Computational Methods in Accelerator Physics

An introduction

Werner Herr
CERN, AB Department

(http://cern.ch/Werner.Herr/talks/zakopane_comp1.pdf)

Why do we need computations and simulations ?

- To explore new fields
- To answer scientific or technical questions
- To make design choices

People did that in the past without computers, but:

Why do we need computations and simulations ?

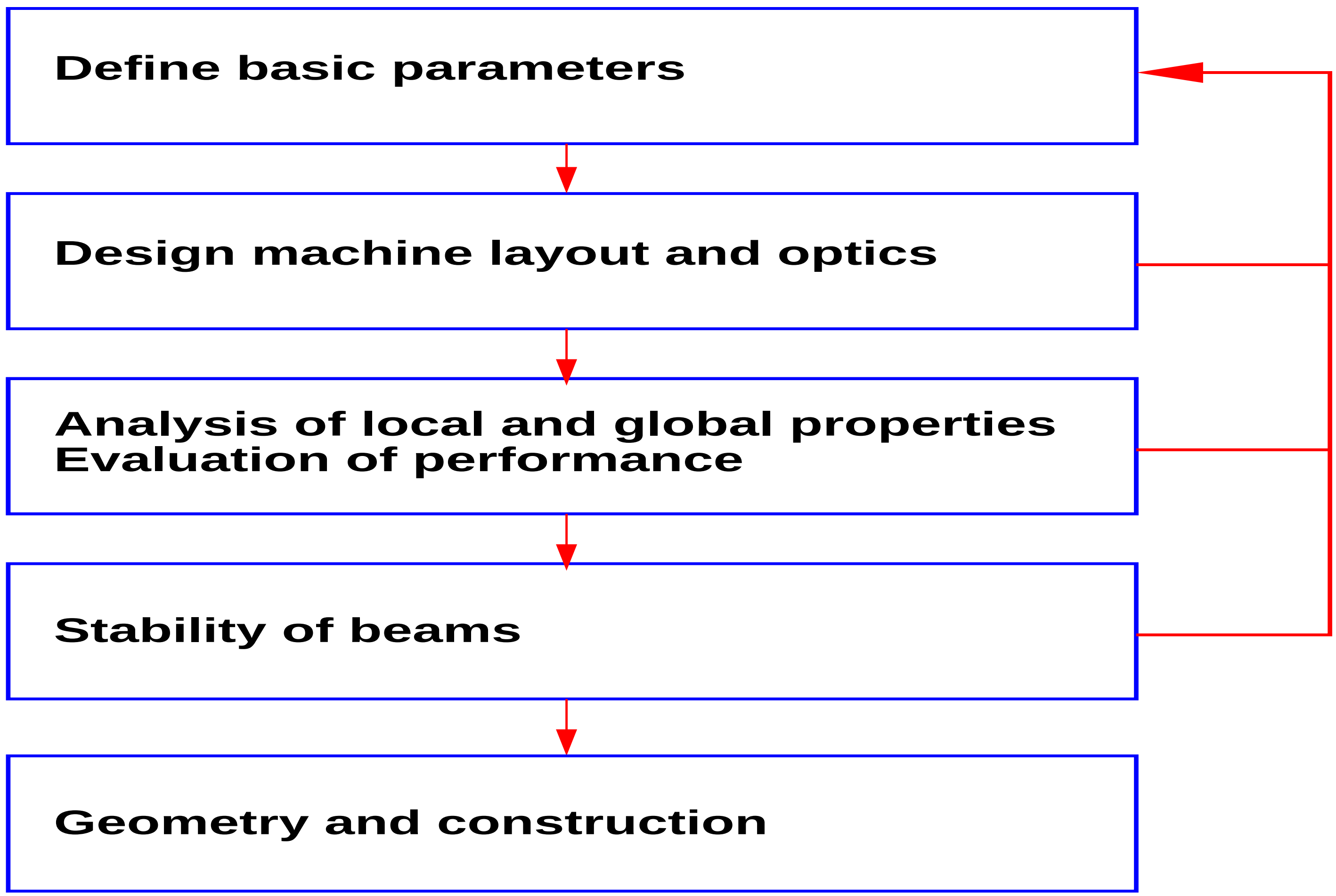
- **Early days of accelerators:**
  - Mainly for nuclear and particle physics
  - Design and operation by trial and error
- **Nowadays:**
  - Larger equipment, more people, more money
  - Many more applications
  - Safety issues
  - Complex control and operation

Where are computational methods needed ?

- **Studies of Beam Dynamics**
  - **Design and simulation of an accelerator**
  - **Control and operation**

- **Design of accelerator equipment**
  - **Magnets, RF cavities ...**
  - **Vacuum components, cryogenics**

Steps of accelerator design



Accelerator physics program needed

- Initial parameter calculation
- Optics design program
- Single particle dynamics modelling
- Multi particle dynamics modelling
- Geometry

→ Probably several programs needed

→ Have to understand what they are doing (algorithms and technicalities)

Accelerator design with an optics program

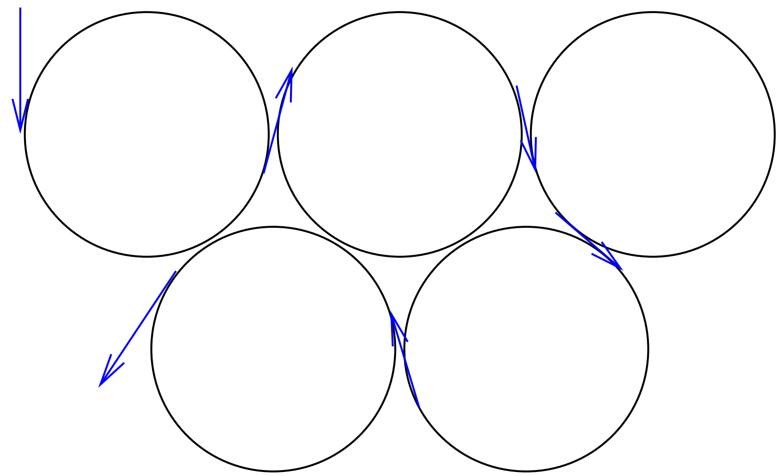
- **It needs: Description of machine in standard format**
- **It does: Optics calculations**
  - **Linear and non-linear optics computations**
  - **Linear corrections**
  - **Non-linear and chromatic corrections**
- **Parameter matching**

How does an accelerator look like to a computer ?

Not like:

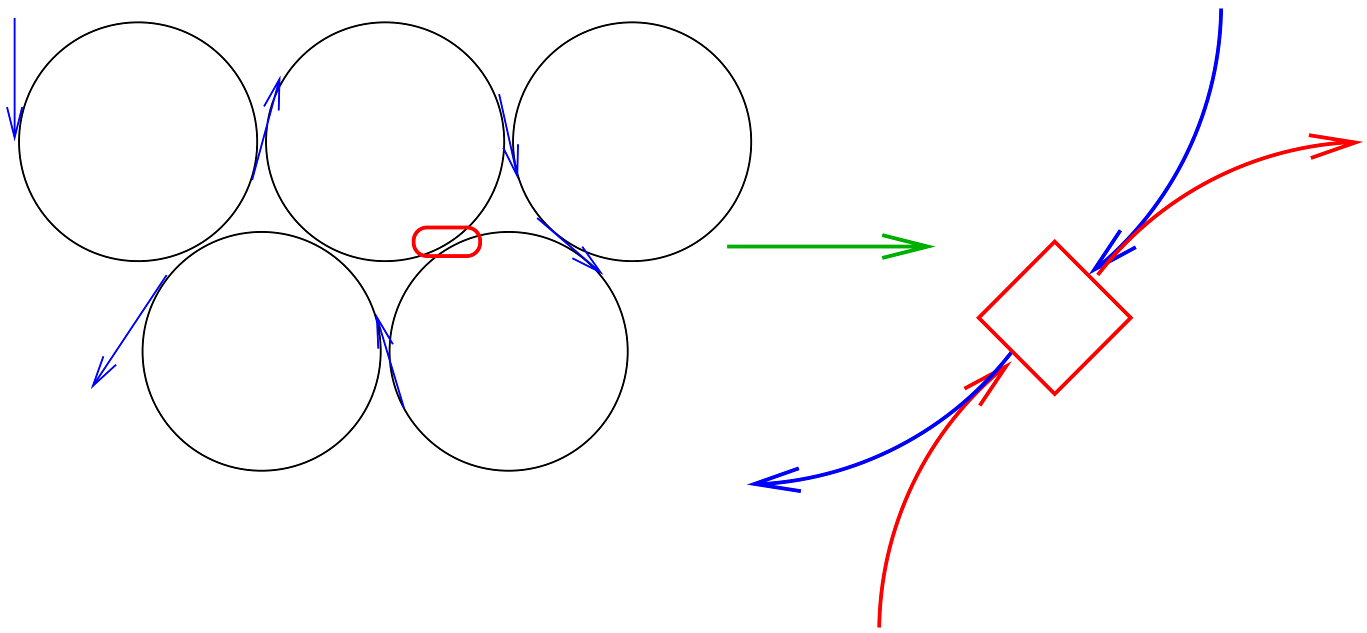
$$\frac{d^2x}{ds^2} + K(s)\,x = 0$$

- **The challenge:**
  - **Describe a machine with several thousand elements**
  - **Describe a complicated structure**



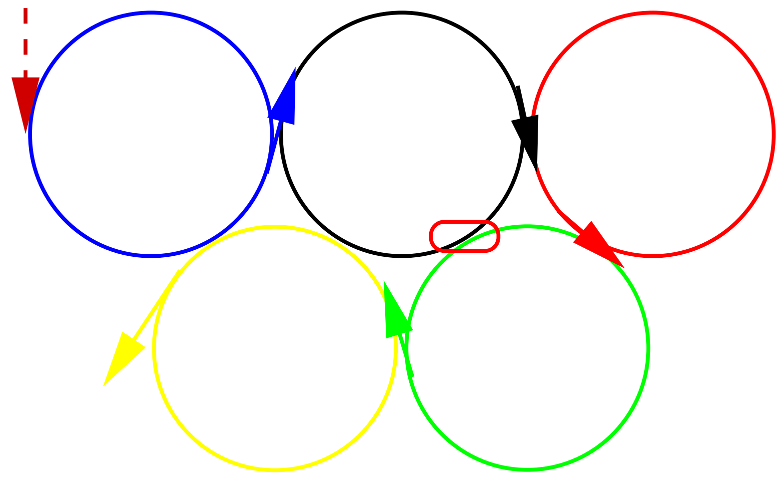
How does an accelerator look like to a computer ?

- **Additional complications:**
  - **Common elements**



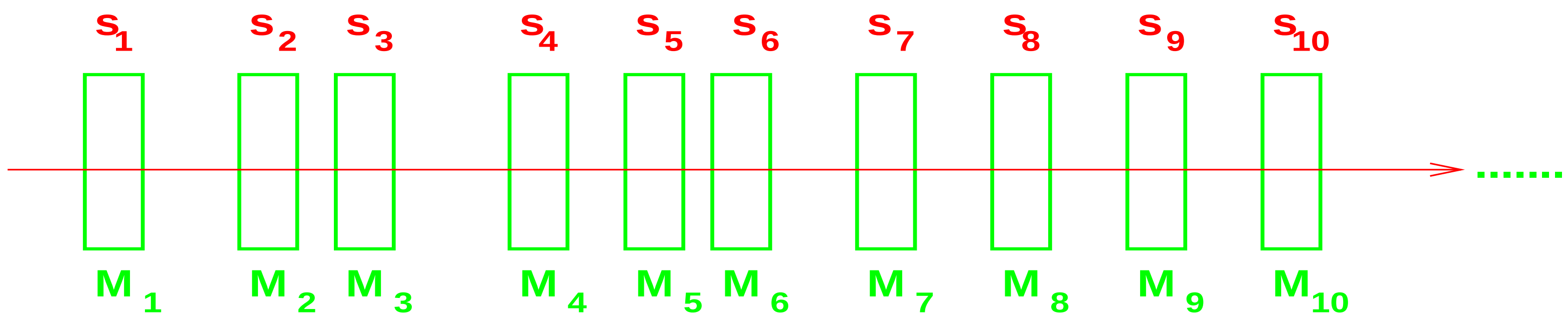
How does an accelerator look like to a computer ?

- **Additional complications:**
  - **Common elements**
  - **Changing energy**



How does an accelerator look like to a computer ?

- Bending and focusing is (in general) not a continuous function of s
- A computer sees it like a particle sees it !
- A (finite) sequence of machine elements $\mathcal{M}$ at longitudinal positions $s_1, s_2, s_3, \ldots$:



How does an element look like to a computer ?

- Each element $\mathcal{M}$ acts on the beam locally in a deterministic way

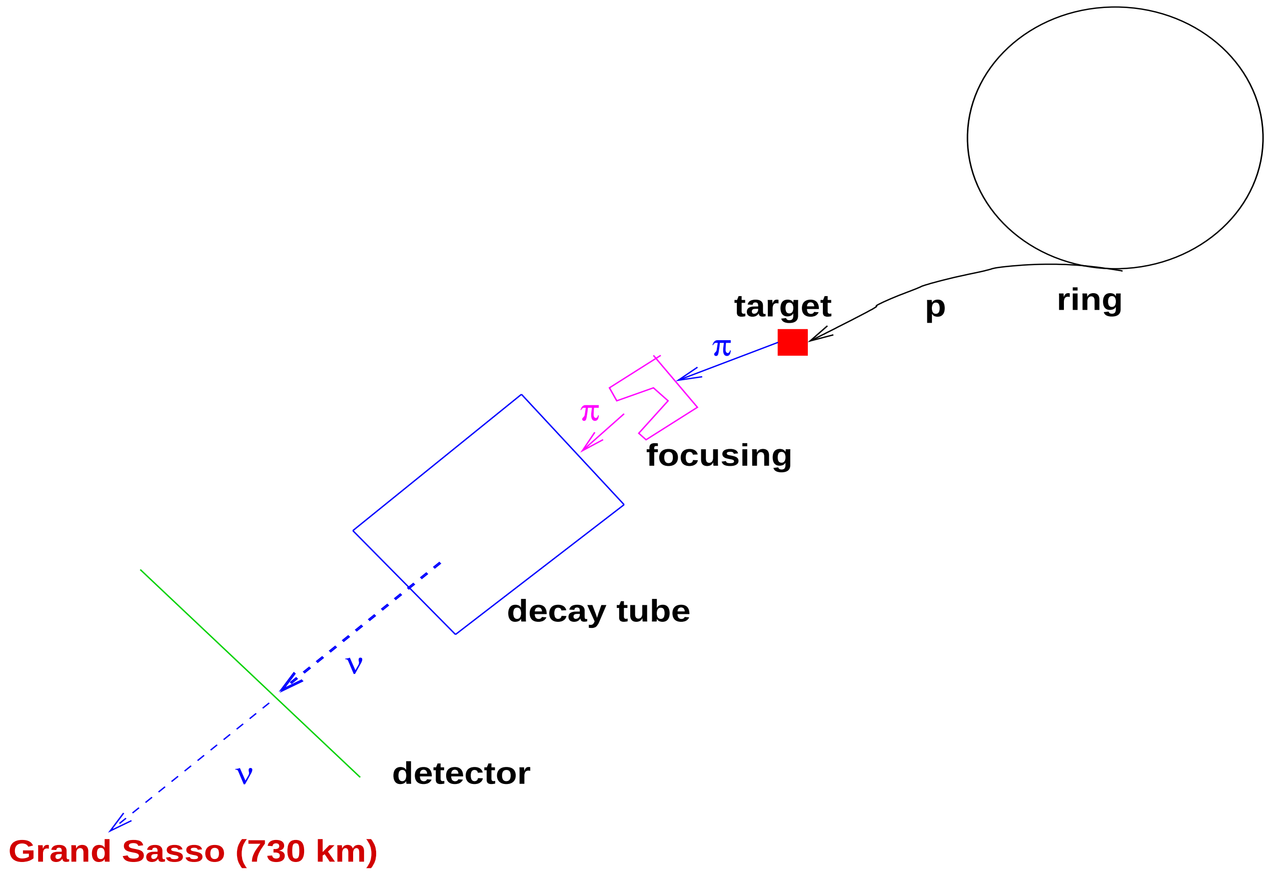
- In general: $\vec{z_2} \neq \vec{z_1}$

→ This sounds rather abstract, but:

What is $\mathcal{M}$? It can represent:

- Single machine elements:
  - magnet: dipole, quadrupole
  - RF cavity
- Single machine elements (not only magnets):
  - collimators, targets, obstacles
  - vacuum chamber
  - drift
  - ......
- Important to integrate in the design !

Example (CNGS)



Example (collimation systems)

- **needed to control beam size and for protection**
- **may constrain the optics and layout**
- **when particles hit a collimator:**
  - → **what happens to them ?**
  - → **where do they go ?**
  - → **can they damage the machine ?**

The map $\mathcal{M}$ can also represent:

- **Several machine elements combined, (i.e. part of a ring made of m elements: $\mathcal{M}_{part}$)**

$$\mathcal{M}_{part} \;=\; ......\circ \mathcal{M}_l \circ ......\circ \mathcal{M}_{l+m}$$

- **The whole machine (e.g. complete turn in a ring with N elements, $\mathcal{M}_{ring}$)**

$$\mathcal{M}_{ring} \;=\; \mathcal{M}_1 \circ \mathcal{M}_2 \circ ......\circ \mathcal{M}_N$$

- **Many turns in a ring (M $=$ $\mathcal{M}_{ring}{}^n$)**

How is an element described to a computer ?

- Let $\vec{z}_1, \vec{z}_2$ describe a quantity (coordinates, beam sizes ...) before and after the element
- Take an machine element (e.g. magnet) and build a mathematical object $\mathcal{M}$ for this quantity
  - ➡ $\mathcal{M}$ describes the element in terms of this quantity
  - ➡ In general: $\vec{z}_2 \; = \; \mathcal{M} \circ \vec{z}_1$
  - ➡ $\mathcal{M}$ is a so-called **MAP**
- The complete sequence of MAPS connects the pieces together to make a ring (or beam line)

MAPS transform coordinates through an element

- **4 coordinates needed for 2 dimensions (off-momentum effects ignored)**
- **Coordinate vector:** $\vec{z} \;=\; (x, \quad x' = \frac{\delta x}{\delta s}, \quad y, \quad y' = \frac{\delta y}{\delta s})$
- $\mathcal{M}$ **transforms the coordinates** $\vec{z_1}(s_1)$ **at position** s_1 **to new coordinates** $\vec{z_2}(s_2)$ **at position** s_2**:**

$$\vec{z_2}(s_2) = \begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_{s_2} = \mathcal{M} \circ \begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_{s_1} = \mathcal{M} \circ \vec{z_1}(s_1)$$

... or OPTICAL functions

- **6 optical functions used for 2 dimensions:**

$$\vec{\nu_2}(s_2) = \begin{pmatrix} \beta_x \\ \alpha_x \\ \gamma_x \\ \beta_y \\ \alpha_y \\ \gamma_y \end{pmatrix}_{s_2} = \mathcal{M} \circ \begin{pmatrix} \beta_x \\ \alpha_x \\ \gamma_x \\ \beta_y \\ \alpha_y \\ \gamma_y \end{pmatrix}_{s_1} = \mathcal{M} \circ \vec{\nu_1}(s_1)$$

How does $\mathcal{M}$ look like ?

The map $\mathcal{M}$ describes local properties of a machine element and can be:

- **A simple linear matrix or transformation**
- **High order integration algorithm**
- **Derived from local Hamiltonian**
- **A computer program, subroutine etc.**
- **Any "description" to go from $\vec{z}_1$ to $\vec{z}_2$**
- **Can in principle be VERY abstract !**

Simple examples (linear, one dimensional)
(Matrix formulation for linear* elements)

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{s_2} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \circ \begin{pmatrix} x \\ x' \end{pmatrix}_{s_1}$$

$$\begin{pmatrix} \beta \\ \alpha \\ \gamma \end{pmatrix}_{s_2} = \begin{pmatrix} m_{11}^2 & -2m_{11}m_{12} & m_{12}^2 \\ -m_{11}m_{21} & m_{11}m_{22} + m_{12}m_{21} & -m_{12}m_{22} \\ m_{21}^2 & -2m_{21}m_{22} & m_{22}^2 \end{pmatrix} \circ \begin{pmatrix} \beta \\ \alpha \\ \gamma \end{pmatrix}_{s_1}$$

→ **The maps become so-called transport matrices**

* **The changes depend on x or x' only**

(Interlude I: Σ-matrix)

The transformation of the optical functions can also be written using the Σ-matrix formalism:

$$\Sigma_{s_2} = \mathcal{M} \circ \Sigma_{s_1} \circ \mathcal{M}^{\mathcal{T}}$$

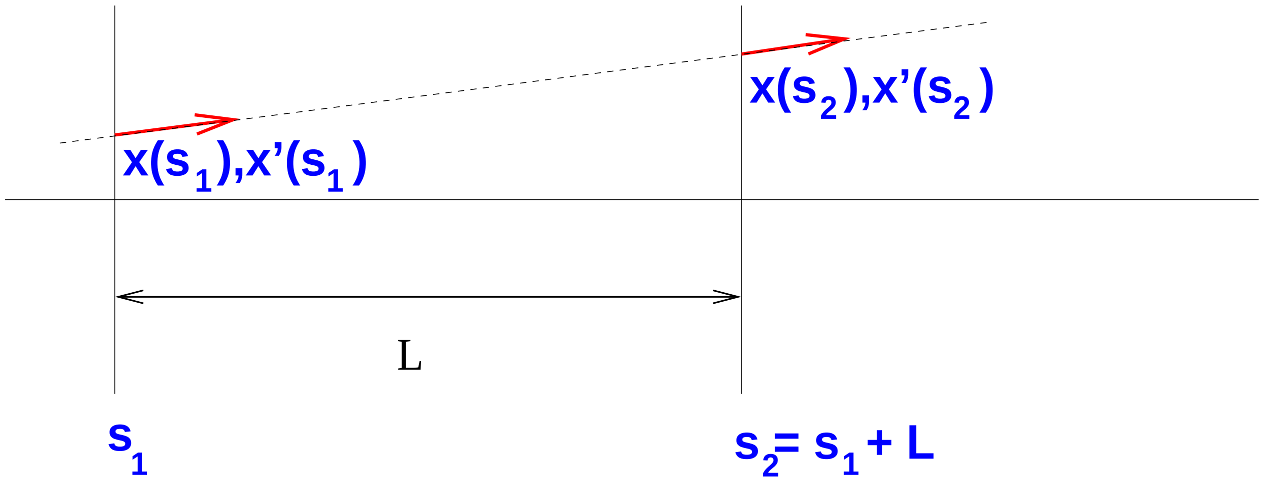
i.e. for example in the linear case:

$$\begin{pmatrix} \beta & -\alpha \\ -\alpha & \gamma \end{pmatrix}_{s_2} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \circ \begin{pmatrix} \beta & -\alpha \\ -\alpha & \gamma \end{pmatrix}_{s_1} \circ \begin{pmatrix} m_{11} & m_{21} \\ m_{12} & m_{22} \end{pmatrix}$$

- **Allows formal extension to higher order effects (e.g. synchrotron radiation)**
- **Prove that it is equivalent to previous formula**

Transformation of coordinates (one dimension)

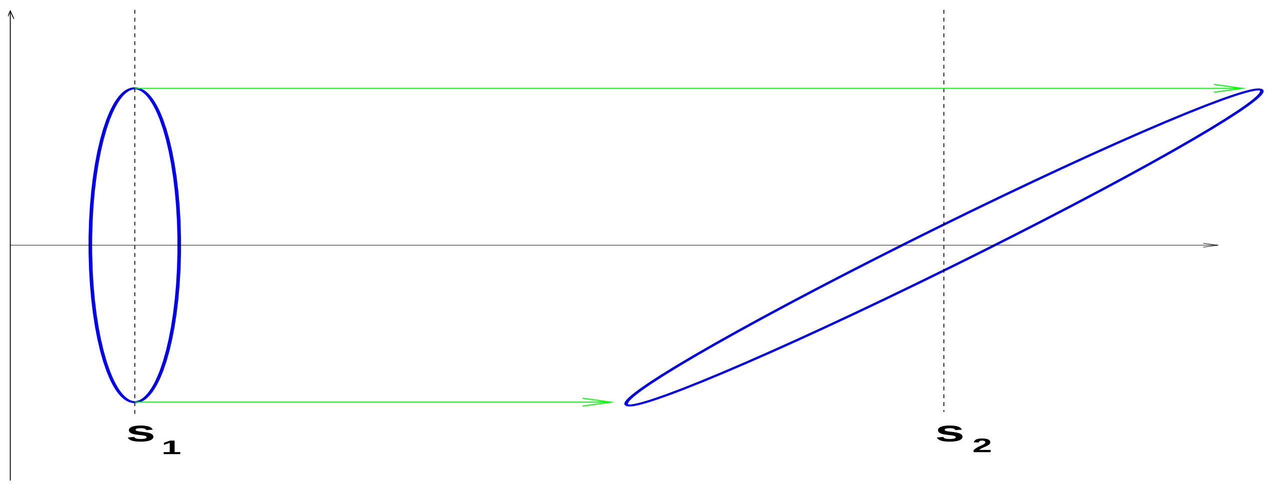
Drift space of length $L = s_2 - s_1$**:**



$$\begin{pmatrix} x \\ x' \end{pmatrix}_{s_2} = \begin{pmatrix} x \\ x' \end{pmatrix}_{s_1} + \begin{pmatrix} x' \cdot L \\ 0 \end{pmatrix}_{s_1} = \begin{pmatrix} 1 & L \\ 0 & 1 \end{pmatrix} \circ \begin{pmatrix} x \\ x' \end{pmatrix}_{s_1}$$

Transformation of beam ellipse

Drift space of length $L = s_2 - s_1$:



$$\begin{pmatrix} \beta \\ \alpha \\ \gamma \end{pmatrix}_{s_2} = \begin{pmatrix} 1 & -2L & L^2 \\ 0 & 1 & -L \\ 0 & 0 & 1 \end{pmatrix} \circ \begin{pmatrix} \beta \\ \alpha \\ \gamma \end{pmatrix}_{s_1} = \begin{pmatrix} \beta_0 - 2L\alpha_0 + L^2\gamma_0 \\ \alpha_0 - L\gamma_0 \\ \gamma_0 \end{pmatrix}_{s_2}$$

Simple examples (one dimensional)

Focusing quadrupole of length L and strength K:

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{s_2} = \begin{pmatrix} \cos(L \cdot K) & \frac{1}{K} \cdot \sin(L \cdot K) \\ K \cdot \sin(L \cdot K) & \cos(L \cdot K) \end{pmatrix} \circ \begin{pmatrix} x \\ x' \end{pmatrix}_{s_1}$$

Quadrupole with short length L (i.e.: $1 \gg L^2 \cdot K^2$)

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{s_2} = \begin{pmatrix} 1 & 0 \\ K^2 \cdot L(= -\frac{1}{f}) & 1 \end{pmatrix} \circ \begin{pmatrix} x \\ x' \end{pmatrix}_{s_1}$$

Initial steps for optics calculation

- **The optics program reads the sequence of elements of a machine (their order, their positions ..)**
- **It reads properties of the elements, i.e. type (dipole, quadrupole, drift ...)**
- **It sets strength of the elements**
- **It sets up the maps (matrices)**

→ **A ”standard” for the input language exists, plus converters (do not forget this issue !)**

Simplest machine description (MADX format)

```
// description of elements and their strengths
// dipoles and quadrupoles only ...
   mb: dipole,     l=6.0, angle=0.03570;
   qf: quadrupole, l=3.0, k1= 0.013426;
   qd: quadrupole, l=3.0, k1=-0.013426;

// centre position of elements in the ring
   start:    at=0;
   qf.1: qf, at=1.5000e+00;
   mb: mb,   at=9.0000e+00;
   mb: mb,   at=1.9000e+01;
   qd.1: qd, at=2.6500e+01;
   mb: mb,   at=3.4000e+01;
   mb: mb,   at=4.4000e+01;
   qf.2: qf, at=5.1500e+01;
             ...
   end:      at=2.2000e+03;
```

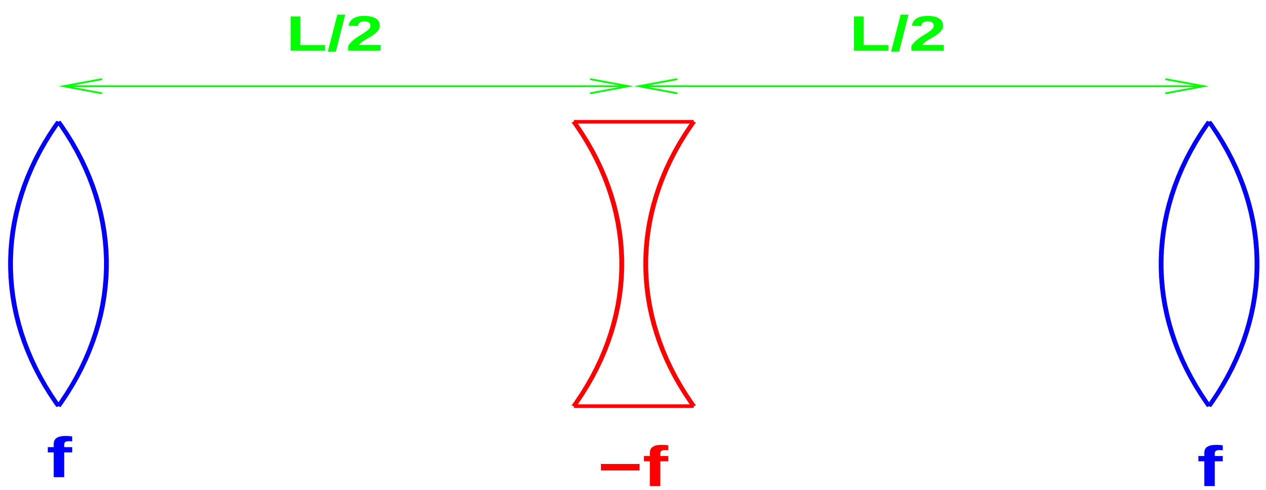
Putting the "pieces" together

Starting from a position s_0 and applying all maps (for N elements) in sequence around a ring with circumference C to get the One-Turn-Map (OTM) for the position s_0 (for one dimension only):

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{s_0 + C} = \mathcal{M}_1 \circ \mathcal{M}_2 \circ \ldots \circ \mathcal{M}_N \circ \begin{pmatrix} x \\ x' \end{pmatrix}_{s_0}$$

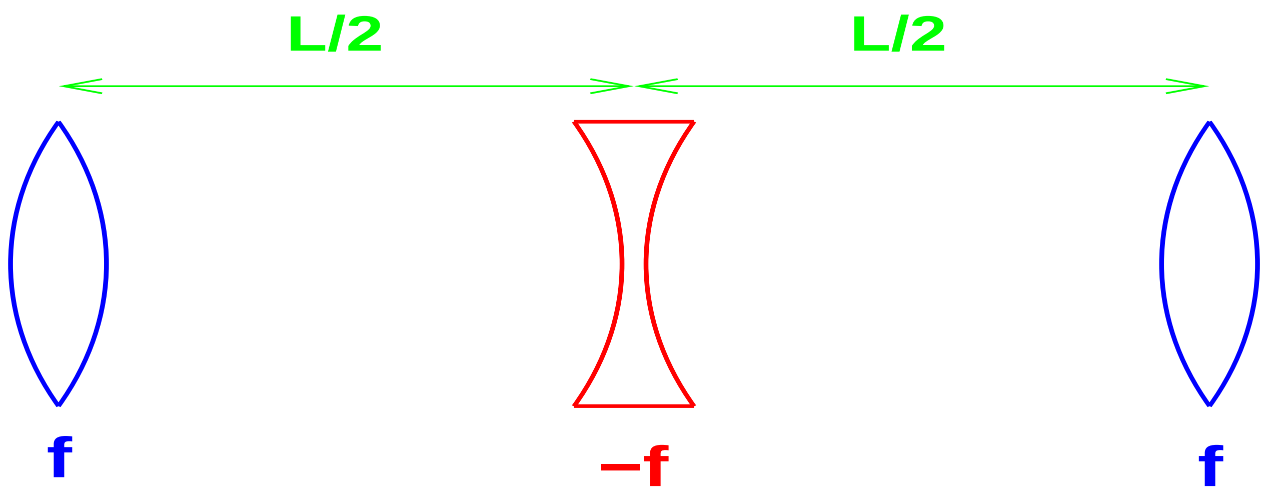
$$\Longrightarrow \begin{pmatrix} x \\ x' \end{pmatrix}_{s_0 + C} = \mathcal{M}_{ring}(s_0) \circ \begin{pmatrix} x \\ x' \end{pmatrix}_{s_0}$$

Composition of elements (FODO cell)
(here: simple matrix multiplications)



$$\mathcal{M}_{cell} = \begin{pmatrix} 1 & 0 \\ -\frac{1}{f} & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & L/2 \\ 0 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 0 \\ \frac{1}{f} & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & L/2 \\ 0 & 1 \end{pmatrix}$$

Composition of elements (FODO cell)
(here: simple matrix multiplications)



$$\mathcal{M}_{cell} = \begin{pmatrix} 1 + \frac{L}{2f} & L + \frac{L^2}{4f} \\ -\frac{L}{2f^2} & 1 - \frac{L}{2f} - \frac{L^2}{4f^2} \end{pmatrix}$$

What can we do with $\mathcal{M}_{ring}$?

- **The map $\mathcal{M}_{ring}$ is extremely important:**
  - **It describes the global behaviour**
  - **A computer does not know Hill's equation**
  - **Courant-Snyder ansatz (formalism) assumes motion is linearly stable, periodic, confined, and has a closed orbit.**
  - **The OTM $\mathcal{M}_{ring}$ contains all information about global behaviour in the ring, i.e. stability, tune, β, closed orbit etc.**
- **No need for assumptions**

What else can we do with $\mathcal{M}_{ring}$?

- $\mathcal{M}_{ring}$ or $\mathcal{M}_{part}$ allow the analysis of imperfections (and their correction !)
- "Straightforward" to formally extend it to complicated (e.g. non-linear) problems - additional tools and concepts needed (invariants, fixpoints, normal forms etc.)

(Interlude II: Fixed Points)

- **Certain points in phase space $\vec{z}_1$ repeat itself after n completed turns**

$$\mathcal{M}^{n}_{ring} \circ \vec{z}_1 = \vec{z}_2 \equiv \vec{z}_1$$

- **Defines a Fixed Point of order n**
- **Fixed Point of order 1 is the closed orbit**
- **Stability requires existence of such a fixed point**
- **Closed orbit is found (or not !) in optics programs by searching for the first order fixed point**

Analysis of the results

- If all maps are matrices (i.e. only linear elements)
- Usually the case for initial design

→ The One-Turn-Map is a MATRIX:

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{s_0 + C} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \circ \begin{pmatrix} x \\ x' \end{pmatrix}_{s_0}$$

→ After all multiplications we get the One-Turn-Matrix which depends on the starting point s_0.

Find the tune Q

We can find the tune Q from the One-Turn-Matrix $\mathcal{M}_{ring}(s_0)$ by computing the eigenvalues of $\mathcal{M}_{ring}(s_0)$:

$$\det(\mathcal{M}_{ring}(s_0) - \lambda) = 0$$

gives

$$\lambda = \cos(2\pi Q) \pm i \cdot \sin(2\pi Q)$$

(verify with the One-Turn-Matrix you know from previous lecture !)

The analysis of the results

What else can we do with the One-Turn-Matrix ?

We can express the One-Turn-Matrix $\mathcal{M}_{ring}(s_0)$ in terms of Courant-Snyder parameters:

We know that $\mathcal{M}_{ring}(s_0)$ for one dimension must be:

$$\mathcal{M}_{ring}(s_0) \equiv \begin{pmatrix} \cos\mu + \alpha(s_0)\sin\mu & \beta(s_0)\sin\mu \\ -\gamma(s_0)\sin\mu & \cos\mu - \alpha(s_0)\sin\mu \end{pmatrix}$$

and we also know that (for a ring):

$$\alpha(s_0+C) \equiv \alpha(s_0), \quad \beta(s_0+C) \equiv \beta(s_0), \quad \gamma(s_0+C) \equiv \gamma(s_0)$$

The analysis of the results

Comparison of:

$$\begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} = \mathcal{M}_1 \quad \circ \quad \mathcal{M}_2 \quad \circ \quad ... \quad \circ \quad \mathcal{M}_N$$

and :

$$\mathcal{M}_{ring}(s_0) = \begin{pmatrix} \cos\mu + \alpha(s_0)\sin\mu & \beta(s_0)\sin\mu \\ -\gamma(s_0)\sin\mu & \cos\mu - \alpha(s_0)\sin\mu \end{pmatrix}$$

gives optical functions at position s_0:

➡ $\beta(s_0)$, $\alpha(s_0)$, $\gamma(s_0)$ **(depend on position s_0)**

➡ μ **is independent of s_0: (2πQ)**

We have now:

- **Values for β_x, β_y, α_x, ... etc. at the position s_0**
- **Tunes for both planes, closed orbit**

The next step:

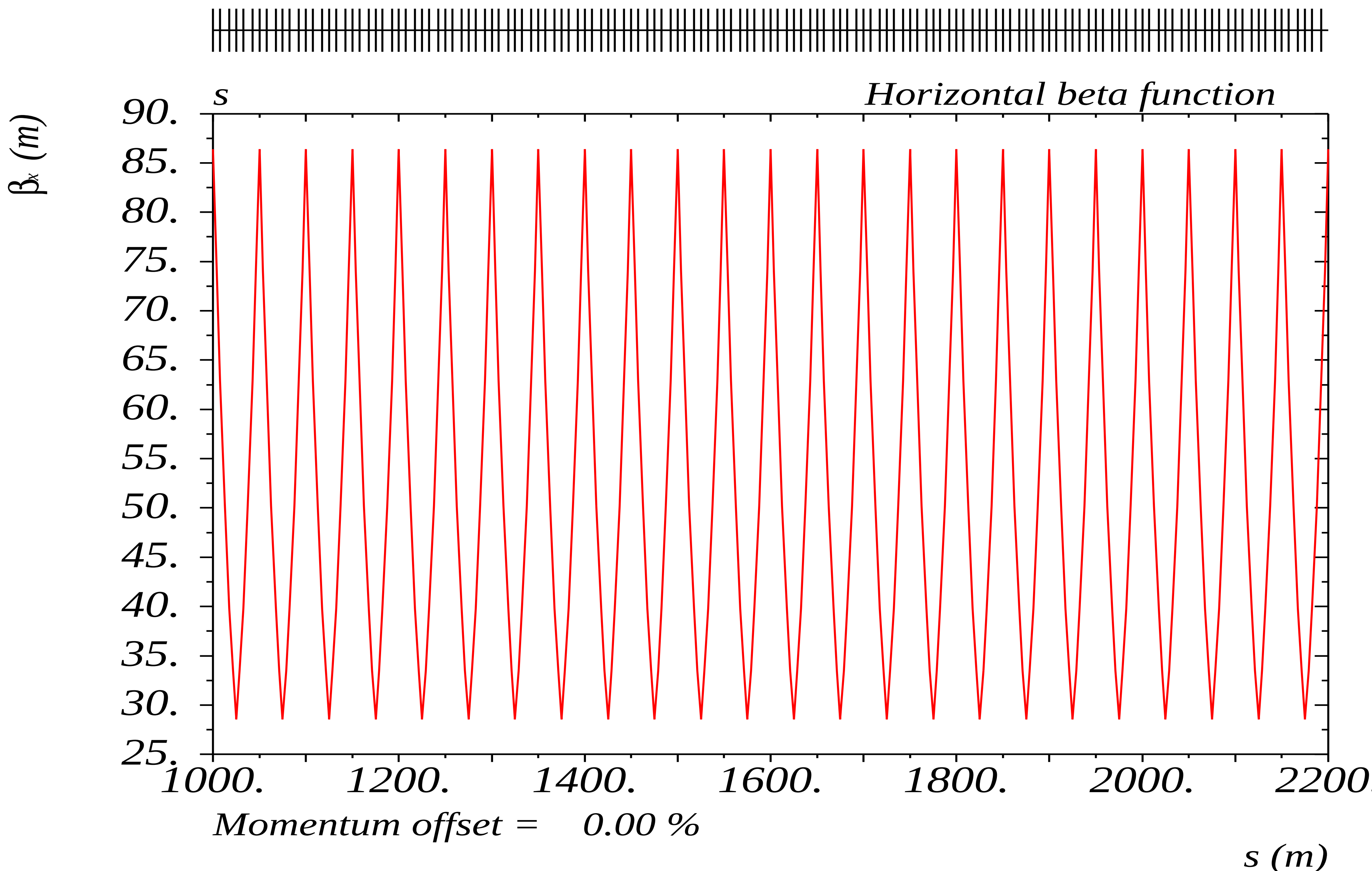
- **Starting from initial optical (Twiss) functions at s_0, transforming β_x, β_y, α_x, ... through the lattice gives functions at all positions s.**
- **Question: what are the β-functions etc. of a linear accelerator or a beam line ???**

Computation of optical functions around the ring

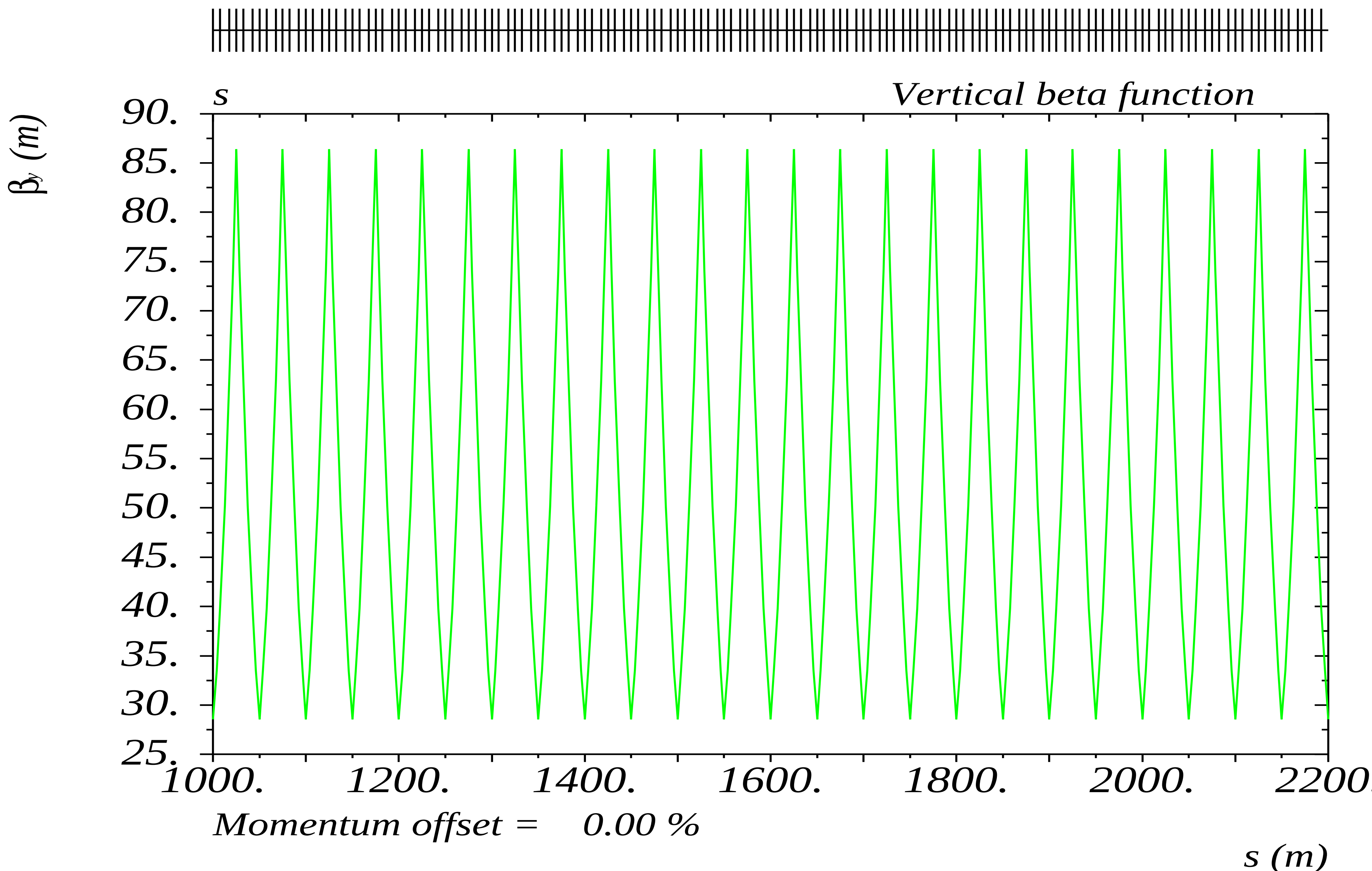
$$\begin{pmatrix} \beta \\ \alpha \\ \gamma \end{pmatrix}_{s} = \begin{pmatrix} m_{11}^2 & -2m_{11}m_{12} & m_{12}^2 \\ -m_{11}m_{21} & m_{11}m_{22} + m_{12}m_{21} & -m_{12}m_{22} \\ m_{21}^2 & -2m_{21}m_{22} & m_{22}^2 \end{pmatrix} \circ \begin{pmatrix} \beta \\ \alpha \\ \gamma \end{pmatrix}_{s_0}$$

Successive application of matrices give Twiss functions at each element around the ring and at each position s →

Optical functions (horizontal β):



Optical functions (vertical β):



Extension to two dimensions

- Can be written as separate equations, or:
- Extend vectors for coordinates or optical parameters
- Extend transfer maps/matrices

$$\begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_{s_2} = \begin{pmatrix} m_{11} & m_{12} & 0 & 0 \\ m_{21} & m_{22} & 0 & 0 \\ 0 & 0 & m_{33} & m_{34} \\ 0 & 0 & m_{43} & m_{44} \end{pmatrix} \circ \begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_{s_1}$$

Extension to two dimensions (coupling)

- **The horizontal and vertical motion can be coupled:**

→ **Additional elements in matrix**

$$\begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_{s_2} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{pmatrix} \circ \begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_{s_1}$$

Off momentum effects

- **Introduces longitudinal motion and off momentum trajectories**

Strength k of element modified by non-zero $\frac{\Delta p}{p}$:

$$k \Longrightarrow k/(1 \ + \ \frac{\Delta p}{p})$$

- **Closed orbit and tune are usually different for non-zero $\frac{\Delta p}{p}$**

→ **Dispersion**

→ **Chromatic effects**

Going to three dimensions

- **Formally extended by adding two new variables:**
  - $\Delta s = c\Delta t$: **longitudinal displacement with respect to reference particle**
  - $\frac{\Delta p}{p}$: **relative momentum difference with respect to reference particle**

$$\begin{pmatrix} x \\ x' \\ y \\ y' \\ c\Delta t \\ \frac{\Delta p}{p} \end{pmatrix}_{s_2} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} & m_{16} \\ m_{21} & m_{22} & m_{23} & m_{24} & m_{25} & m_{26} \\ m_{32} & m_{32} & m_{33} & m_{34} & m_{35} & m_{36} \\ m_{42} & m_{42} & m_{43} & m_{44} & m_{45} & m_{46} \\ m_{32} & m_{32} & m_{33} & m_{34} & m_{55} & m_{56} \\ m_{62} & m_{62} & m_{63} & m_{64} & m_{65} & m_{66} \end{pmatrix} \begin{pmatrix} x \\ x' \\ y \\ y' \\ c\Delta t \\ \frac{\Delta p}{p} \end{pmatrix}_{s_1}$$

(Interlude III: Symplecticity)

- **Not all possible maps are allowed !**
- **Requires for a matrix** $\mathcal{M} \rightarrow \mathcal{M}^T \cdot S \cdot \mathcal{M} = \mathcal{S}$

  with:

$$S = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

- **It basically means:** $\mathcal{M}$ **is area preserving and**

$$\lim_{n \to \infty} \mathcal{M}^n = \text{finite}$$

Introducing non-linear elements

Effect of a (short) quadrupole depends linearly on amplitude (re-written from the matrix form):

$$\begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_{s_2} = \begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_{s_1} + \begin{pmatrix} 0 \\ k_1 \cdot x_{s_1} \\ 0 \\ k_1 \cdot y_{s_1} \end{pmatrix}$$

- $\vec{z}(s_2) \;=\; \mathbf{M} \;\cdot\; \vec{z}(s_1)$
- $\mathbf{M}$ **is a matrix**

Non-linear elements (e.g. sextupole)

Effect of a (thin) sextupole with strength k_2 is:

$$\begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_{s_2} = \begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix}_{s_1} + \begin{pmatrix} 0 \\ k_2 \cdot (x_{s_1} \cdot y_{s_1}) \\ 0 \\ \frac{1}{2} k_2 \cdot (x_{s_1}^2 - y_{s_1}^2) \end{pmatrix}$$

- $\vec{z}(s_2) = \mathcal{M} \circ \vec{z}(s_1)$
- **$\mathcal{M}$ is not a matrix, i.e. cannot be expressed by matrix multiplication**

Non-linear elements

Cannot be written in linear matrix form !

We need something like:

$$\begin{aligned} x \quad = \quad & R_{11} \cdot x \quad + R_{12} \cdot x' \quad + R_{13} \cdot y + ... \\ & + T_{111} \cdot x^2 + T_{112} \cdot xx' + T_{122} \cdot x'^2 + \\ & + T_{113} \cdot xy + T_{114} \cdot xy' + ... \\ & + U_{1111} \cdot x^3 + U_{1112} \cdot x^2x' + ... \end{aligned}$$

and the equivalent for all other variables ...

Higher order MAPS:

Definition of a second order map $\mathcal{A}_2$ (Taylor expansion):

$$\mathrm{Let's\ call\ it:}\quad \mathcal{A}_2 \;=\; [R, T], \quad \mathrm{defined\ as} \qquad (\mathrm{for: j = 1...4}):$$

$$z_j(s_2) \;=\; \sum_{k=1}^{4} R_{jk} z_k(s_1) \;+ \sum_{k=1}^{4}\sum_{l=1}^{4} T_{jkl} z_k(s_1) z_l(s_1)$$

Higher orders can be defined as needed ...

$$\mathcal{A}_3 \;=\; [R, T, U] \quad \Longrightarrow \qquad + \sum_{k=1}^{4}\sum_{l=1}^{4}\sum_{m=1}^{4} U_{jklm} z_k(s_1) z_l(s_1) z_m(s_1)$$

Second order MAPS composition:

Assume now 2 maps of second order:

$\mathcal{A}_2 = [R^A, T^A]$ **and** $\mathcal{B}_2 = [R^B, T^B]$

the combined second order map

$\mathcal{C}_2 \; = \; \mathcal{A}_2 \circ \mathcal{B}_2$ **is** $\mathcal{C}_2 \; = \; [R^C, T^C]$ **with:**

$$R^C \; = \; R^A \cdot \; R^B$$

and (after truncation of higher order terms !!):

$$T^C_{ijk} \; = \; \sum_{l=1}^{4} R^B_{il} T^A_{ljk} \; + \; \sum_{l=1}^{4} \sum_{m=1}^{4} T^B_{ilm} R^A_{lj} R^A_{mk}$$

Symplecticity for higher order MAPS

- **Truncated Taylor expansions are not matrices !!**
- **It is the associated Jacobian matrix $\mathcal{J}$ which must fulfil the symplecticity condition:**

$$\mathcal{J}_{ik} = \frac{\delta z_2^i}{\delta z_1^k}$$

$$\mathcal{J} \quad \textbf{must fulfil:} \quad \mathcal{J}^t \cdot S \cdot \mathcal{J} = \mathcal{S}$$

- **In general: $\mathcal{J}_{ik} \neq$ const $\rightarrow$ for truncated Taylor map can be difficult to fulfil for all z**

(Interlude IV: Other higher order MAPS)

There are other types of higher order maps:

- **Lie transformations (always symplectic for any order, ideal for tracking, easy to analyse)**
- **Symplectic or canonical integration algorithms**
- **whatever ...**

→ **Intermediate level school ... !**

Matching optical functions

- Modify machine optics to get desired properties around the machine or in specific places
- For example you may want special conditions
  - for equipment: RF, collimators, diagnostics
  - for experiments: in colliding beam machines
- Algorithms to adjust parameters and layout

→ This process is called MATCHING !

→ Available in most optics programs (for lines and circular machines)

Matching optical functions

In general: $\vec{\nu}(\beta_x(s), \alpha_x(s) ...) \;=\; f(\mathcal{M}_1, \mathcal{M}_2, ...\mathcal{M}_n)$
optical functions depend on all maps, i.e. strengths and layout.

- **Matching can change all parameters (strengths, positions ...)**
- **Additional constraints may be:**
  - **Tunes, chromaticities**
  - **Hardware parameters**
  - $\hat{\beta}_x$, **etc.**

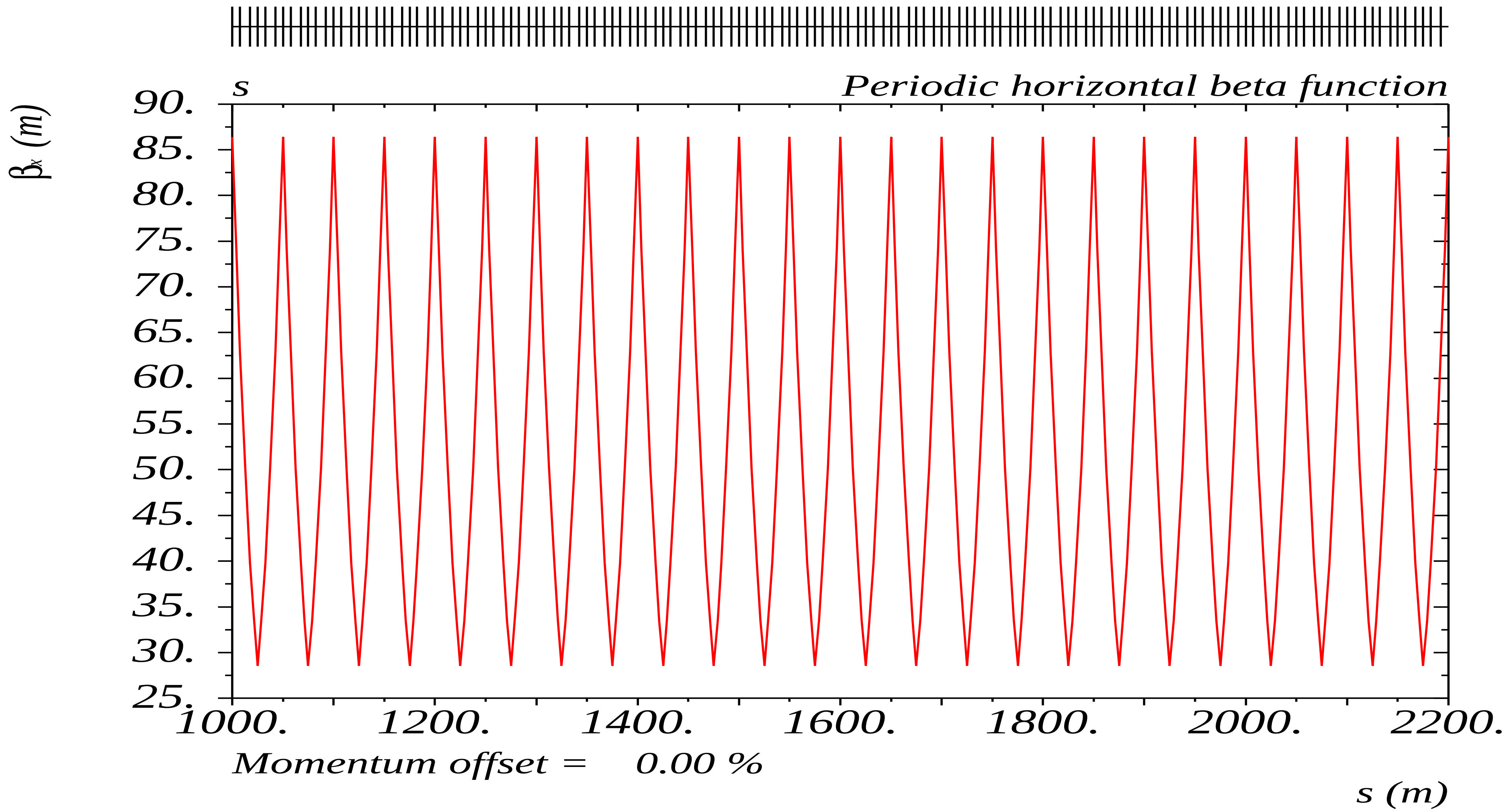
Example: matching of tunes Q (MAD language)

```
match;
   vary,name=kqf, step=0.00001;
   vary,name=kqd, step=0.00001;
   global,QX=7.420;
   global,QY=7.380;
endmatch;
```

- **The strengths of main quadrupoles are varied**
- **Computes the new strengths**

Local adjustment of optical functions

- **Small β_x needed for an experiment**



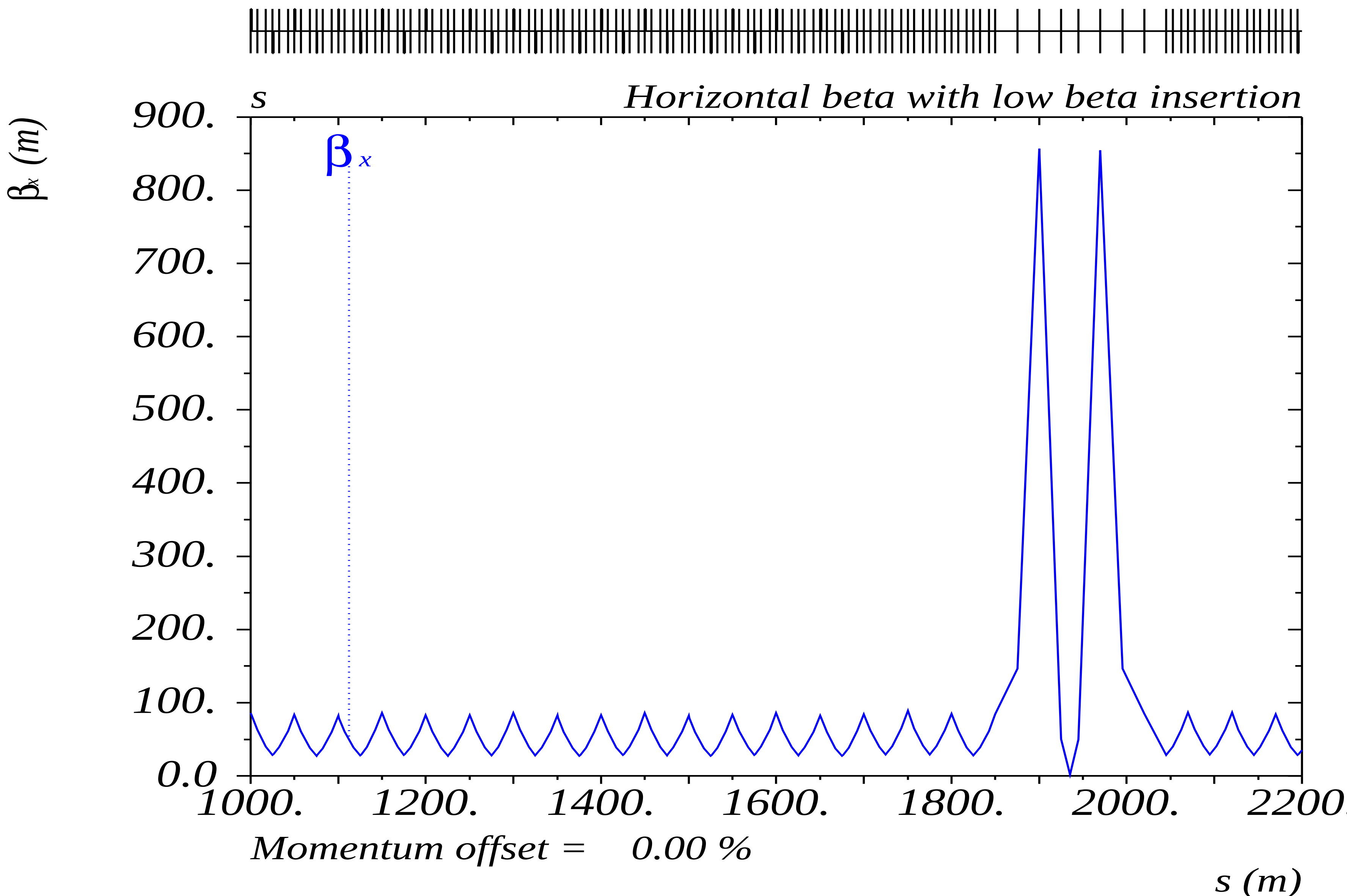
Example: matching of β-functions (MAD language - simplified !)

- **Allow a few quadrupoles to be changed**

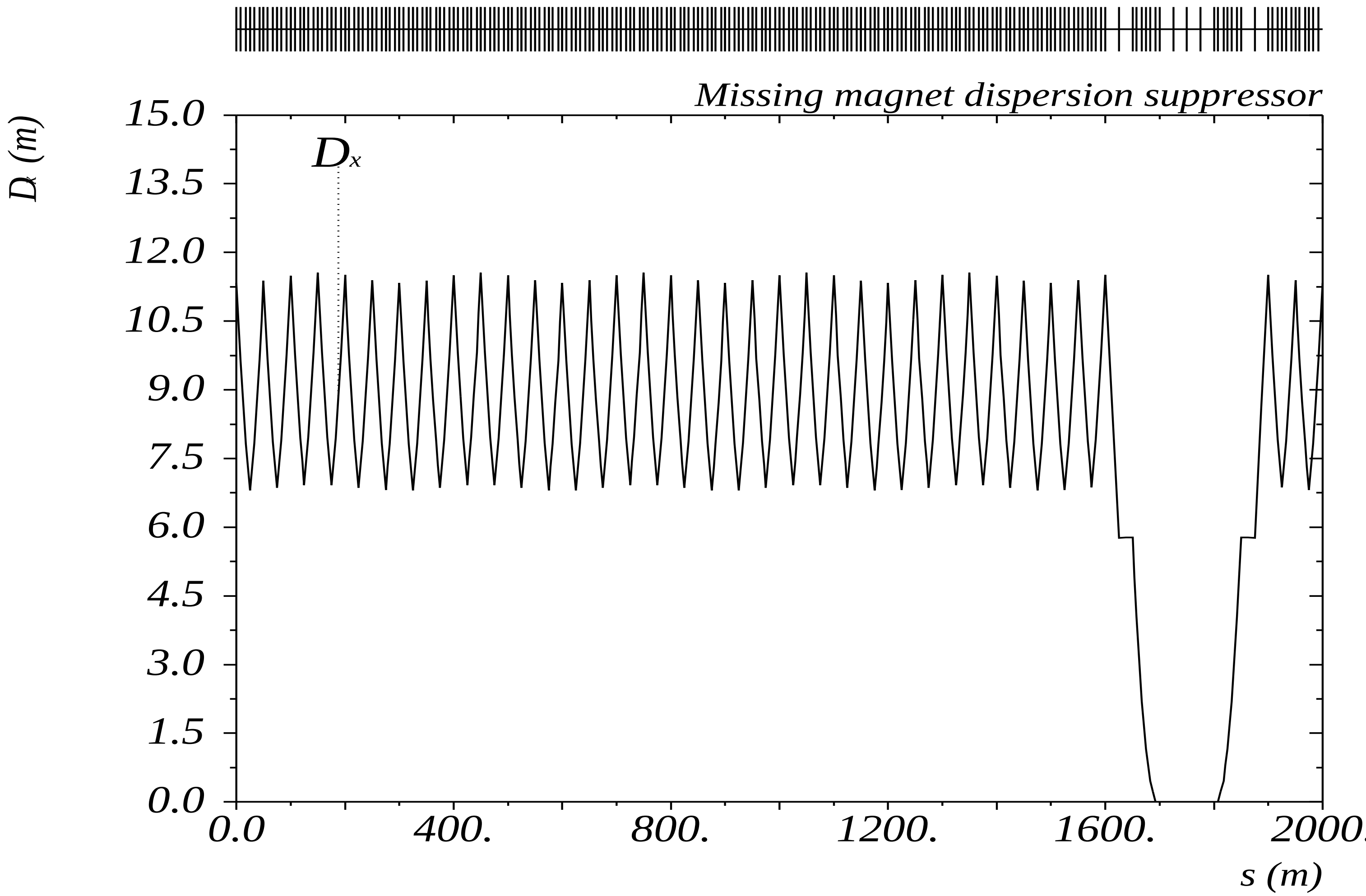
```
match;
    vary,name=kq1.l, step=0.00001;
    vary,name=kq2.l, step=0.00001;
    vary,name=kq3.l, step=0.00001;
    vary,name=kq4.l, step=0.00001;
    constraint,place=IP,betx= 2.0, alfx=0.0;
endmatch;
```

- **The strengths of a small number of additional quadrupoles are varied**

Optical functions (reduced β-function)



Optical functions (reduced dispersion)



General purpose optics programs

- **Always allow to:**
  - **Compute optical parameters (Twiss functions)**
  - **Match the required properties**
- **But also:**
  - **Simulate machine imperfections**
  - **Correct imperfections**

Popular Optics Programs

- BeamOptics (based on Mathematica)
- TURTLE (Beam lines)
- WINAGILE (WINDOWS, interactive, originally for teaching)
- TRANSPORT (General purpose, third-order matrix)
- DIMAD (Second-order matrix, tracking)
- TEAPOT (General purpose, thin element approximation)

Popular Optics Programs (cont.)

- **COSY (Multi purpose, high order maps, differential algebra)**
- **SYNCH (General purpose)**
- **MAD (versions: 8,9,X) (Multi purpose optics and tracking)**
- **SAD (Multi purpose optics and tracking)**
- **MARYLIE (Lie algebra, tracking)**
- **PTC (MAP based, object oriented, exact !)**
- **MADX-PTC (combined MADX-PTC)**

Which Optics Program should I use ?

Very application dependent, you have:

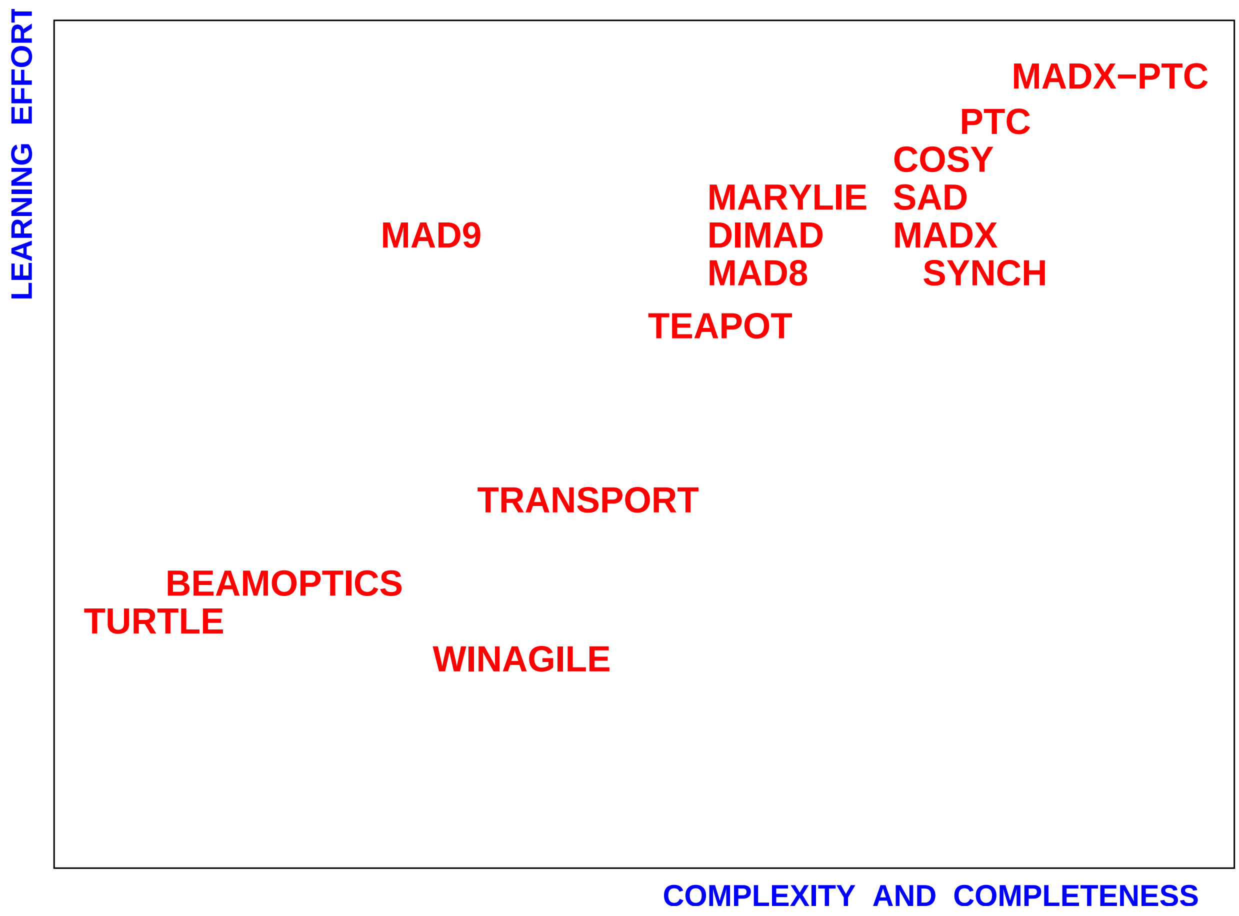
- ➡ **Beam line**
- ➡ **Large ring**
- ➡ **Small ring**
- ➡ **Large momentum offset or changing momentum (e.g. FFAG, acceleration)**
- ➡ **Linear accelerator**
- ➡ **Unconventional geometry**
- ➡ **Collider (one or more rings/lines)**

Which Optics Program should I use ?

Very application dependent, you want to do:

- **Design linear optics**
- **Linear optical matching**
- **Introduce and correct imperfections**
- **Non-linear optical matching**
- **Particle tracking**
- **Evaluate the dynamic aperture**
- **Study collective effects**

"Comparison" of some programs



→ **Very difficult to quantify, Strongly biased**

(Interlude V: Course on optics design)

- Intermediate level CAS 2005 (and 2007) offers a course on optics design
  - Purpose is to develop a realistic accelerator optics
  - Includes correction elements, optical matching, dispersion suppressors ...
  - MAD is used for practical implementation
- The course is available on CD-ROM (on request) or from the web

Simulation of an accelerator

- Purpose is to imitate the behaviour of a particle or a beam in an accelerator
- Use local properties of the machine element to describe its interaction with a particle
- The aim is to derive the global behaviour
  - Stability
  - Lifetime
  - ....

Evaluation by simulation (1)

- **Single particle behaviour**
- **Usually concerns long term behaviour such as beam loss**
- **The effect of the accelerator components on a single particle**
  - **Non-linear elements**
  - **Machine imperfections (e.g. field errors)**
  - **External distortions (e.g. scattering)**

Evaluation by simulation (2)

- Multi particle behaviour
- Usually concerns collective behaviour: coherent motion, emittance increase, damping etc.
  - The effect of the accelerator components on an ensemble of particles (e.g. impedances)
  - Interactions of particles between each other. These are usually dictated by the properties of the accelerator (e.g. space charge, beam-beam effects ...)

Single particle tracking

- The motion of a test particle through the elements of a machine is simulated for a defined number of turns $\rightarrow$ "tracking"
- Use appropriate coordinates, start with $\vec{z_0}$.
- In each element (or part of the machine), the coordinates are transformed by $\vec{z_{n+1}} = \mathcal{M} \circ \vec{z_n}$
- $\mathcal{M}$ must be symplectic
- We must distinguish thick and thin elements

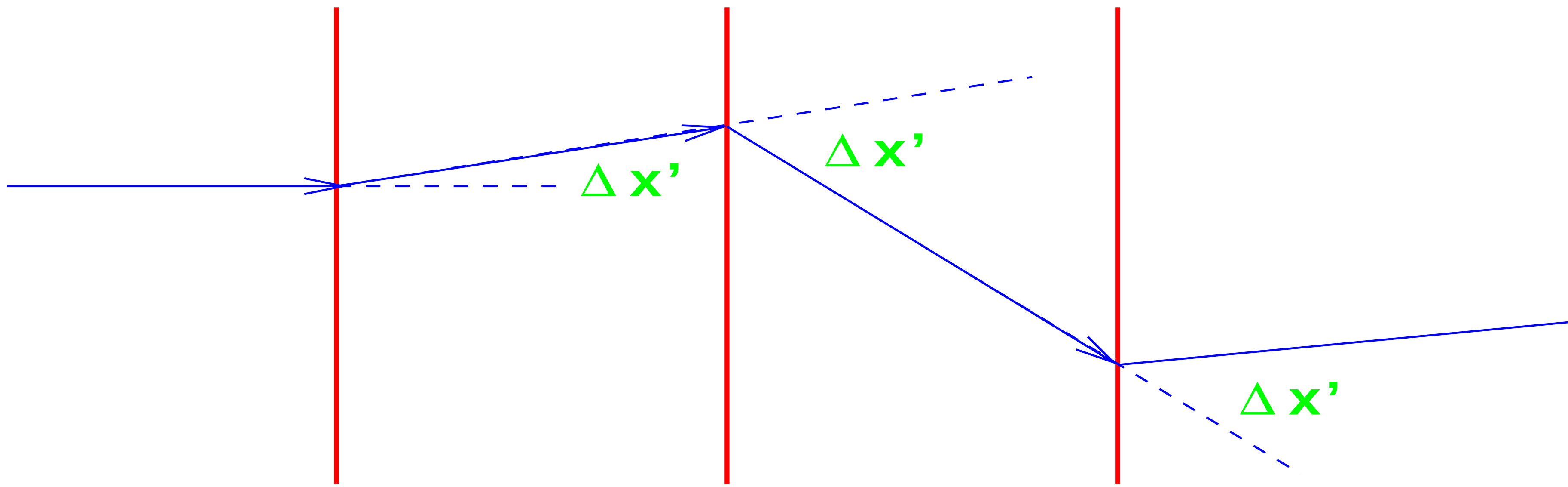
Tracking through thin elements

- **Thin "magnet": let the length go to zero, but keep field integral finite (constant)**
- **No change of amplitudes x and y**

→ $x \rightarrow x$ **and** $y \rightarrow y$

- **The momenta x' and y' receive an amplitude dependent deflection (kick)**

→ $x' \rightarrow x' + \Delta x'$ **and** $y' \rightarrow y' + \Delta y'$



- **No change of amplitudes x and y**
- **The momenta x' and y' receive an amplitude dependent deflection (kick)**

→ $x' \rightarrow x' + \Delta x'$ **and** $y' \rightarrow y' + \Delta y'$

Tracking through thin elements

- So-called kick-codes (thin lens tracking)
- Always symplectic ! (homework)
- Kick can be non-linear
- Usually rather fast on computers

→ A "thick" element can be sliced into several thin elements

Analysis tools

- **Fourier analysis, diffusion coefficients, chaos detection ...**
- **Phase space plots (simple example):**
  - **Start with a "particle" with initial coordinates x and x' at a position s_0**
  - **Pass through the One-Turn-Map (for position s_0 !)**
  - **Plot the new x and x' coordinates at position s_0 after every turn**

A simple example

Linear transformation plus a constant deflection (i.e. orbit kick from displaced quadrupole)

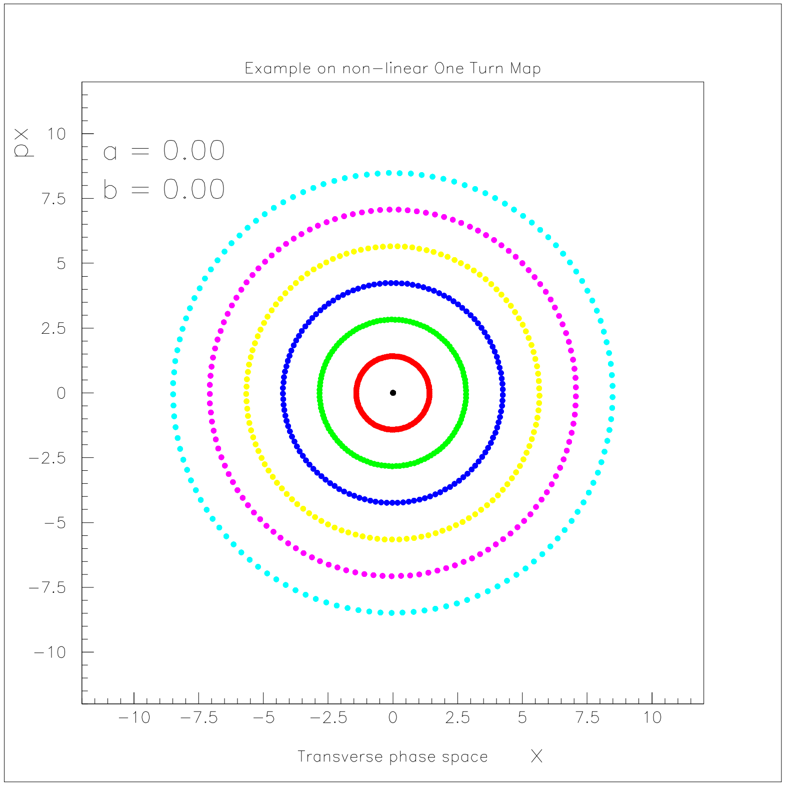
$$\begin{pmatrix} x \\ x' \end{pmatrix}_{n+1} = \begin{pmatrix} \cos(\mu) & \sin(\mu) \\ -\sin(\mu) & \cos(\mu) \end{pmatrix}_{s_0} \cdot \begin{pmatrix} x \\ x' \end{pmatrix}_{n} + \begin{pmatrix} 0 \\ b \end{pmatrix}$$

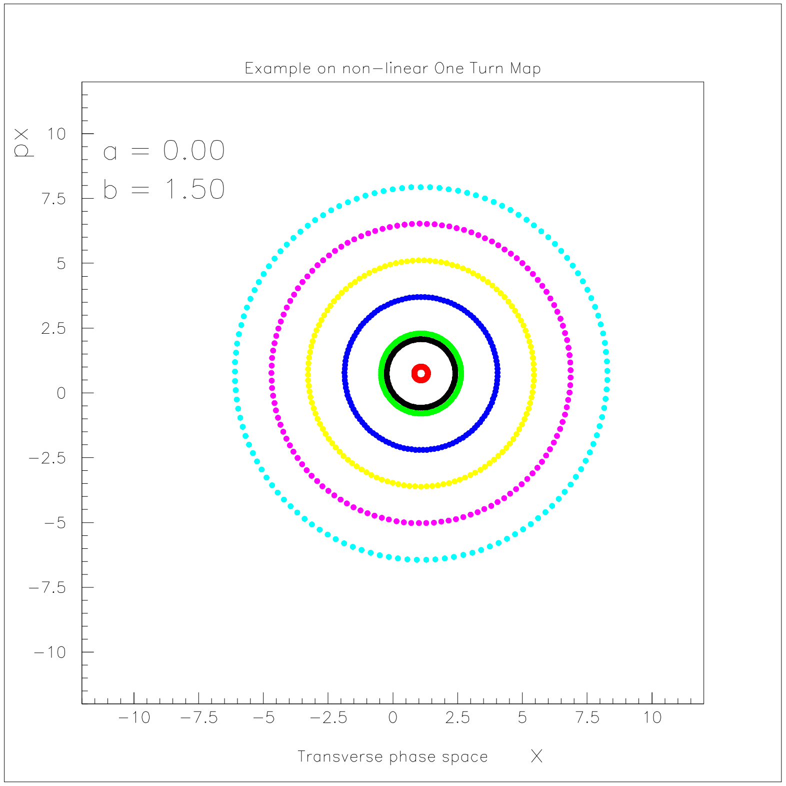
$\mu = 2\pi \mathrm{Q}_x = \mathbf{2}\pi\cdot\mathbf{0.19}$,

constant b is a free parameter

→ **Find the fixed point(s) (closed orbit)**

A simple example ..





- **Start at different amplitudes and "observe" x and x' at position** s_0

A (still) simple example

Linear transformation plus a quadratic non-linearity (e.g. (thin) sextupole) plus a constant deflection

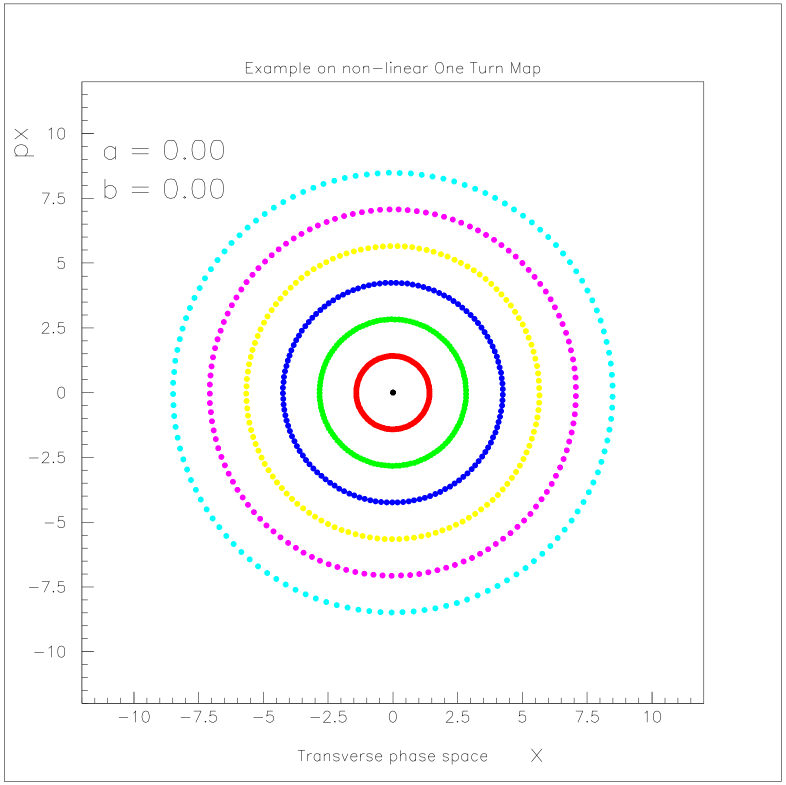
$$\begin{pmatrix} x \\ x' \end{pmatrix}_{n+1} = \begin{pmatrix} \cos(\mu) & \sin(\mu) \\ -\sin(\mu) & \cos(\mu) \end{pmatrix}_{s_0} \cdot \begin{pmatrix} x \\ x' \end{pmatrix}_{n} + \begin{pmatrix} 0 \\ ax^2 \end{pmatrix} + \begin{pmatrix} 0 \\ b \end{pmatrix}$$

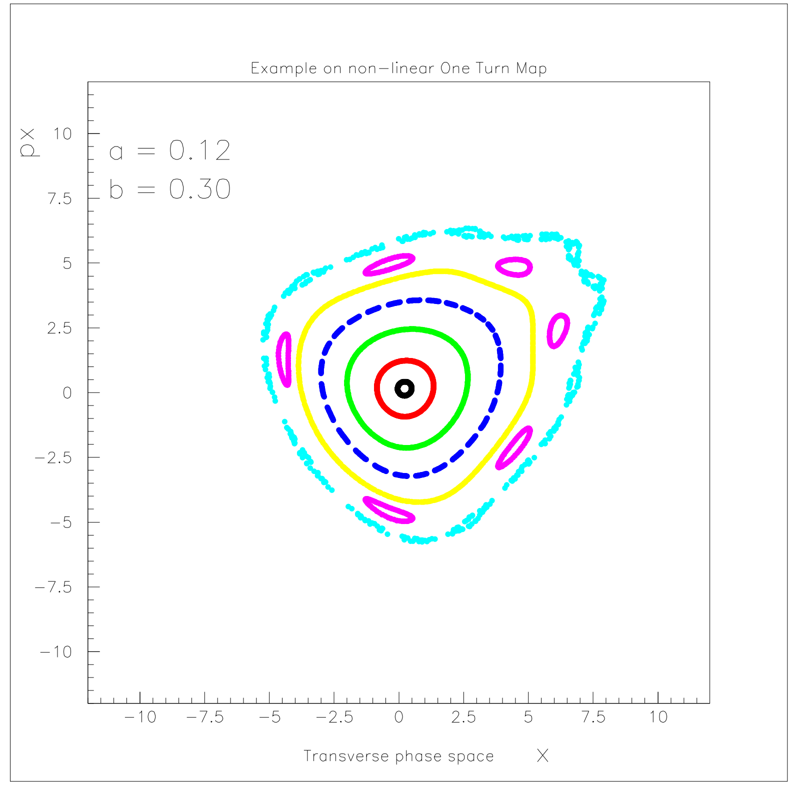
$\mu = 2\pi Q_x = \mathbf{2\pi \cdot 0.19}$,

constants a and b are free parameters

➡ **Find the fixed points (how many do you see ?)**

A (still) simple example ..





- **Motion at different amplitudes distorted**
- **Stability region reduced by non-linear effect**

Tracking through thick elements

- Real magnets are NOT thin !
- Consequence: they are not always linear elements (also not dipoles, quadrupoles ..)
- Especially important for "small" machines (e.g. large deflections, fringe fields etc.)
- Constructing a MAP for a thick element is more involved

MAPS for thick elements

- Non-linear, i.e. not matrices (even for quadrupoles)
- Need integration of equation of motion
- Special techniques have been developped, some keywords:
  - Lie transforms
  - Symplectic integration
  - Differential Algebra ...
- In the end: we get again a One-Turn-Map

Single particle dynamics in a nutshell

- Try to compute a (one turn) MAP
- It contains everything
- Its analysis will tell you what you need to know
- It doesn't matter how you got it !
- Use the Lorentz force (or equivalent) for the construction of maps (no need to "apply" accelerator physics concepts !)

You can derive:

- Tunes (we already did)
- Betatron functions (we already did)
- Stability borders, dynamic aperture
- Detuning with amplitudes
- Fixpoints, closed orbit, resonance strength
- ... and much more !

You do **not** get:

- Rules of thumb for simple calculations
- "Handy" formulae
- Scaling laws for estimates (indirectly)

Complications: light particles

- **Light particles (e^-, e^+ etc.) emit synchrotron radiation and motion is damped**
  - **Stochastic component**
  - **No symplecticity, no invariants (but equilibrium parameters, e.g. emittance)**
- **Synchrotron motion must be simulated**
- **Computation of damping properties**

Single particle tracking codes

- Most optics programs can perform single particle tracking
- Some specialized programs exist (e.g. SIXTRACK)
- Some codes have analysis tools (normal forms, chaos detection etc.)

Simulation of multi particle effects

- Often requires the simulation of a beam: simulate many (up to 10^8) particles simultaneously and study their behaviour:
  - Beam shape (density distribution)
  - Centre of mass motion of all particles
- Must be self-consistent: changes of the beams must be taken into account
- Fields generated by the beam need to be computed

Complications: many particles

- Particles have different amplitudes
- Particles have different tunes
- Particles have different momenta !
- Definition of emittance becomes more complicated

Strategy for multi-particle simulations (1)

- Generate and simulate many particles ($10^4 - 10^8$ per beam) simultaneously
- Every particle interacts with the machine elements individually
- The whole ensemble interacts with the machine elements
- Every particle interacts with other particles !

→ Feed back into motion of individual particles

Strategy for multi-particle simulations (2)

- **All particles must be treated in parallel**
  - **For realistic LHC: 10^7 to 10^8 particles to simulate**
  - **Already for storage requires $\approx$ 10 Gb memory**
  - **Parallel processing needed for reasonable computing time**
- **Often requires (intelligent) simplifications**

Simulation of interactions with environment

This means: interaction of the individual particles **and** the whole beam with:

- Machine elements (e.g. magnets, RF, ...)
- Wake fields
- Impedances
- Electron cloud
- Intercepting elements (e.g. collimators, ...)

→ Strategies have changed with fast computers ...

Simulation of interactions with itself

Particles inside a beam interact with other particles from the same beam:

- **Space charge effects**
- **Intra-beam scattering**
- **Multi-bunch effects**

Simulation of interactions with other beams

So-called beam-beam effects

- Other beam acts like a (very) non-linear lens
- Incoherent beam-beam effects (on each individual particle)
- Coherent beam-beam effects (on ensemble of particles)

→ Requires the knowledge of fields generated by the other beam

Matrix formulation (linear models)

- **One-Turn-Maps can be written for two bunches or two beams (e.g. 1 and 2)**

→ **Here only 1 dimension for illustration**

$$\begin{pmatrix} x_1 \\ x_1' \\ x_2 \\ x_2' \end{pmatrix}_{n+1} = \begin{pmatrix} m_{11} & m_{12} & 0 & 0 \\ m_{21} & m_{22} & 0 & 0 \\ 0 & 0 & m_{33} & m_{34} \\ 0 & 0 & m_{43} & m_{44} \end{pmatrix} \circ \begin{pmatrix} x_1 \\ x_1' \\ x_2 \\ x_2' \end{pmatrix}_{n}$$

Matrix formulation (linear models)

- **One-Turn-Maps can be written for two bunches or two beams (e.g. 1 and 2)**

→ **Additional elements couple two beams**

$$\begin{pmatrix} x_1 \\ x_1' \\ x_2 \\ x_2' \end{pmatrix}_{n+1} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{pmatrix} \circ \begin{pmatrix} x_1 \\ x_1' \\ x_2 \\ x_2' \end{pmatrix}_{n}$$

→ **Allows computation of eigenmodes, eigenfrequencies of multi bunch systems**

Field computation

- Some simulations require the computation of fields (or forces) produced by a beam from Poisson equation (here 2-dimensional):

$$\Delta V = -4\pi \cdot \rho(x, y)$$

- → The density of the beam is $\rho(x, y)$
- → For simple distributions (Gaussian, uniform ...) can be solved analytically
- → In general (i.e. in the interesting cases !) it is done numerically

Some basic methods

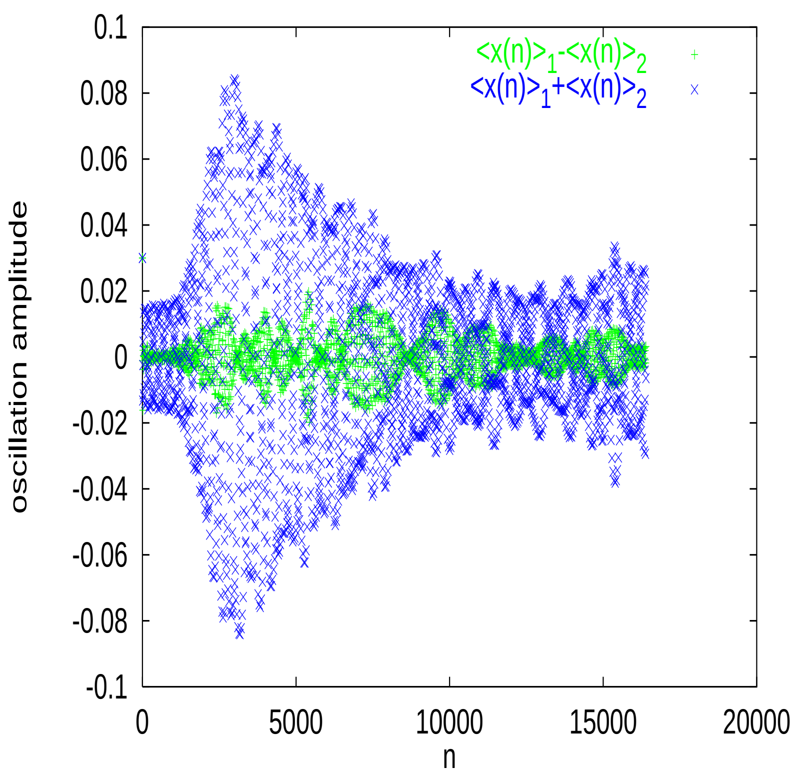
Particle - particle methods: compute field between each particle pair and add up (not practical for large number of particles, sometimes used in celestial mechanics)

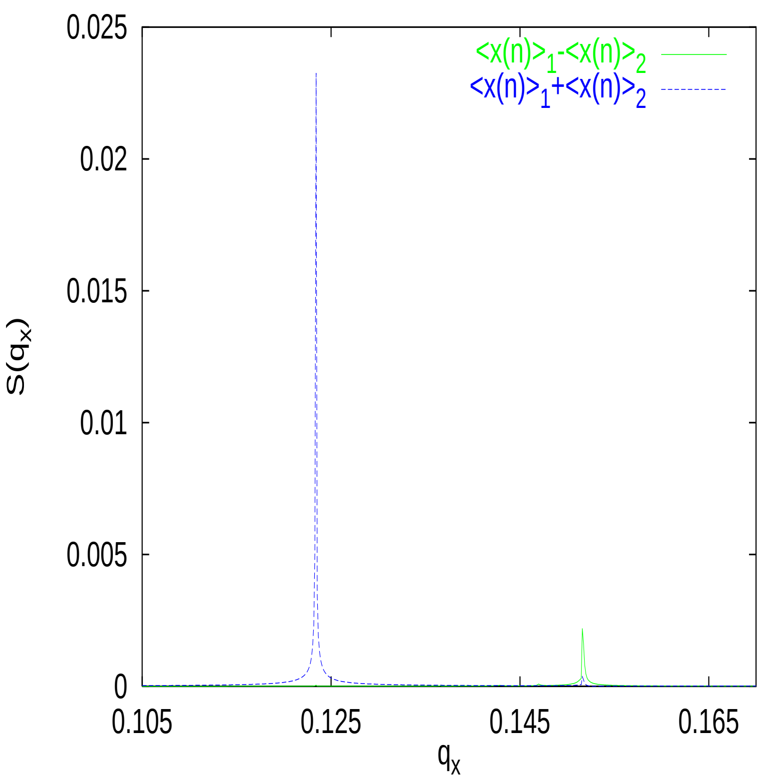
Particle - mesh methods: distribute particles on a mesh (grid) and solve the Poisson equation for discrete points

Multipole methods: develope potentials/fields as multipole expansion

→ Choice depends on application and parameters

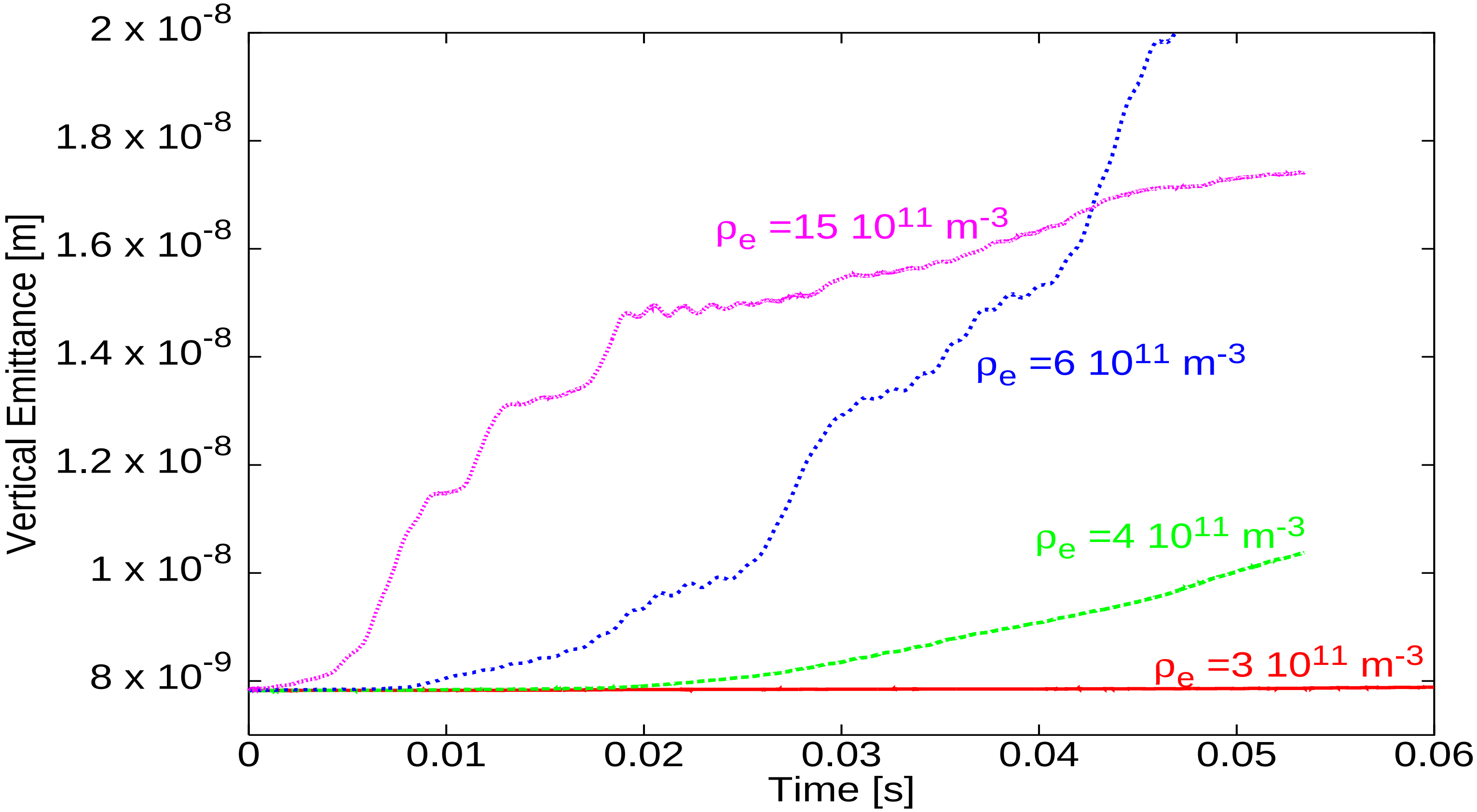
Example: Centre of mass motion as function of tim





➡ **From beam-beam simulations: Bunch oscillations and frequency spectrum**

Example: Beam size as function of time



From electron-cloud simulation (courtesy: E. Benedetto)

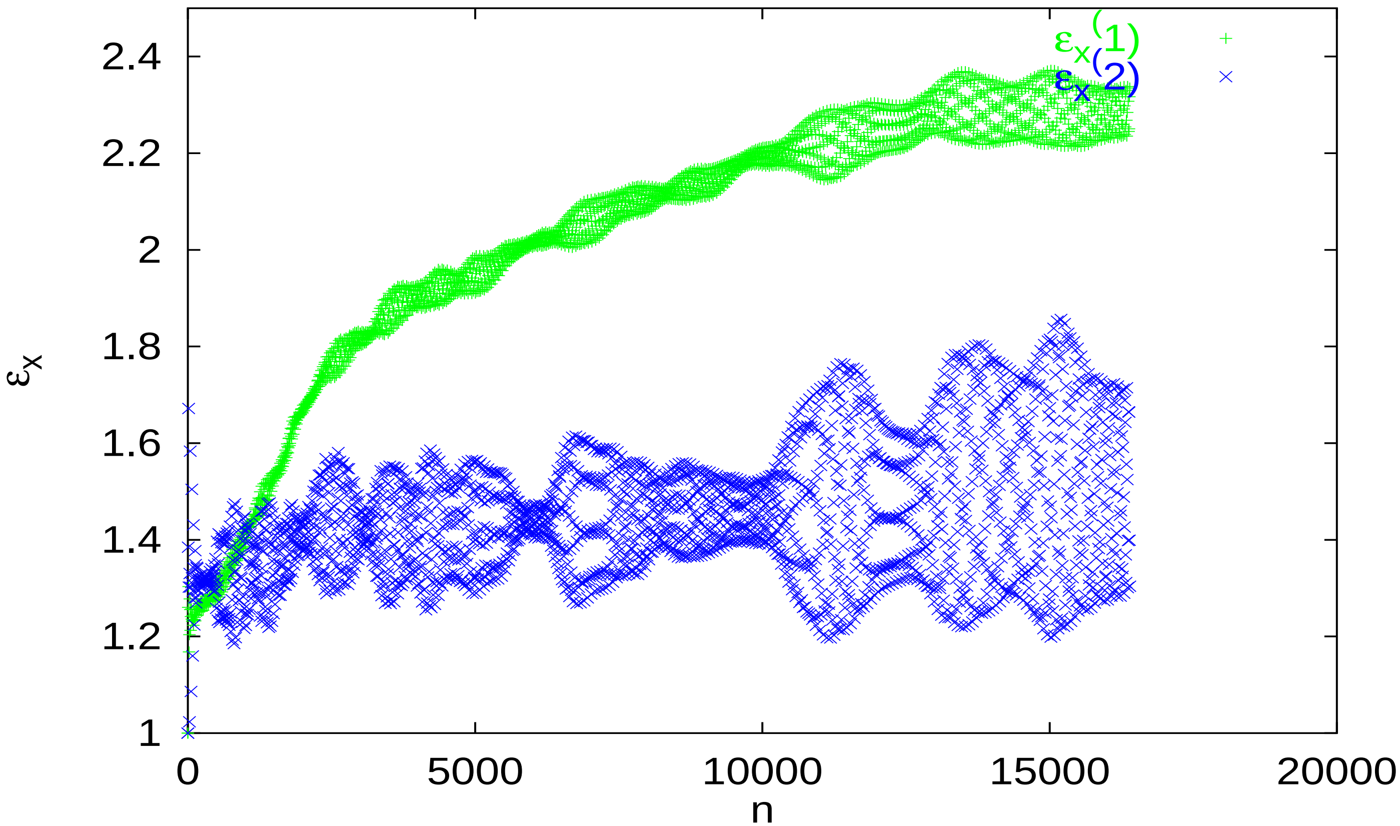
Alternatives

- Sometimes multi-particle simulations are too time consuming
- Numerical solution of the Vlasov-equation

→ E.g. finite difference methods

→ Intermediate level school

Example: Beam size as function of time



➡ **Beam-beam simulation: numerical solution of Vlasov equation gives evolution of beam sizes**

Multi particle simulation codes

- **Many codes exist, always specialized:**
  - **Collective instabilities**
  - **Beam-beam effects**
  - **Electron cloud effects**
  - **etc. ...**
- **Often compact and linked to optics codes**

(Personal) Comments on simulations:

- Here I gave only a selective overview of what can be done
- Techniques and tools in dedicated schools and courses (some in Intermediate CAS Course)
- What can be done has changed a lot in the last decade
- It is easy to write a program !
- Analysis and interpretation is usually the difficult part

What can go wrong ?

- Wrong or missing physics in the program
- Numerical problems
- Different results on different computers
- Programming bugs ...
- Biased analysis

→ Be aware of the limitations of the program

→ Make sure it is reproducible

Control and operation

- **Basic aim: optimize performance**

As operator or accelerator physicist:

- **Provide and improve model of machine**
- **Measure and interprete beam parameters**
- **Correct and control beam parameters**
- **Conduct machine experiments**

Control and operation of an accelerator

Basic problem: measure and control beam parameters

- Control (orbit, chromaticities ..) depend on machine model which may be incomplete
- Feedback from measurements improves the model and simulations
- Should use the same strategies and methods as during design (Remember: matching !)

→ May be an iterative process

Control and operation of an accelerator

- **Very similar to simulation or design of a machine except:**
  - **Interface to hardware and control (e.g. power converters)**
  - **Beam instrumentation !**
  - **Communication (networks, etc.)**
  - **Issues such as: timing, alarms, interlocks,**
- **Treated in dedicated workshops and schools**