MAD-X, compilers, gnuplot ...

  - Bonus material:
    You get all your solutions and our suggested solutions together with the MAD-X binary and source code after the school on the memory stick you have.

Available to help

Werner Herr,
Bernhard Holzer,
Verena Kain,
Oliver Brüning (part time),
Yannis Papaphilippou (part time),

for computers: Vegard Moen, Egil Holvik

How to get in ?

- For LINUX: see instructions or ask for help or download from website:
  - http://cern.ch/Werner.Herr/CAS2013/bin
  - http://cern.ch/Werner.Herr/CAS2013/doc
  - http://cern.ch/Werner.Herr/CAS2013/examples
- For Windows: download executable and examples from website:
  - http://cern.ch/Werner.Herr/CAS2013/exe
  - http://cern.ch/Werner.Herr/CAS2013/doc
  - http://cern.ch/Werner.Herr/CAS2013/examples

Introduction to MADX

Werner Herr, CERN

For all MAD details:

(http://cern.ch/mad)

see also:

MADX primer

Where you find all that:

Documentation: /COURSE/doc

Your exercises in: /COURSE/doc/problems.pdf

Examples in: /COURSE/examples

Executable:
/COURSE/bin/madx (LINUX)
/COURSE/exe/madx.exe (WINDOWS)

Everything also at:
http://cern.ch/Werner.Herr/CAS2013

Werner Herr, MAD introduction, CAS 2013, Trondheim

MADX - part 1 (today)

- Description of the basic concepts and the language
- Define a machine and compute optical functions
- Get the parameters you want
  - Beam dimensions
  - Tune, chromaticity

Required lectures: Transverse dynamics, Lattice cells

MADX - part 2 (as we proceed ..)

- Machines with imperfections and corrections
- Design of insertions
  - Dispersion suppressor
  - Low β insertion
- Particle tracking

Required lectures: Insertions, Non-linear dynamics

General purpose lattice programs

- For circular machines, beam lines or linacs
- Calculate optics parameters from machine description
- Compute (match) desired quantities
- Simulate and correct machine imperfections
- Simulate beam dynamics

→ Used in this course: MADX

What is MADX ?

- The latest version in a long line of development
- Used at CERN since more than 30 years for machine design and simulation (PS, SPS, LEP, LHC, future linacs, beam lines)
- (still) Existing versions:
  MAD8, MAD9, MADX (version 5, with PTC)
- Mainly designed for large projects (LEP, LHC, CLIC ..)

Why we use MADX here ?

This is not a large project, but:

- Multi purpose:
  - From early design to final evaluation
- Running on all systems
- Source is free and easy to extend
- Input easy to understand
- Easy to understand what is happening:
  - No hidden or invisible actions or computations
  - Every computational step is explicit

Data required by an optics program ?

- Description of the machine:
  - Definition of each machine element
  - Attributes of the elements
  - Positions of the elements
- Description of the beam(s)
- Directives (what to do ?)

How does MAD get and use this information ?

- MAD is an ”interpreter”:
  - Accepts and executes statements
  - Statements can be assignments, expressions or initiate complex actions
  - Can be used interactively or in batch
  - → Reads statements from the input stream or a file (but has no GUI)
- Many features of a programming language (loops, if conditions, macros, subroutines ...)

MAD input language

- Strong resemblance to ”C” language
- All statements are terminated with ;
- Comment lines start with: // or !
- Arithmetic expressions, including basic functions (exp, log, sin, cosh ...)
- Immediate (=) and deferred expressions (:=)
- In-built random number generators for various distributions
- Predefined constants (clight, e, pi, m_p, m_e ...)

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
  - i.e. x = y = 0 follows the curvilinear system (reference orbit)

More conventions

- MAD variables are floating point numbers (double precision)
- Variables can be used in expressions:
  - ANGLE = 2*PI/NBEND;
  - AIP = ATAN(SX1/SX2);
- The assignment symbols = and := have a very different behaviour (here random number generator)!
  - DX = GAUSS()*1.5E-3;
    The value is computed once and kept in DX
  - DX := GAUSS()*1.5E-3;
    The value is recomputed every time DX is used

How to use MADX ?

```
> madx
X: ==> angle = 2*pi/1232;
X: ==> value, angle;
X: ==> value,asin(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;
```


How to use MADX ?

if you store everything in a file: my.file

```
> madx
X: ==>
```

How to use MADX ?

if you store everything in a file: my.file

> **madx**

X: ==> call, file=my.file; (WINDOWS or LINUX)

How to use MADX ?

if you store everything in a file: my.file

> **madx**

X: ==> call, file=my.file; (WINDOWS or LINUX)

> **madx < my.file (LINUX)**

⚠ Warning ! ⚠

- **For WINDOWS users:**
  - **The input file must be a plain text (ASCII) file !**
  - **NOT a WORD, POWERPOINT or EXCEL file ...**

MAD input statements (what we need)

- Typical assignments:
  - Properties of machine elements
  - Set up of the lattice
  - Definition of beam properties (particle type, energy, emittance ...)
  - Assignment of errors and imperfections
- Typical actions:
  - Compute lattice functions
  - Correct machines

Definitions of machine elements

- **All machine elements have to be described**
  - **Each machine element can be defined individually**
  - **As an instance of a previously defined CLASS**

    All elements from the same class have the same attributes

How to define machine elements ?

- **MAD-X Keywords used to define the type of an element.**
- **General format:**
  - **name: keyword, attributes;**
- **Can define single *element* or *class* of elements and give it a name**

→ **Some examples:**

Definitions of strengths

Dipole (bending) magnet:

$$k_0 \;=\; \frac{1}{p/c} B_y \,[\; in \quad m^{-1} \;] \left[\; = \frac{1}{\rho} \;=\; \frac{angle}{l} \right] [\; in \quad rad/m \;]$$

DIP01: SBEND, L=10.0, ANGLE=$angle$**, K0 =** k_0**;**

Quadrupole magnet:

$$k_1 \;=\; \frac{1}{p/c} \frac{\partial B_y}{\partial x} [\; in \quad m^{-2} \;] \left[\; = \frac{1}{l \cdot f} \right]$$

MQA: QUADRUPOLE, L=3.3, K1 = k_1**;**

Definitions of strengths

Sextupole magnet:

$$k_2 \;=\; \frac{1}{p/c} \frac{\partial^2 B_y}{\partial x^2} [\; in \quad m^{-3} \;]$$

KLSF = k_2**;**

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

Octupole magnet:

$$k_3 \;=\; \frac{1}{p/c} \frac{\partial^3 B_y}{\partial x^3} [\; in \quad m^{-4} \;]$$

KLOF = k_3**;**

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

Example: definitions of elements

Define a class of Quadrupole magnets:

```
MQF: QUADRUPOLE, L=3.3, K1 = +1.23E-02;
MQD: QUADRUPOLE, L=3.3, K1 = -1.23E-02;
```

QUAD01, QUAD02, ... are instances of the class MQF etc., all with the same properties:

```
QUAD01: MQF;
QUAD02: MQD;
QUAD03: MQF;
QUAD04: MQD;
 ....
```


Example: definitions of elements

LHC dipole magnet:

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

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

Try it ..

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


Thick and thin elements

- **Thick elements:** so far all examples were thick elements (or: lenses)
- Specify **length** and **strength** separately (except dipoles !)
  - \+ More precise, path lengths and fringe fields correct
  - − Not symplectic in tracking
  - − May need symplectic integration

Thick and thin elements

- **Thin elements:** specified as elements of **zero** length
- Specify **field integral**, e.g.: $k_0 \cdot L,\ k_1 \cdot L,\ k_2 \cdot L, ...$
  - + Easy to use
  - + Uses (amplitude dependent) kicks → always symplectic
  - + Used for tracking
  - − Path lengths not correctly described
  - − Fringe fields not correctly described
  - − Maybe problematic for small machines

Special MAD element: multipoles

Multipole: general element of zero length (thin lens), can be used with one or more components of any order:

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

→ knl = $k_n \cdot L$ (normal components of nth order)

Very simple to use:

mul1: multipole, knl := {0,k_1L,0,0,....};
is equivalent to definition of quadrupole ($k_1 L = \int \frac{1}{p/c} \frac{\partial B_y}{\partial x} \cdot dl$)

mul0: multipole, knl = {angle,0,0,....};
is equivalent to definition of a bending magnet

Thick and thin elements

- For all exercises: → use thin lenses (multipoles) unless explicitly requested to use thick elements
- Easier to handle and analytic calculation are precise

E.g. for a dipole you can use:
MYD: MULTIPOLE, KNL = {angle,0,0,....};

E.g. for a quadrupole you can use:
MYQ: MULTIPOLE, KNL := {0,k_1L,0,0,....};

Definitions of sequence (position)

Have to assign position to the elements.
Positions are defined in a sequence with a name.
A position can be defined at CENTRE or EXIT or ENTRY of an element .
Defined as absolute or relative position:

```
cassps: SEQUENCE, REFER=CENTRE, L=6912;
...
...
        here specify position of all elements ...
...
...
ENDSEQUENCE;
```

Definitions of sequence (position)

```
cassps: SEQUENCE, refer=centre, l=6912;
...
...
MBL01:  MBLA,   at = 102.7484; ! absolute position
MBL02:  MBLB,   at = 112.7484;
MQ01:  MQA,   at = 119.3984;
BPM01:  BPM,   at = 1.75, from MQ01; ! relative position
COR01:  MCV01,   at = LMCV/2 + LBPM/2, from BPM01;
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;
```


Complete example: SPS (thick)

```
circum = 6912;
// bending magnets as thin lenses
mbsps: multipole,knl={0.007272205};

// quadrupoles and sextupoles
kqf =  0.0146315;
kqd = -0.0146434;
qfsps: quadrupole,l=3.085,k1 := kqf;
qdsps: quadrupole,l=3.085,k1 := kqd;
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;

cassps: sequence, l = circum;
start_machine: marker, at = 0;
qfsps,  at = 1.5425;
```

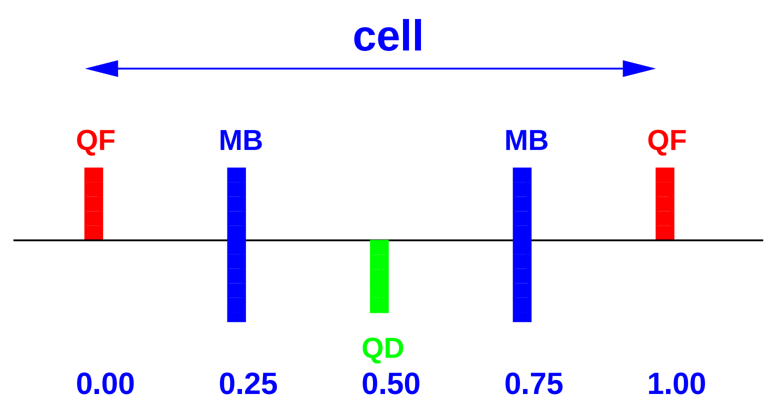
```
lsf, at = 3.6425;
ch,  at = 4.2425;
bpm, at = 4.3425;
mbsps, at = 5.0425;
mbsps, at = 11.4425;
mbsps, at = 23.6425;
mbsps, at = 30.0425;
qdsps, at = 33.5425;
lsd, at = 35.6425;
cv,  at = 36.2425;
bpm, at = 36.3425;
....
....
qdsps, at = 6881.5425;
lsd, at = 6883.6425;
cv,  at = 6884.2425;
bpm, at = 6884.3425;
mbsps, at = 6885.0425;
mbsps, at = 6891.4425;
mbsps, at = 6903.6425;
mbsps, at = 6910.0425;
end_machine: marker, at = 6912;
endsequence;
```

spsall.seq

Definition of large machines ...

- **For large machines with many elements:**
  - **Time consuming to specify every element individually (e.g. LHC more than 13000 elements needed)**
  - **Very inflexible (e.g. change of cell length)**
- **Several options:**
  - **Loops over elements possible**
  - **Elements can be combined into new objects**

A very simple cell ..



A very simple cell ..

- Positions can be defined in loops:
- Loop over number of cells (*ncell*)

```
lcell = 64;
ncell = 108;
circum = ncell*lcell;
cassps:  sequence, refer=centre, l=circum;
   n = 1;
   while (n < ncell+1)
      qfsps:  qfsps, at=(n-1)*lcell;
      mbsps:  mbsps, at=(n-1)*lcell + lcell*0.25;
      qdsps:  qdsps, at=(n-1)*lcell + lcell*0.50;
      mbsps:  mbsps, at=(n-1)*lcell + lcell*0.75;
      n = n + 1;

endsequence;
```

Inserting sequences

- Sequences can be defined and used like (new) elements:

```
cascell1:  sequence, refer=centre, l=lcell;   (cascell1 is now a CLASS)
   qfsps:  qfsps, at=0.0;
   mbsps:  mbsps, at=0.25*lcell;
   qdsps:  qdsps, at=0.50*lcell;
   mbsps:  mbsps, at=0.75*lcell;
endsequence;


allcells:  sequence, refer=centre, l=ncell*lcell;
   n = 1;
   while (n < ncell+1) {
      cascell1, at=(n-1)*lcell;
      n = n + 1;
   }
endsequence;
```


Simple MAD directives

- Define the input
- Define the beam
- Initiate computations (Twiss calculation, error assignment, orbit correction etc.)
- Output results (tables, plotting)
- Match desired parameters
- Beware: may have default values !

Input definition and selection

- Define the input:
  - call,"sps.seq";
  - → Selects a file with description of machine
  - → Can be split into several files
- Activate the machine:
  - USE, sequence=cassps;
  - → Activates the sequence you want (described in "sps.seq", which can contain more than one)

We still need a beam !

Some computations need to know the type of beam and its properties:

- Particle type
- Energy
- Emittance, number of particles, intensity ....

BEAM, PARTICLE=name, MASS=mass, NPART=Nb,
CHARGE=q, ENERGY=E,.......;

Example:

BEAM, PARTICLE=proton, NPART=1.1E11, ENERGY=450,.......;

Initiate the computations

Execute an action (calculation of all lattice parameters around the (circular !) machine):

twiss;   or:
twiss, file=output;   or:
twiss, file=output, sequence=cassps;

Execute an action (produce graphical output of β-functions):

plot, haxis=s, vaxis=betx, bety;

Initiate the computations

Set parameters for an action with the SELECT command (or defaults are used)

Calculation of Twiss parameters around the machine, store selected lattice functions on file and plot β-functions:

select,flag=twiss,column=name,s,betx,bety;
twiss, sequence=cassps, file=twiss.out;

plot, haxis=s, vaxis=betx, bety, colour=100;

Initiate the computations

Calculation of Twiss parameters around the machine, store and plot lattice functions for quadrupoles only (name starting with "q"):

```
select,flag=twiss,pattern="^q.*",column=name,s,betx,bety;
twiss, sequence=cassps, file=twiss.out;

plot, haxis=s, vaxis=betx, bety, colour=100;
```


Initiate the computations

Calculation of Twiss parameters around the machine, plot between 10th and 16th quadrupoles only:

```
select,flag=twiss,pattern="^q.*",column=name,s,betx,bety;
twiss, sequence=cassps, file=twiss.out;

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

Initiate the computations

Make a geometrical survey of the machine layout, available in a file:

```
select,flag=twiss,pattern="^q.*",column=name,s,betx,bety;
twiss, sequence=cassps, file=twiss.out;

plot, haxis=s, vaxis=betx, bety, colour=100,
range=qd[10]/qd[16];
survey, file=survey.cas;
```


Typical MAD example input:

```
// Read input file with machine description
call file="sps.seq";

// Define the beam for the machine
Beam, particle=proton, sequence=cassps, energy=450.0;

// Use the sequence with the name: cassps
use, sequence=cassps;

// Define the type and amount of output
select,flag=twiss,column=name,s,betx,bety;

// Execute the Twiss command to calculate the Twiss parameters
// Compute at the centre of the element and write to: twiss.out
twiss,save,centre,file=twiss.out;

// Plot the horizontal and vertical beta function between the
// 10th and 16th occurence of a defocussing quadrupole
plot, haxis=s, vaxis=betx, bety,colour=100, range=qd[10]/qd[16];

// get the geometrical layout (survey)
survey,file=survey.cas;

stop;
```

sps.madx

Typical MAD example input:

```
// Read input file with machine description
call file="sps.seq";

// Define the beam for the machine
Beam, particle=proton, sequence=cassps, energy = 450.0;

// Use the sequence with the name: cassps
use, sequence=cassps;

// Define the type and amount of output
select,flag=twiss,column=name,s,betx,bety;

// Execute the Twiss command to calculate the Twiss parameters
// Compute at the centre of the element and write to: twiss.out
twiss,save,centre,file=twiss.out;

// Plot the horizontal and vertical beta function between the
// 10th and 16th occurence of a defocussing quadrupole
plot, haxis=s, vaxis=betx, bety,colour=100, range=qd[10]/qd[16];

// get the geometrical layout (survey)
survey,file=survey.cas;

stop;
```


Typical MAD example input:

```
// Read input file with machine description
call file="sps.seq";

// Define the beam for the machine
Beam, particle=proton, sequence=cassps, energy=450.0;

// Use the sequence with the name:  cassps
use, sequence=cassps;

// Define the type and amount of output
select,flag=twiss,column=name,s,betx,bety;

// Execute the Twiss command to calculate the Twiss parameters
// Compute at the centre of the element and write to: twiss.out
twiss,save,centre,file=twiss.out;

// Plot the horizontal and vertical beta function between the
// 10th and 16th occurence of a defocussing quadrupole
plot, haxis=s, vaxis=betx, bety,colour=100, range=qd[10]/qd[16];

// get the geometrical layout (survey)
survey,file=survey.cas;

stop;
```

Typical MAD example input:

```
// Read input file with machine description
call file="sps.seq";

// Define the beam for the machine
Beam, particle=proton, sequence=cassps, energy=450.0;

// Use the sequence with the name: cassps
use, sequence=cassps;

// Define the type and amount of output
select,flag=twiss,column=name,s,betx,bety;

// Execute the Twiss command to calculate the Twiss parameters
// Compute at the centre of the element and write to: twiss.out
twiss,save,centre,file=twiss.out;

// Plot the horizontal and vertical beta function between the
// 10th and 16th occurence of a defocussing quadrupole
plot, haxis=s, vaxis=betx, bety,colour=100, range=qd[10]/qd[16];

// get the geometrical layout (survey)
survey,file=survey.cas;

stop;
```


Typical MAD example input:

```
// Read input file with machine description
call file="sps.seq";

// Define the beam for the machine
Beam, particle=proton, sequence=cassps, energy=450.0;

// Use the sequence with the name: cassps
use, sequence=cassps;

// Define the type and amount of output
select,flag=twiss,column=name,s,betx,bety;

// Execute the Twiss command to calculate the Twiss parameters
// Compute at the centre of the element and write to:  twiss.out
twiss,save,centre,file=twiss.out;

// Plot the horizontal and vertical beta function between the
// 10th and 16th occurence of a defocussing quadrupole
plot, haxis=s, vaxis=betx, bety,colour=100, range=qd[10]/qd[16];

// get the geometrical layout (survey)
survey,file=survey.cas;

stop;
```

Typical MAD example input:

```
// Read input file with machine description
call file="sps.seq";

// Define the beam for the machine
Beam, particle=proton, sequence=cassps, energy=450.0;

// Use the sequence with the name: cassps
use, sequence=cassps;

// Define the type and amount of output
select,flag=twiss,column=name,s,betx,bety;

// Execute the Twiss command to calculate the Twiss parameters
// Compute at the centre of the element and write to: twiss.out
twiss,save,centre,file=twiss.out;

// Plot the horizontal and vertical beta function between the
// 10th and 16th occurence of a defocussing quadrupole
plot, haxis=s, vaxis=betx, bety,colour=100, range=qd[10]/qd[16];


// get the geometrical layout (survey)
survey,file=survey.cas;


stop;
```


Typical MAD example input:

```
// Read input file with machine description
call file="sps.seq";

// Define the beam for the machine
Beam, particle=proton, sequence=cassps, energy=450.0;

// Use the sequence with the name: cassps
use, sequence=cassps;

// Define the type and amount of output
select,flag=twiss,column=name,s,betx,bety;

// Execute the Twiss command to calculate the Twiss parameters
// Compute at the centre of the element and write to: twiss.out
twiss,save,centre,file=twiss.out;
// Plot the horizontal and vertical beta function between the\\
// 10th and 16th occurence of a defocussing quadrupole\\
plot, haxis=s, vaxis=betx, bety,colour=100, range=qd[10]/qd[16];\\

// get the geometrical layout (survey)
survey,file=survey.cas;

stop;
```

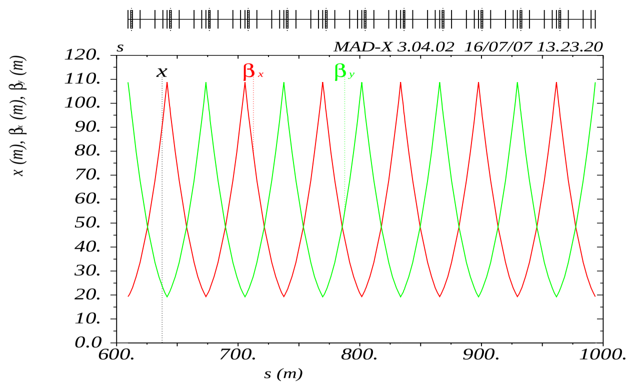
Typical MAD output (summary):

```
++++++ table: summm
          length           orbit5             alfa          gammatr
            6912               -0   0.001667526597       24.4885807

              q1              dq1          betxmax            dxmax
     26.57999204 -8.828683153e-09      108.7763569      2.575386926

           dxrms           xcomax           xcorms               q2
     1.926988371                0                0      26.62004577

             dq2          betymax            dymax            dyrms
  4.9186549e-08      108.7331749                0                0

          ycomax           ycorms           deltap          synch_1
               0                0                0                0
```

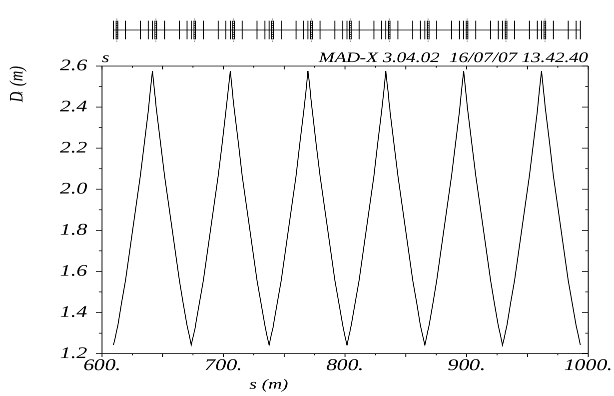

Typical MAD output (all elements):

```
* NAME                           S            BETX            BETY
$ %s                           %le             %le             %le
 "CASSPS$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
  .......
  .......
```

Graphical output (β)

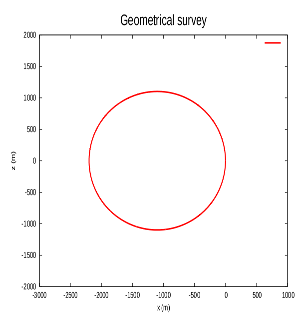


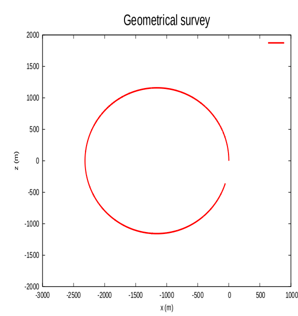
Graphical output (dispersion)



Graphical output (geometrical survey)

- Output gives x, y, z, θ in absolute (terrestrial) coordinates, plotting x versus z should be a ring:





Optical matching

- To get the optical configuration you want → compute settings yourself or use MAD for matching
- Main applications:
  - Setting global optical parameters (e.g. tune, chromaticity)
  - Setting local optical parameters (e.g. β-function, dispersion ..) → part 2
  - Correction of imperfections → part 2

Matching global parameters

- **Adjust strengths etc. to get desired properties (e.g. tune, chromaticity)**
- **Define the properties you want and the elements to vary**
- **Examples for global parameters (MAD convention):**
  - **Q1, Q2:(horizontal and vertical tune)**
  - **dQ1, dQ2:(horizontal and vertical chromaticity)**

Matching global parameters

!Example, match horizontal (Q1) and vertical (Q2) tunes:
!Vary the quadrupole strengths kqf and kqd
!Quadrupoles must be defined with: ..., k1:=kqf, ... etc.

```
match, sequence=cassps;
    global,sequence=cassps,Q1=26.58;      →   you want that !
    global,sequence=cassps,Q2=26.62;      →   you want that !
    vary,name=kqf, step=0.00001;      →   you vary that !
    vary,name=kqd, step=0.00001;      →   you vary that !
    Lmdif, calls=10, tolerance=1.0e-21;  → (Method to use !)
endmatch;
```

Changing MADX variables

- Deferred (:=) variables can be changed at any time during execution

```
use, period=cascell3;
ksf = 0.0;
ksd = 0.0;
select,flag=twiss,column=name,s,betx,muy,bety,dx,dy;
twiss,file=twiss1.out;

ksf = +0.017041/20.0;
ksd = −0.024714/20.0;
twiss,file=twiss2.out;
```

- Useful for: closed orbit, matching, chromaticity ...etc.

(Some comments ...)

- Input language seems heavy, but:
  - Can be interfaced to data base
  - Can be interfaced to other programs (e.g. Mathematica, Python,...)
  - Programs exist to generate the input interactively
  - Allows web based applications
  - Allows to develop complex tools

Twiss parameters for beam lines !

Do not close ! No periodic solution !

Must give INITIAL optical parameters !

```
twiss, betx=..., bety=..., alfx=..., ;

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


MADX - part 2

- We can:
  - Design and compute a regular lattice
  - Adjust basic machine parameters (tune, chromaticity, $\hat{\beta}$ ..)
- What next:
  - Machines with imperfections and corrections
  - Design of dispersion suppressor
  - Design of low β insertion

Error assignment

- MAD can assign errors to elements:
  - → Alignment errors on all or selected elements
  - → Field errors (up to high orders of multipole fields) on all or selected elements
- Errors are included in calculations (e.g. Twiss)
- Correction algorithms can be applied

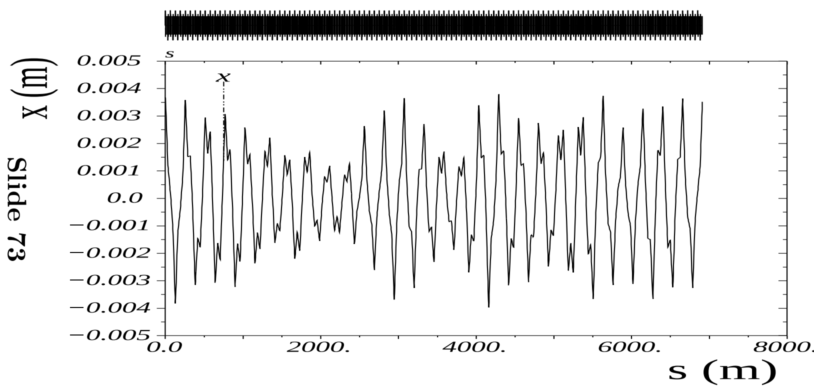
Error assignment

- → Can define alignment errors (EALIGN):

```
! assign error to all elements starting with Q
select,flag=error,pattern="Q.*";
Ealign, dx:=tgauss(3.0)*1.0e-4, dy:=tgauss(3.0)*2.0e-4;
Twiss,file=orbit.out;   ! compute distorted machine
plot,haxis=s,vaxis=x,y;   ! plot orbits in x and y
```

- → Can define field errors of any order (EFCOMP)
- → Remember the := !
- → See MADX Primer: page 14

Orbit with alignment errors



→ Now we want to correct the orbit

How to measure an orbit ?

Needs Beam Position Monitors (keyword → MONITOR):

Gives position in one or both dimensions [*in* *m*]

BPMV: VMONITOR, L=0.1;

BPMV01: VMONITOR, L=0.1;
BPMV02: VMONITOR, L=0.1;
BPMV03: BPMV;
BPMH02: HMONITOR, L=0.1;
BPMHV01: MONITOR, L=0.1;

For orbit correction: consider orbit only at monitors ...

How to correct an orbit ?

Needs Orbit corrector magnets (keyword → HKICKER/VKICKER):

The strength of a corrector is an angle (kick) [*in rad*]

MCV: VKICKER, L=0.1;

MCV01: VKICKER, L=0.1, KICK := KCV01;
MCV02: VKICKER, L=0.1, KICK := KCV02;
MCV03: MCV, KICK := KCV03;
MCH02: HKICKER, L=0.1, KICK := KCH01;

Q: why do I use := ?

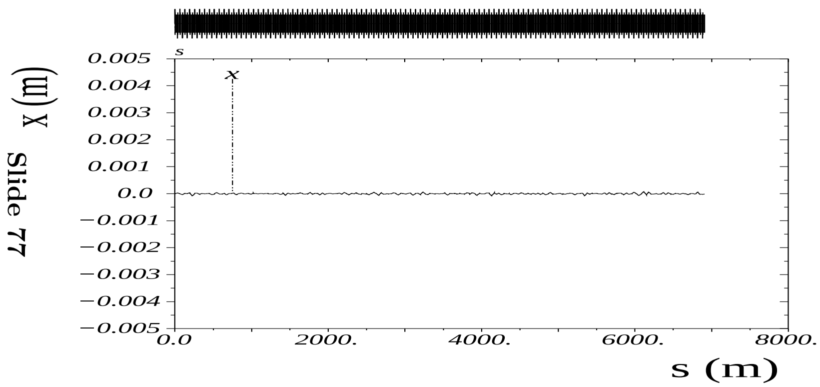
Orbit correction algorithms in MADX

- **Best kick method (MICADO) in horizontal plane:**
  ! Selected with MODE=MICADO

  Correct,mode=MICADO,plane=x,
  clist="c.tab",mlist="m.tab";
- **Singular Value Decomposition (SVD):**
  ! Selected with MODE=SVD

  Correct,mode=SVD,plane=x,
  clist="c.tab",mlist="m.tab";
- **For details: see MADX Primer**

Orbit after correction



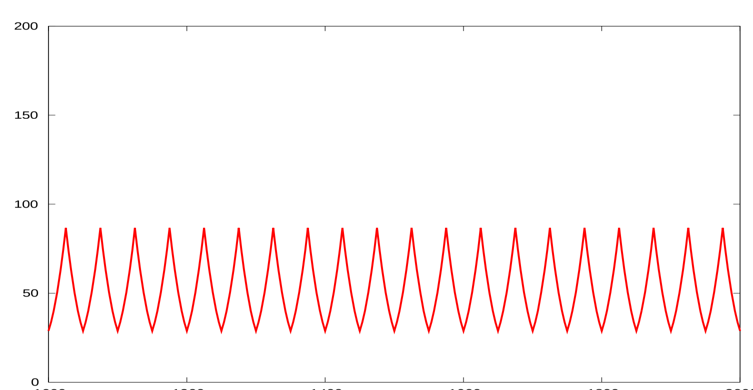
Optical matching

- To get the optical configuration you want → matching
- Main applications:
  - → Setting global optical parameters (e.g. tune, chromaticity)
  - → Setting local optical parameters (e.g. β-function, dispersion ..)
  - → Correction of imperfections

Matching local parameters

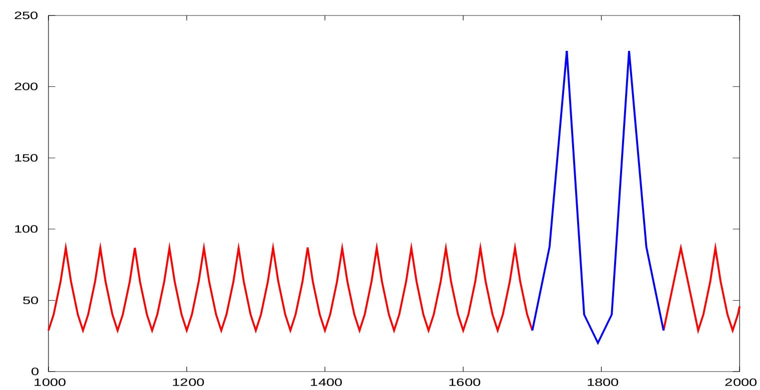
- Get local optical properties, but leave the rest of the machine unchanged
- Adjust strength of individual machine elements
- Examples for local matching:
  - Low (or high) β insertions
  - Dispersion suppressors

Local optical matching



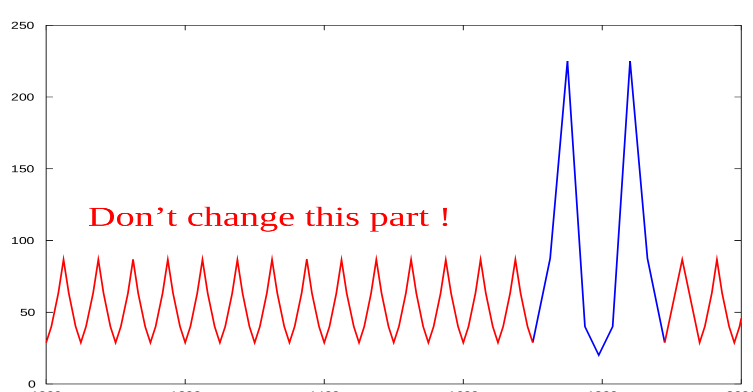
- What we have ...

Local optical matching



- What we want ...

Local optical matching



- What we want ...

Insertions (I)

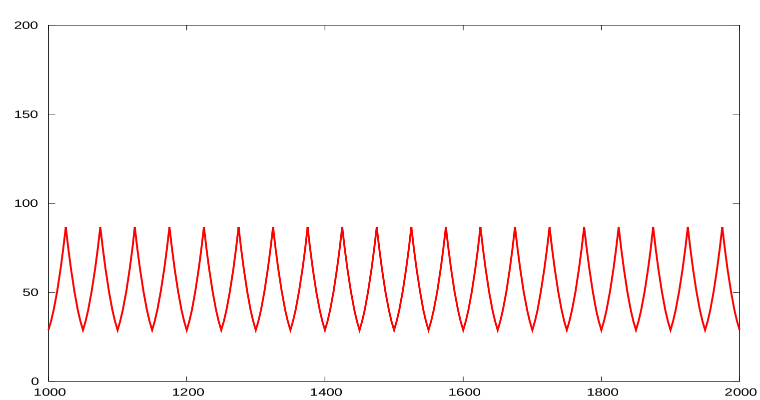
- How to add an insertion, e.g. two special cells ?

➡ **Start with periodic machine :**

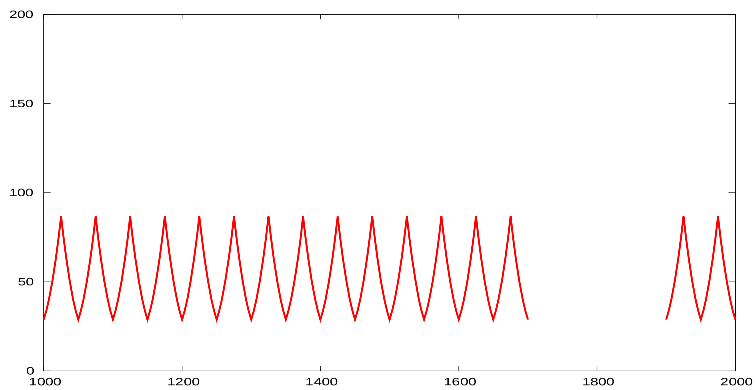
```
cassps: sequence, refer=centre, l=circum;
start_machine: marker, at = 0;
   n = 1;
   while (n <= ncell) {
      qfsps: qfsps,    at=(n-1)*lcell;
      mbsps: mbsps,    at=(n-1)*lcell + lcell*0.25;
      qdsps: qdsps,    at=(n-1)*lcell + lcell*0.50;
      mbsps: mbsps,    at=(n-1)*lcell + lcell*0.75;
   n = n + 1; }
end_machine: marker at=circum;
endsequence;
```

➡ **Split it into several pieces**

Local optical matching

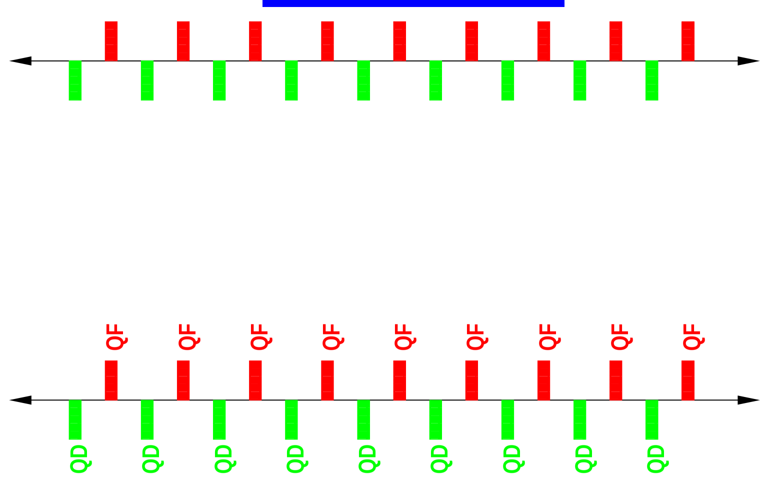


Local optical matching

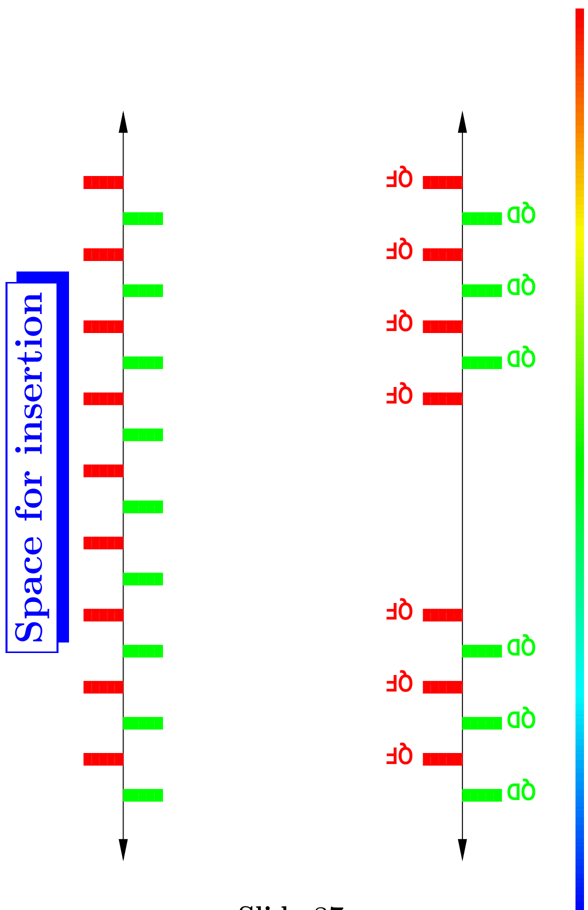


Slide 85

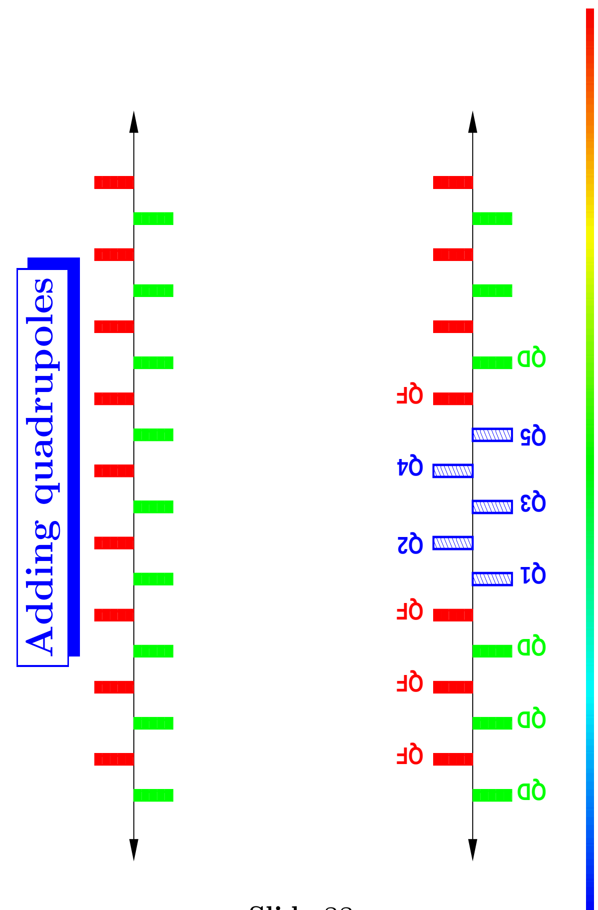
Original lattice



Slide 86



Slide 87



Slide 88

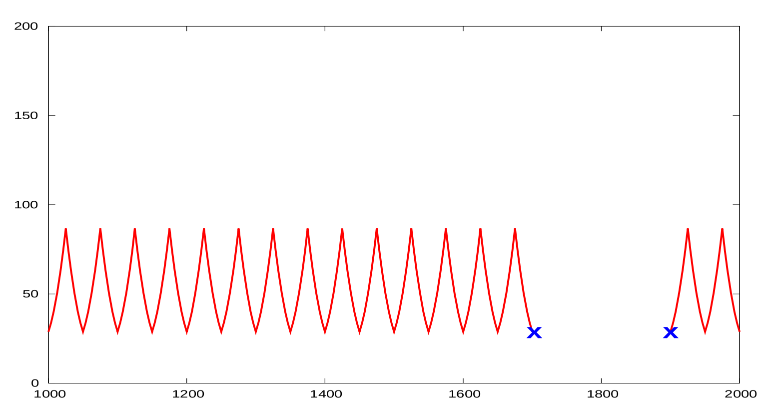
Insertions (II)

- Split it into several pieces

```
cassps: sequence, refer=centre, l=circum;
   n = 1;

while (n ≤ ncell-2) {

   qfsps: qfsps,   at=(n-1)*lcell;
   mbsps: mbsps,   at=(n-1)*lcell + lcell*0.25;
   qdsps: qdsps,   at=(n-1)*lcell + lcell*0.50;
   mbsps: mbsps,   at=(n-1)*lcell + lcell*0.75;
   n = n + 1;

}

   qf1  : qf1  ,   at=(ncell-2)*lcell;
   mbsps: mbsps,   at=(ncell-2)*lcell + lcell*0.25;
   qd1  : qd1  ,   at=(ncell-2)*lcell + lcell*0.50;
   mbsps: mbsps,   at=(ncell-2)*lcell + lcell*0.75;

   qf2  : qf2  ,   at=(ncell-1)*lcell;
   mbsps: mbsps,   at=(ncell-1)*lcell + lcell*0.25;
   qd2  : qd2  ,   at=(ncell-1)*lcell + lcell*0.50;
   mbsps: mbsps,   at=(ncell-1)*lcell + lcell*0.75;
endsequence;
```

s1_ins.seq

Local optical matching



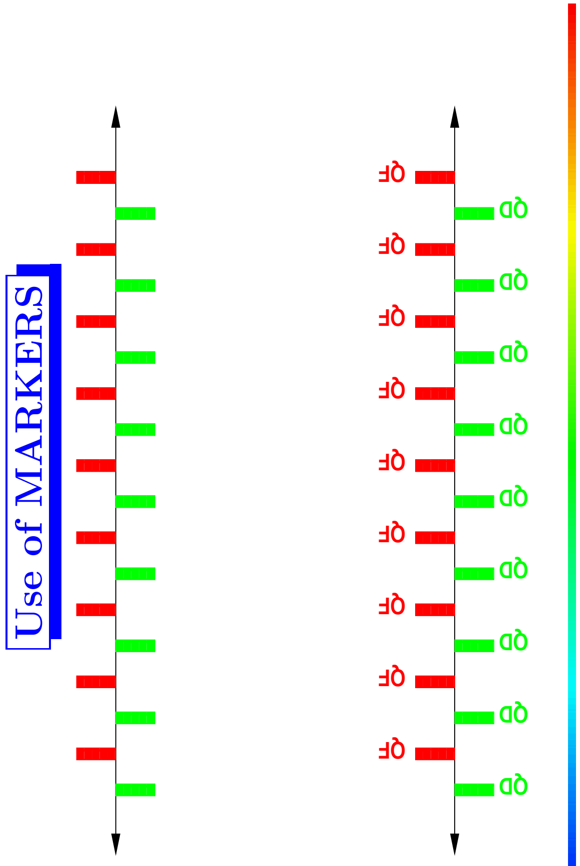
- Fix parameters at beginning and end of insertion

Matching techniques I(a)

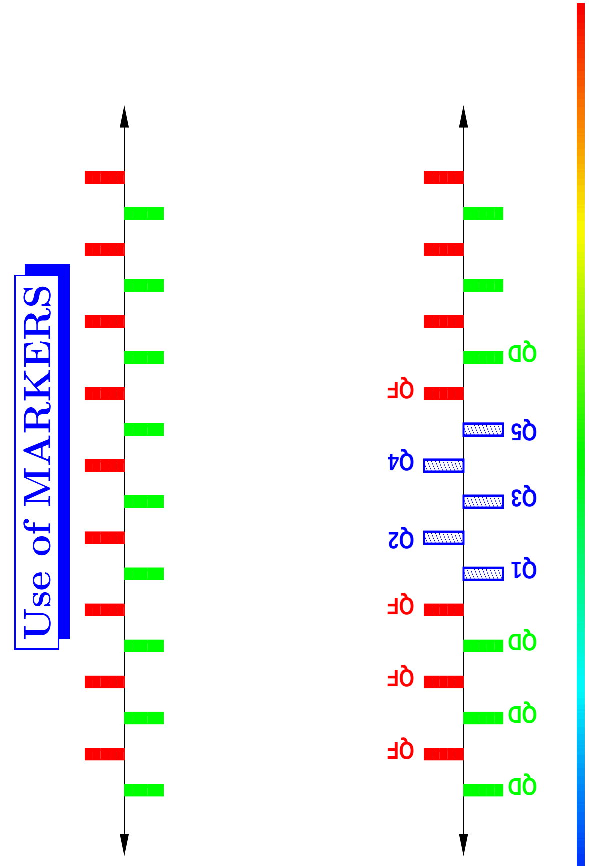
- **Use of markers:**
  - → **Have no effect on the optics**
  - → **Used to mark a position in the machine**
  - → **Can be used as reference in matching etc.**
- **Use:**
  - **left: MARKER, at=***position***;**
  - **right: MARKER, at=***position***;**

Matching techniques I(b)

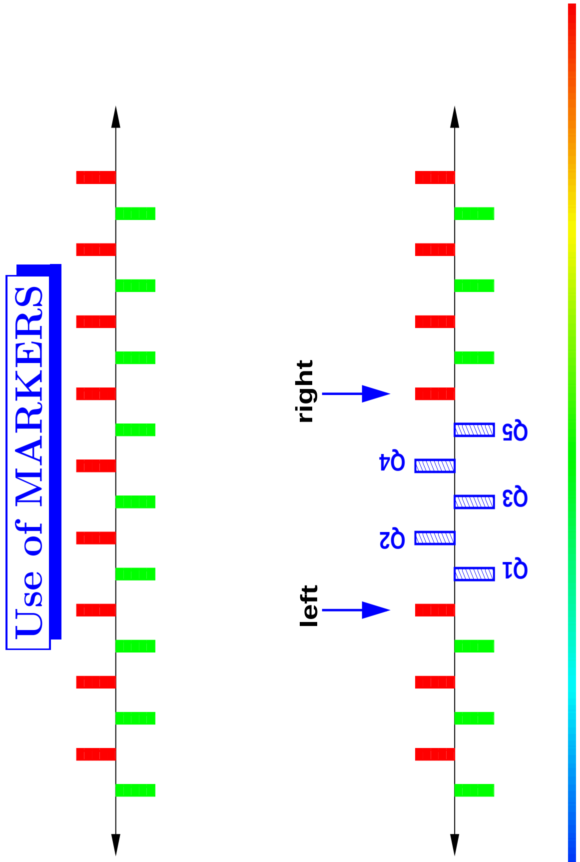
- **Markers:**
  - **can be used with RANGE in PLOT commands:**
    - → **PLOT, range=***left/right* **...;**
  - **can be used with RANGE in MATCH commands:**
    - → **MATCH, range=***left/right* **...;**
  - **can be used with PLACE in SAVEBETA commands to store twiss functions at position of the marker**
    - → **SAVEBETA, label=***left_beta***,place=***left***;**



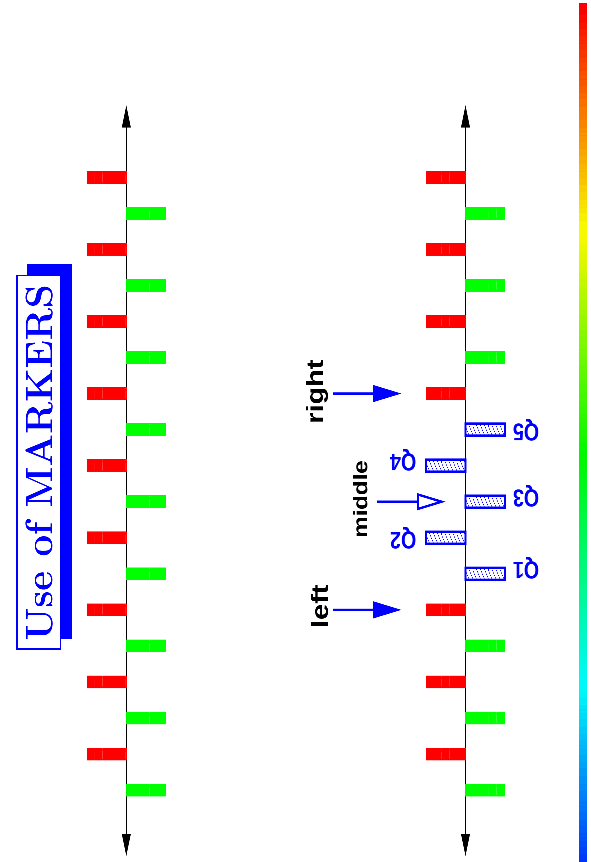
Slide 93



Slide 94



Slide 95



Slide 96

Matching techniques II

- **Matching is done only locally (between markers called left and right), not for the whole machine, needs initial and end conditions ($\beta_x, \alpha_x, ...$**

```
match, range=left/right,betx=..., alfx=..., bety=...;
vary,name=kq1.l, step=0.00001;
vary,name=kq2.l, step=0.00001;
vary,name=kq3.l, step=0.00001;
vary,name=kq4.l, step=0.00001;
vary,name=kq5.l, step=0.00001;
constraint,range=middle,sequence=cascell,betx=20.0,bety=50.0;
constraint,range=right,betx=..., alfx=..., bety=..., ...;
Lmdif, calls=100, tolerance=1.0e-21;
endmatch;
```


Using SAVEBETA to store optical functions

```
savebeta,label=tw_left,place=left;
savebeta,label=tw_right,place=right;
twiss;
match, sequence=cascell,range=left/right,beta0=tw_left;
vary,name=kq1.l, step=0.00001;
vary,name=kq2.l, step=0.00001;
vary,name=kq3.l, step=0.00001;
vary,name=kq4.l, step=0.00001;
vary,name=kq5.l, step=0.00001;
constraint,range=middle,sequence=cascell,betx=20.0,bety=50.0;
constraint,range=right,sequence=cascell,beta0=tw_right;
Lmdif, calls=100, tolerance=1.0e-21;
endmatch;
```

Matching techniques IV

- **Constraints on all quadrupoles, using limits:**

```
match, sequence=cascell;
vary,name=kqf, step=0.00001;
vary,name=kqd, step=0.00001;
constraint,pattern="^qf.*",sequence=cascell,betx < 100.0;
constraint,pattern="^qd.*",sequence=cascell,bety < 100.0;
Lmdif, calls=100, tolerance=1.0e-21;
endmatch;
```


Particle tracking

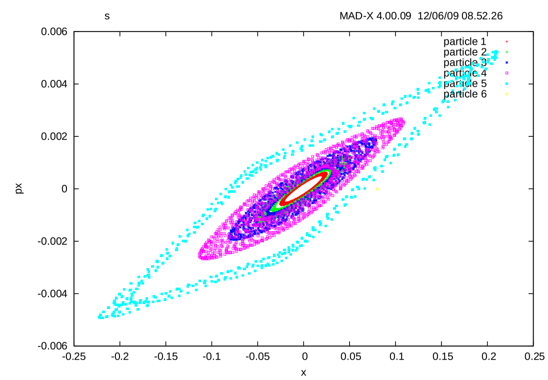
- **To track 4 particles for 1024 turns, add:**

```
track,file=track.out,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;
run,turns=1024;
endtrack;
plot, file="MAD_track",table=track,haxis=x,vaxis=px,
      particle=1,2,3,4, colour=1000, multiple, symbol=3;
plot, file="MAD_track",table=track,haxis=y,vaxis=py,
      particle=1,2,3,4, colour=1000, multiple, symbol=3;
```

tr1.madx

Particle tracking

- Phase space plot in horizontal coordinates:

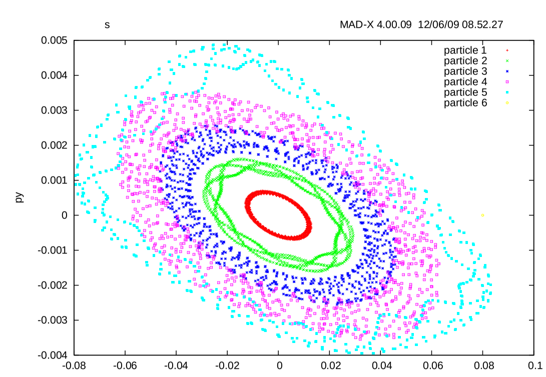


tr2.madx

Slide 101

Particle tracking

- Phase space plot in vertical coordinates:



tr2.madx

Slide 102

What we do not need (here !) ...

- Higher order effects
- IBS, beam-beam elements
- Equilibrium emittance (leptons)
- RF and acceleration