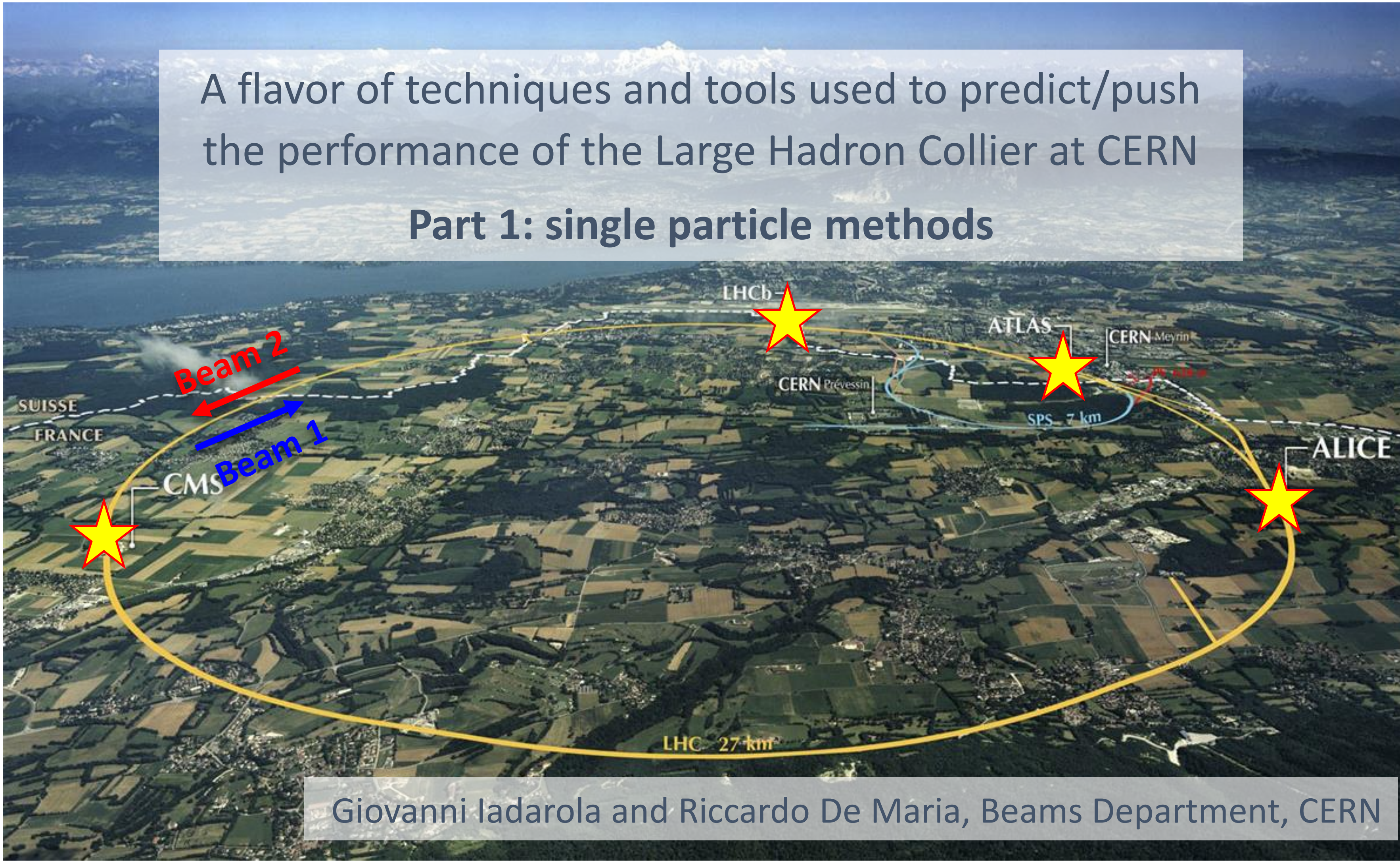
Numerical simulation of beam dynamics in particle accelerators

A flavor of techniques and tools used to predict/push the performance of the Large Hadron Collier at CERN

Part 1: single particle methods



Giovanni Iadarola and Riccardo De Maria, Beams Department, CERN

- **Particle accelerators**
  - Examples of applications
  - Working principles

- **Beam optics calculations**
  - An example: the LHC betatron squeeze
  - Numerical optimization techniques

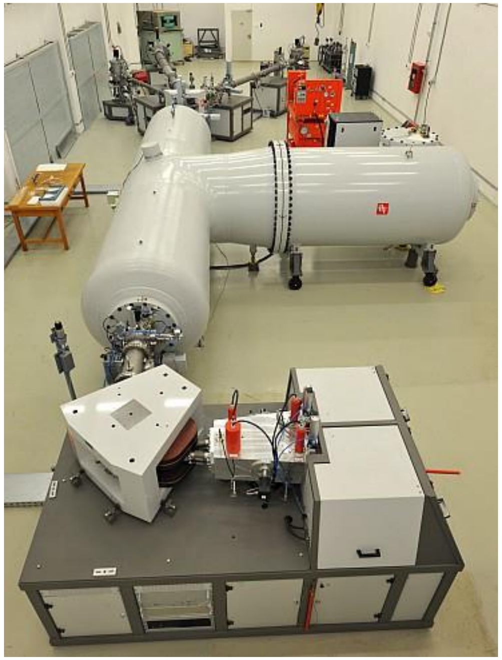
- **Particle tracking**
  - Motivations
  - Need for symplectic methods
  - Experience with GPU computing

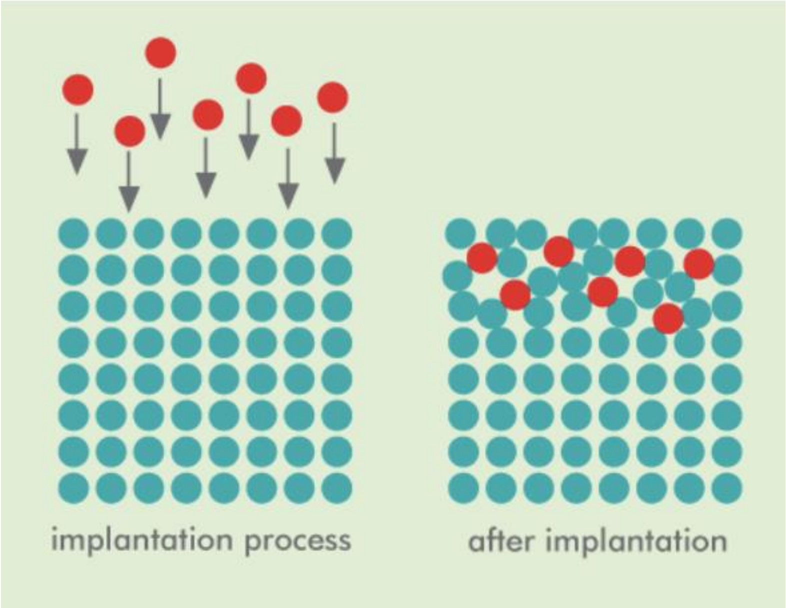
A **particle accelerator** is a machine used to **accelerate particles** (electrons, protons, ions, etc…) using **electromagnetic fields**

→ There are **many accelerators** in the word used for a **variety of applications** (industrial, medical, research)

Example of industrial application:

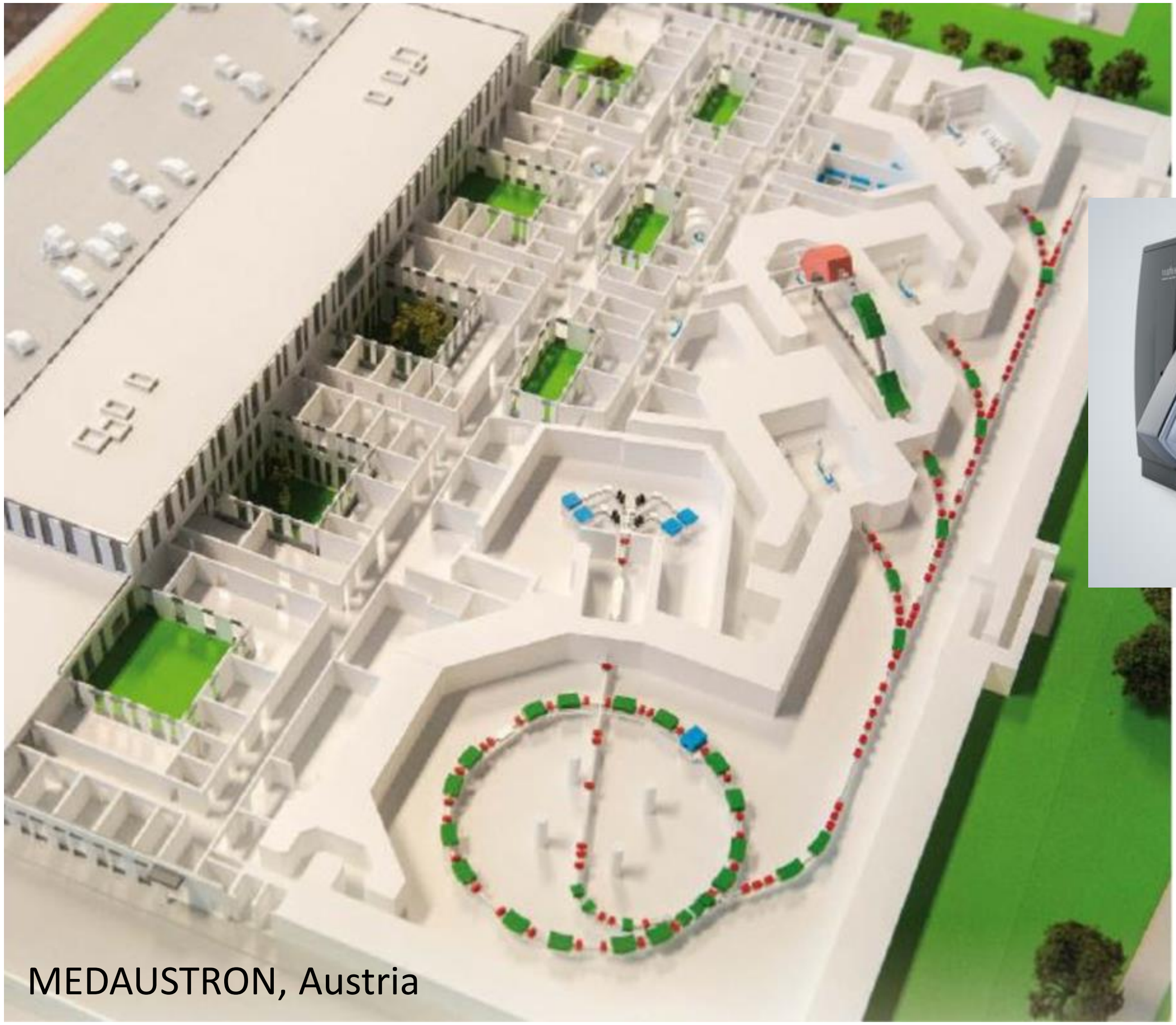
- **Ion implantation** in the fabrication of integrated circuits





Example of medical application:

- Accelerator for **cancer treatment**



Introduction

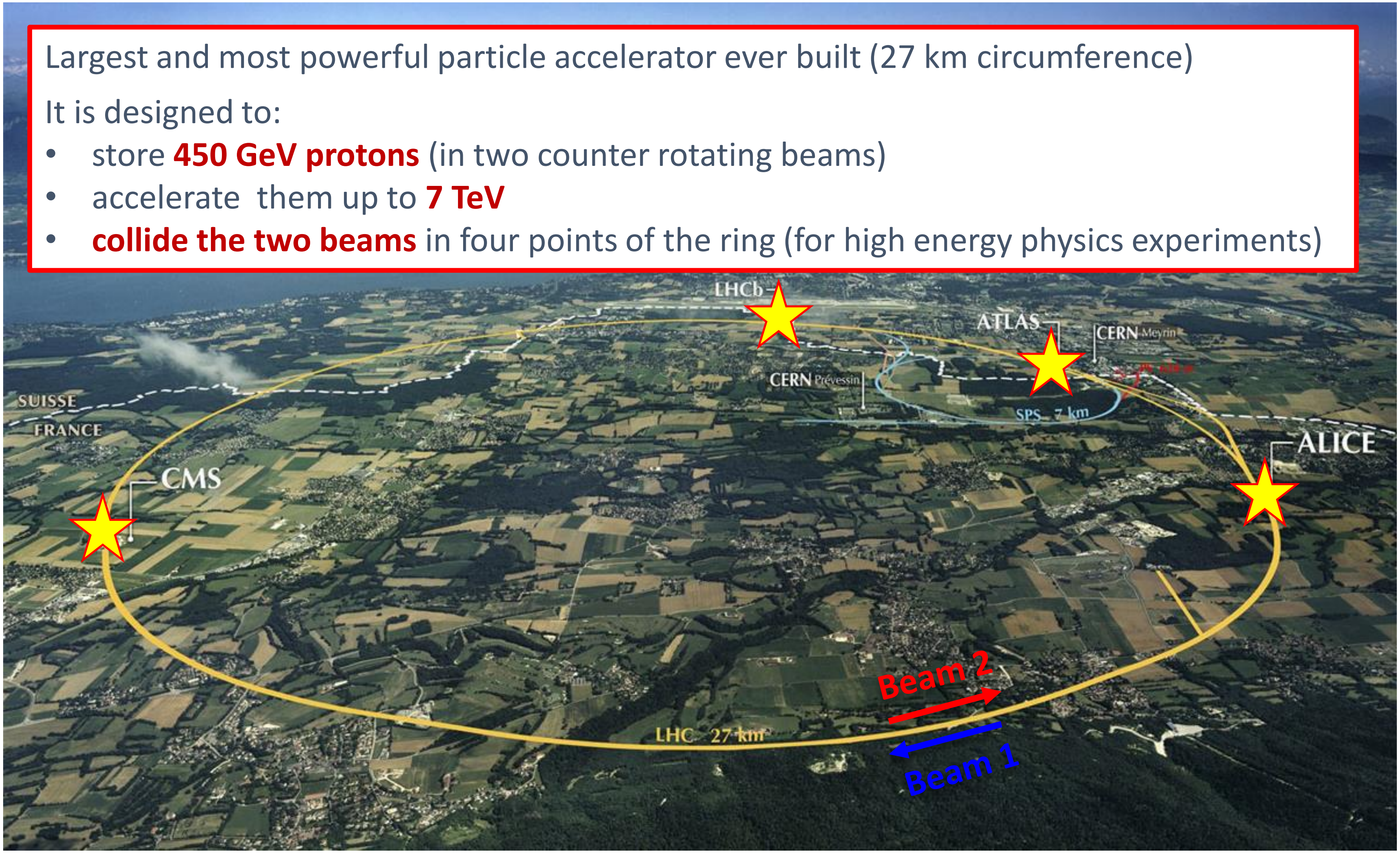
Example of research application:

- **Large Hadron Collider (LHC)** for high-energy physics research

Largest and most powerful particle accelerator ever built (27 km circumference)

It is designed to:

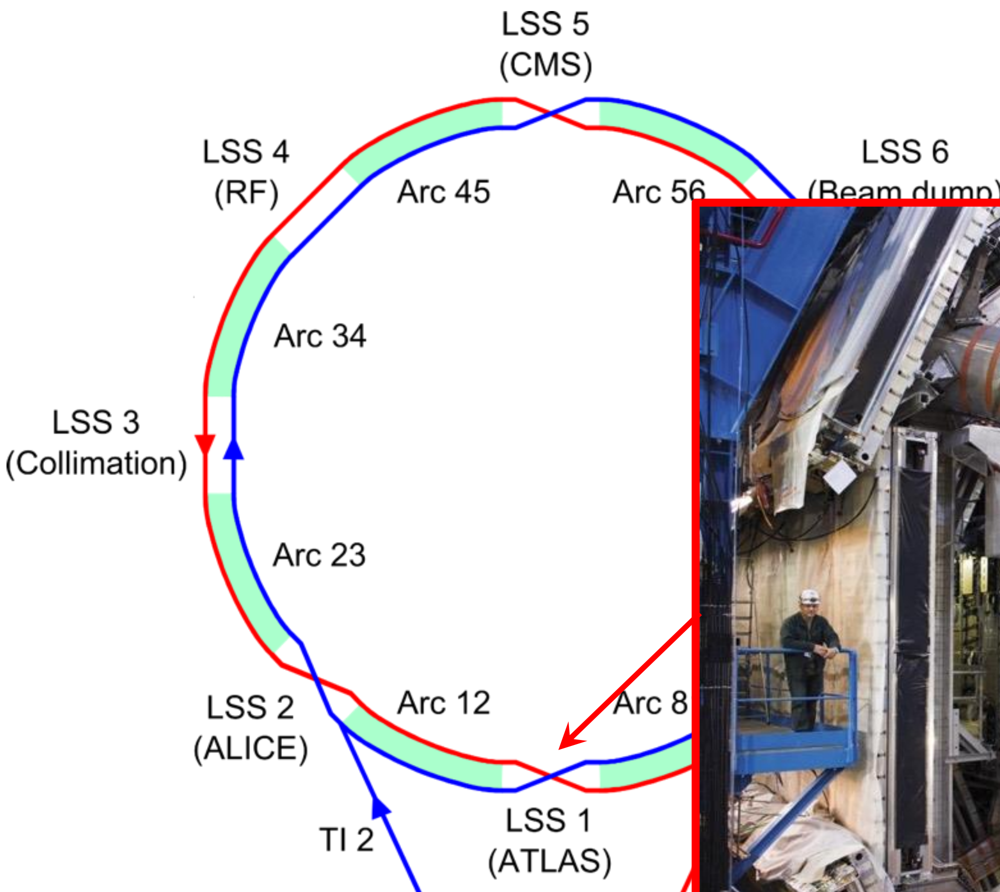
- store **450 GeV protons** (in two counter rotating beams)
- accelerate them up to **7 TeV**
- **collide the two beams** in four points of the ring (for high energy physics experiments)

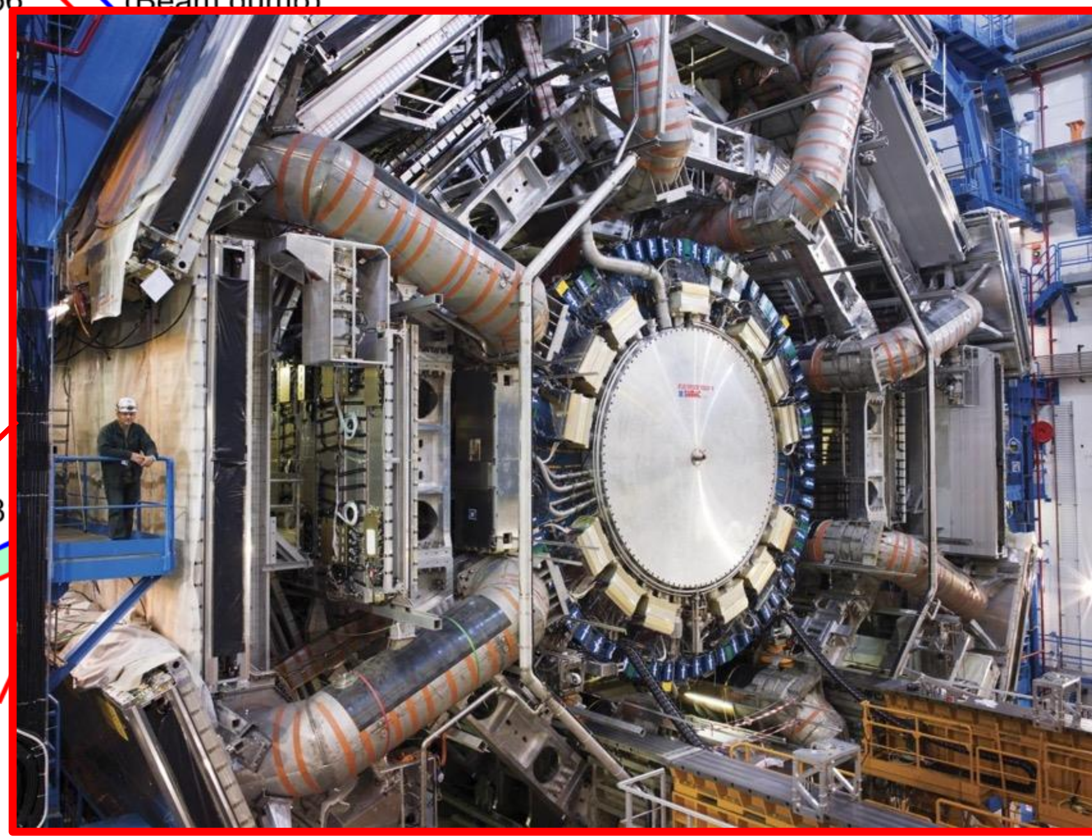


The Large Hadron Collider (LHC)

8-fold symmetric structure:

- **8 Long Straight Sections** (LSS) to host experiments and other equipment

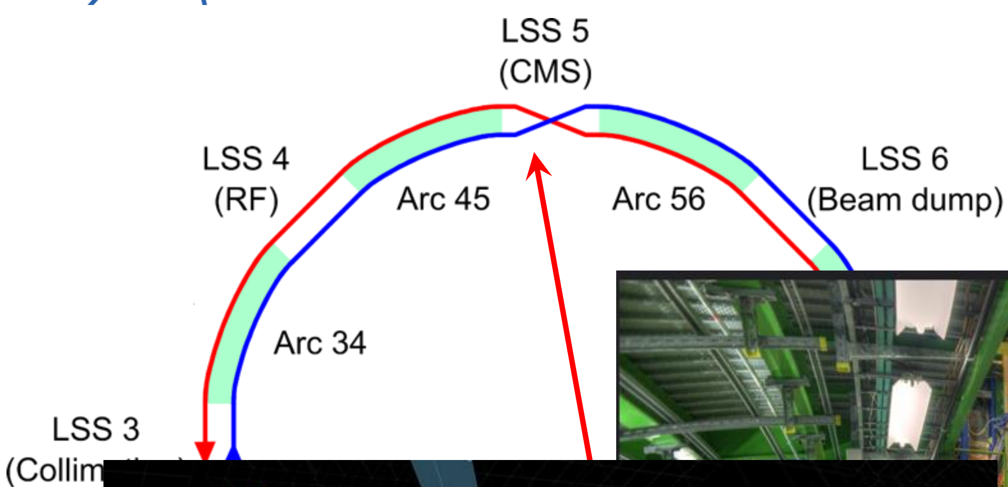


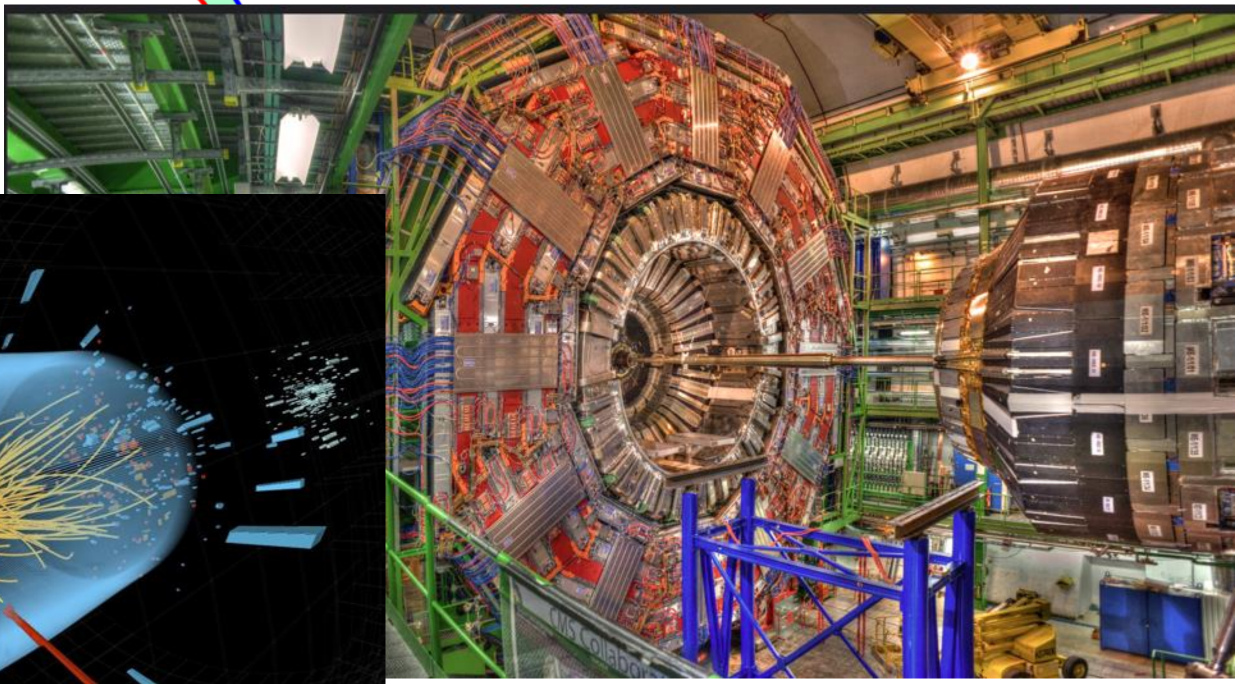


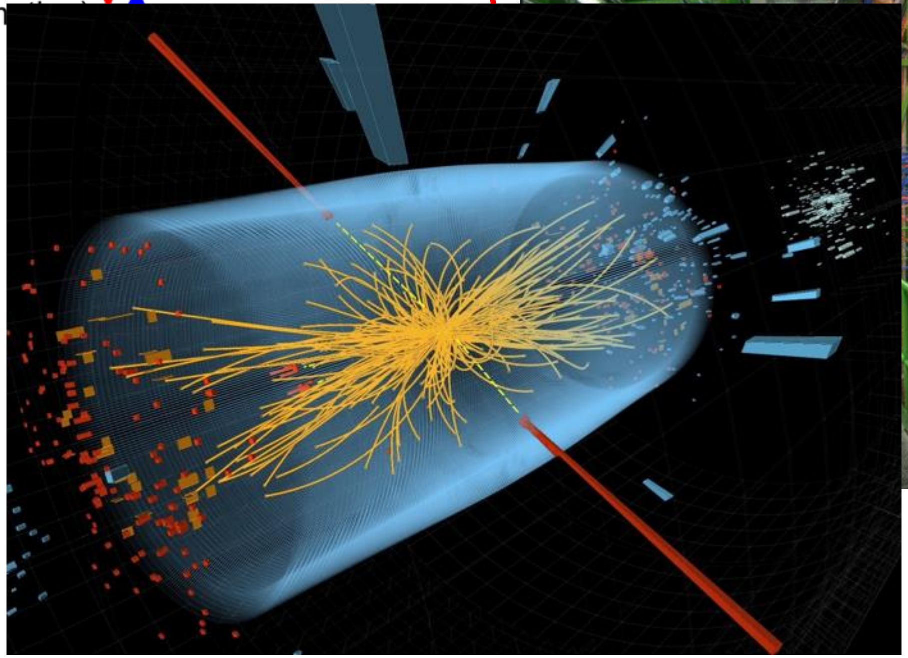
The Large Hadron Collider (LHC)

8-fold symmetric structure:

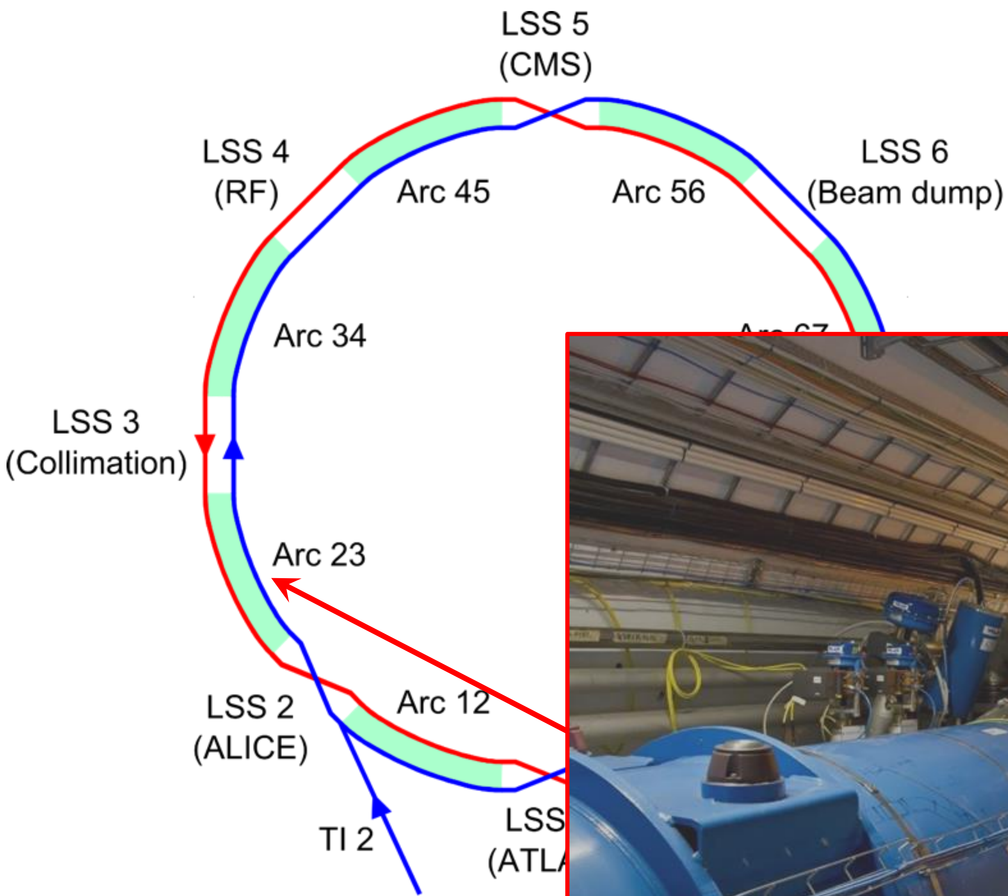
- **8 Long Straight Sections** (LSS) to host experiments and other equipment





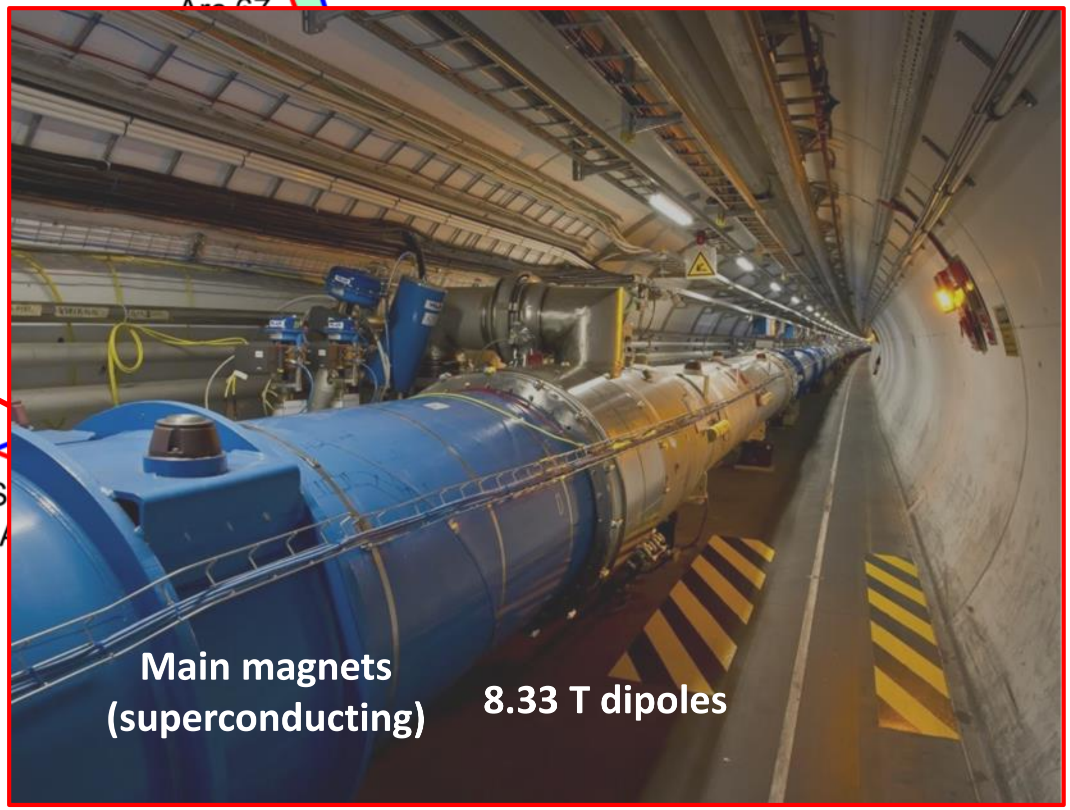


The Large Hadron Collider (LHC)



8-fold symmetric structure:

- **8 Long Straight Sections** (LSS) to host experiments and other equipment
- **8 Arcs** (2.45 km each - Periodic magnet lattice to bend and focus the beams)

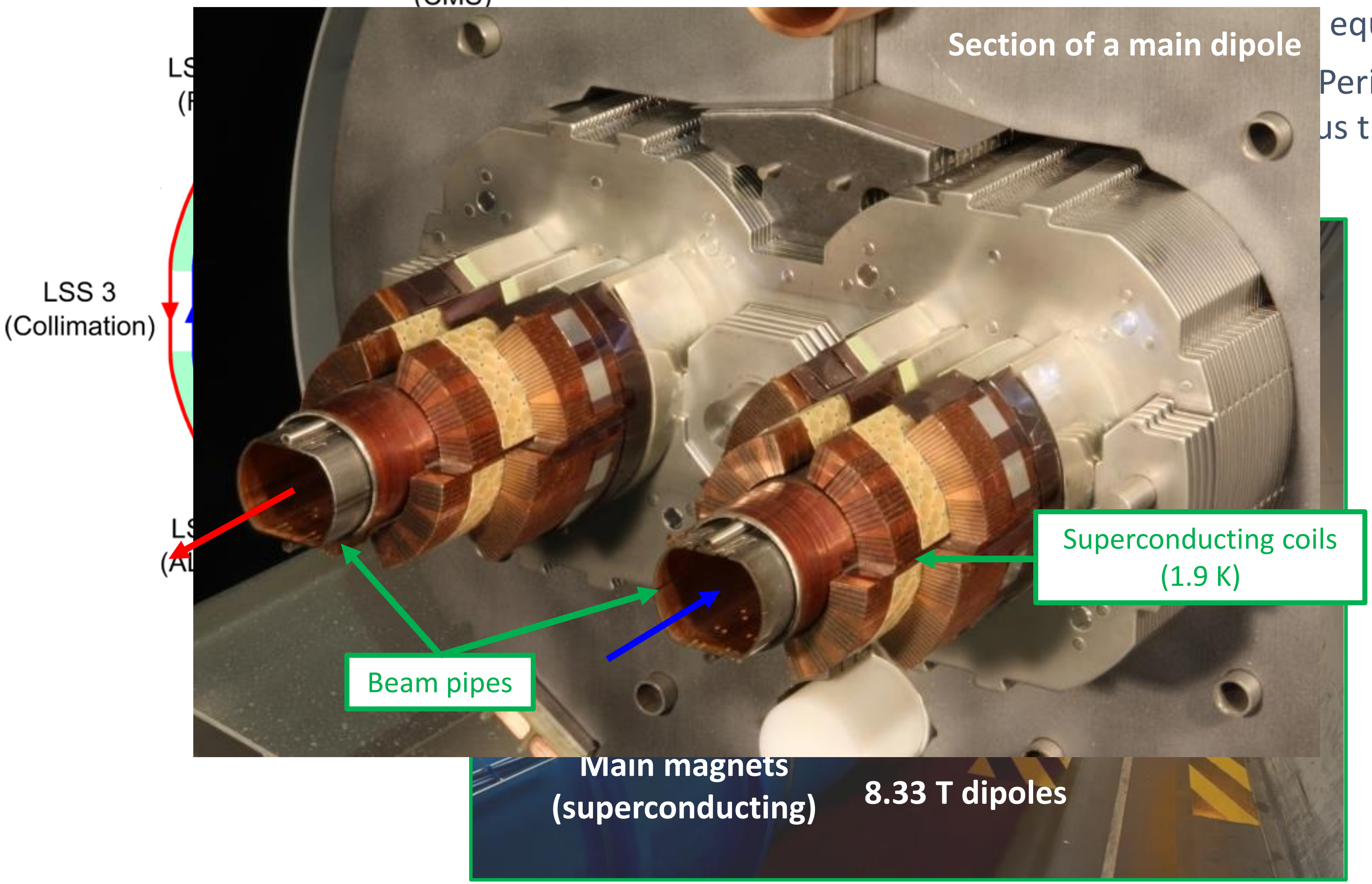


The Large Hadron Collider (LHC)

8-fold symmetric structure:

- **8 Long Straight Sections** (LSS) to host
  equipment

Periodic magnet
us the beams)



Outline

- **Particle accelerators**
  - Examples of applications
  - Working principles

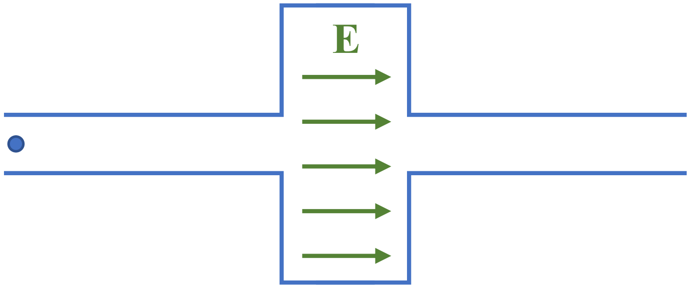
- **Beam optics calculations**
  - An example: the LHC betatron squeeze
  - Numerical optimization techniques

- **Particle tracking**
  - Motivations
  - Need for symplectic methods
  - Experience with GPU computing

Accelerating structures

- To **manipulate and accelerate charged particles** we use **electromagnetic fields**
  - The **Lorentz force** acts on the particles $\mathbf{F} = q\mathbf{E} + q\mathbf{v} \times \mathbf{B}$

- The **magnetic field** force **does not change the energy of the particles**
- The **acceleration** itself needs to be done by an **electric field** in specifically designed accelerating structures



Accelerating structure

Accelerating structures can be concatenated to form a **linear accelerator** ("linac"):

→ Acceleration is **very fast** (single passage)

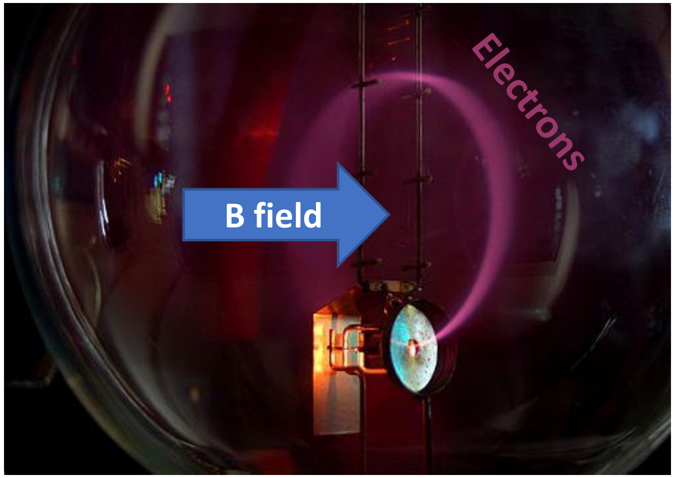
→ But **achievable energy is quite limited**

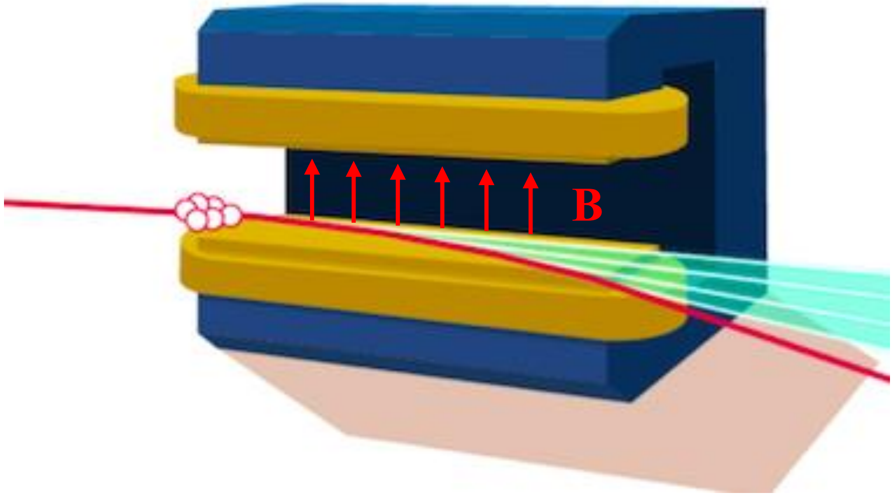
LINAC 4 at CERN

Bending magnets

- To **manipulate and accelerate charged particles** we use **electromagnetic fields**
  - The **Lorentz force** acts on the particles $\mathbf{F} = q\mathbf{E} + q\mathbf{v} \times \mathbf{B}$

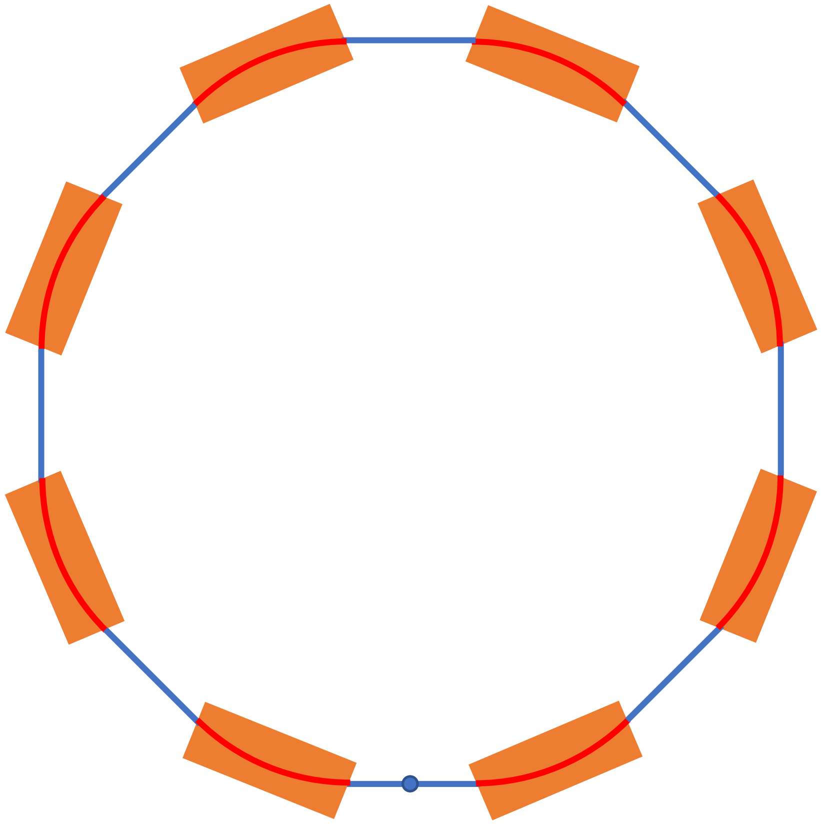
- **Magnetic fields** do not change the energy of the particles but **can be used very effectively to guide them**





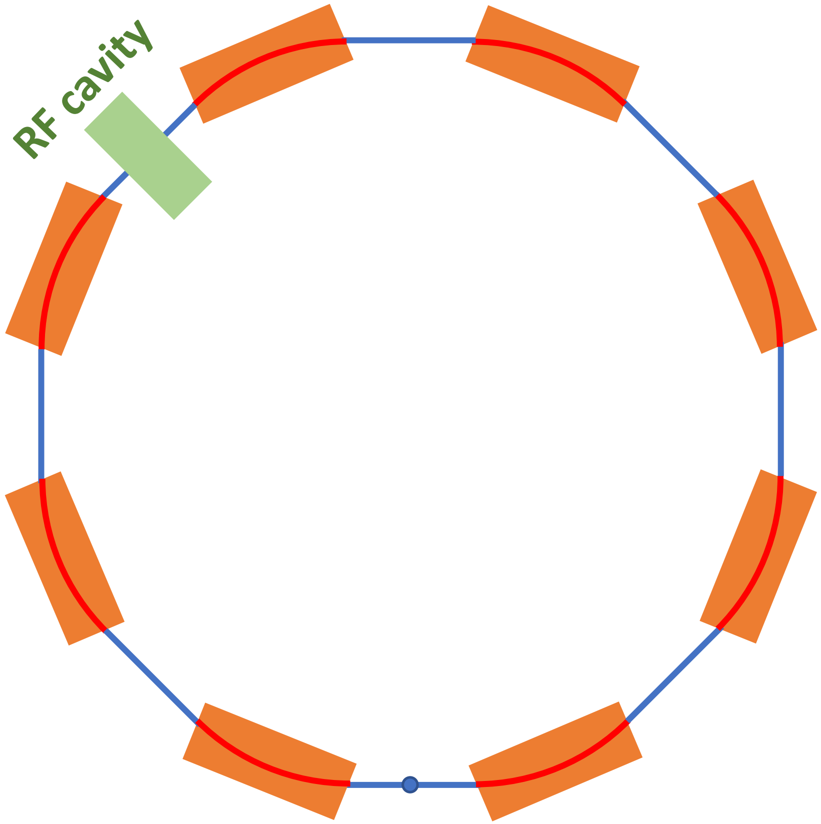
- **"Dipole magnets"** are used to **bend the particles' trajectory**

- We can use a set of **dipole magnets** to keep the particles on a **closed trajectory**



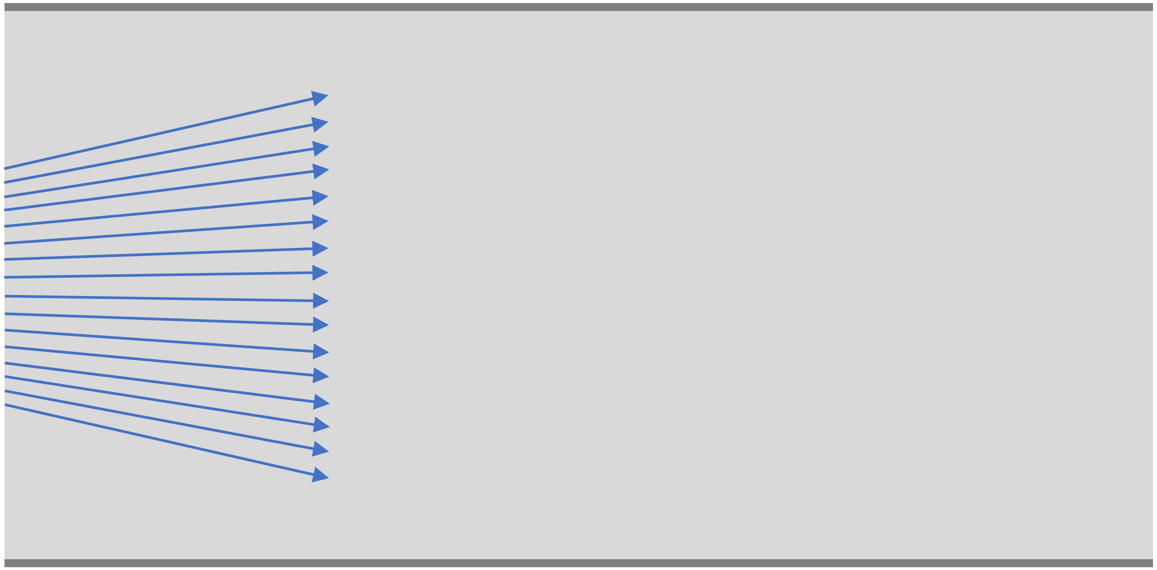
- We can use a set of **dipole magnets** to keep the particles on a **closed trajectory**
  - Allows to **accumulate "re-use" for many turns the energy gain from the same accelerating structure** (RF cavity)

- In the **Large Hadron Collider (LHC):**
  - We want to increase the proton energy by ~**6000 GeV**
  - Accelerating cavities provide only ~**500 keV/turn** (in average)
  - Acceleration is done in about ~**15million turns** (20 minutes)



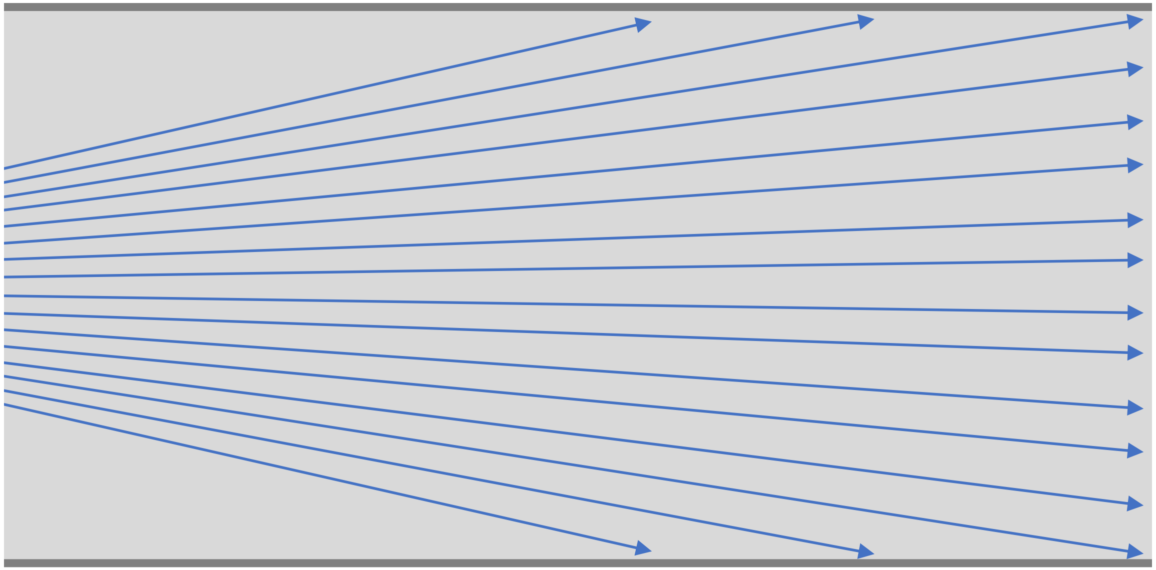
Need for focusing

- Due to the way they are produced, **particle beams always have a small divergence**



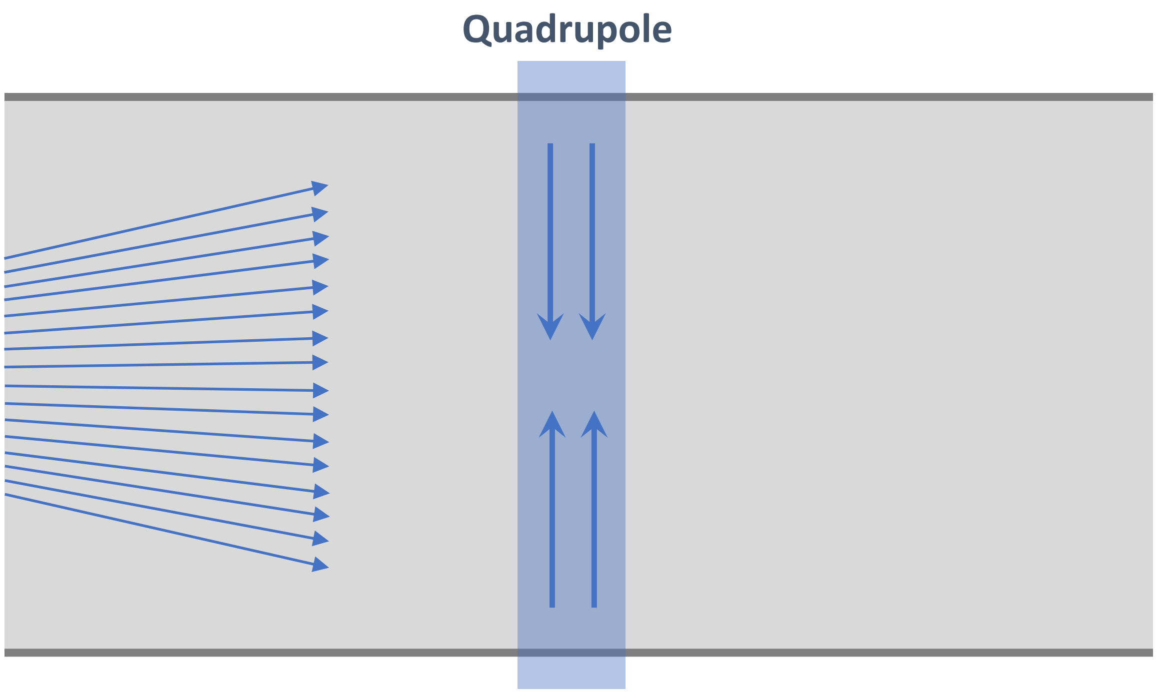
Need for focusing

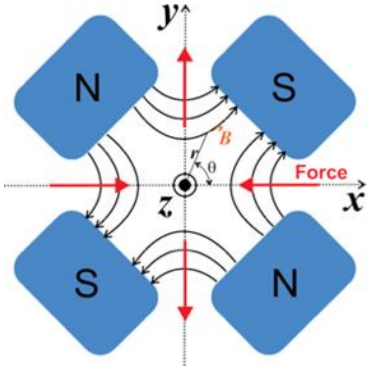
- Due to the way they are produced, **particle beams always have a small divergence**
  - Over time particles would be inevitably **lost on the walls of the beam-pipe**



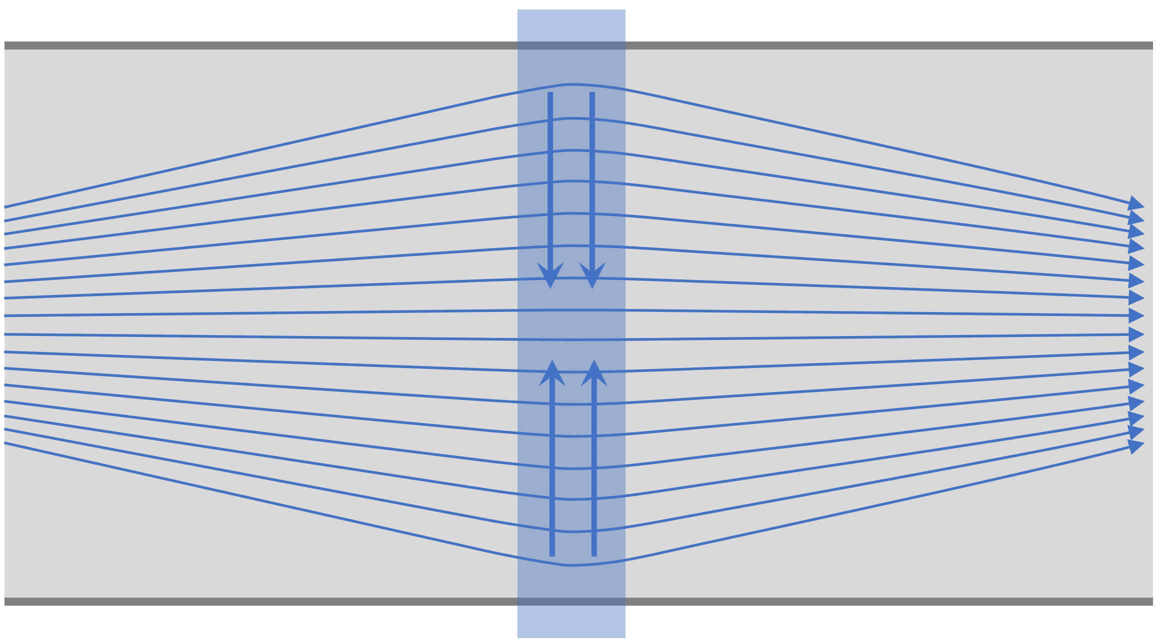
Quadrupole magnets

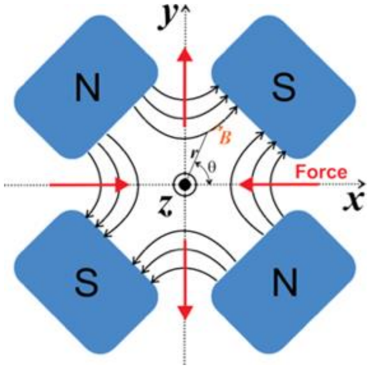
- Due to the way they are produced, **particle beams always have a small divergence**
  - Over time particles would be inevitably **lost on the walls of the beam-pipe**
- To keep the particles close to the center of the beam-pipe we use **quadrupole magnets**
  - In a quadrupole, the magnetic **force is linearly proportional to the distance from the center** of the magnet ($F_x = - k x$)





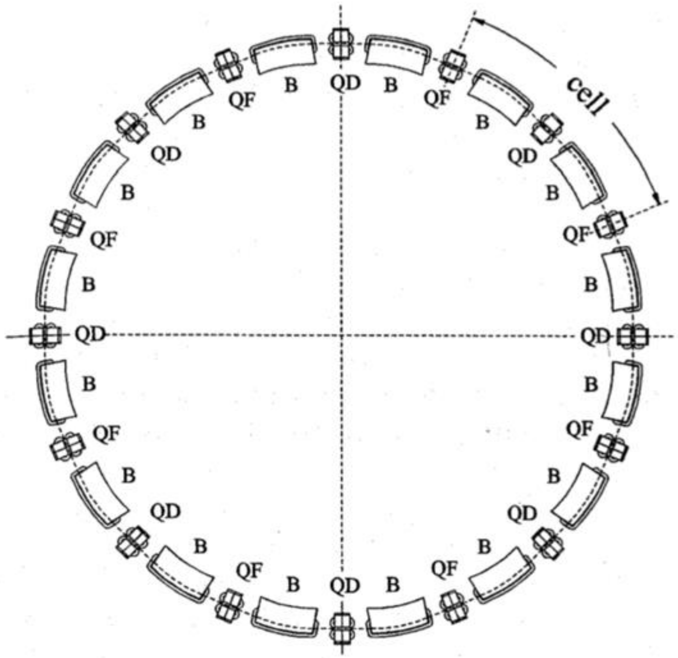
- Due to the way they are produced, **particle beams always have a small divergence**
  - Over time particles would be inevitably **lost on the walls of the beam-pipe**
- To keep the particles close to the center of the beam-pipe we use **quadrupole magnets**
  - In a quadrupole, the magnetic **force is linearly proportional to the distance from the center** of the magnet ($F_x = -k\,x$) → it acts like a **focusing lens**
  - Quadrupole magnets **focus the beam in one direction** but defocus on the orthogonal one

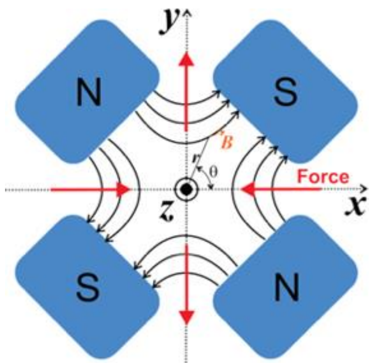




Quadrupole magnets

- Due to the way they are produced, **particle beams always have a small divergence**
  - Over time particles would be inevitably **lost on the walls of the beam-pipe**
- To keep the particles close to the center of the beam-pipe we use **quadrupole magnets**
  - In a quadrupole, the magnetic **force is linearly proportional to the distance from the center** of the magnet ($F_x = -k\,x$) → it acts like a **focusing lens**
  - Quadrupole magnets **focus the beam in one direction** but defocus on the orthogonal one → quadrupoles with opposite polarity are **alternated** to **confine the beams in both planes**





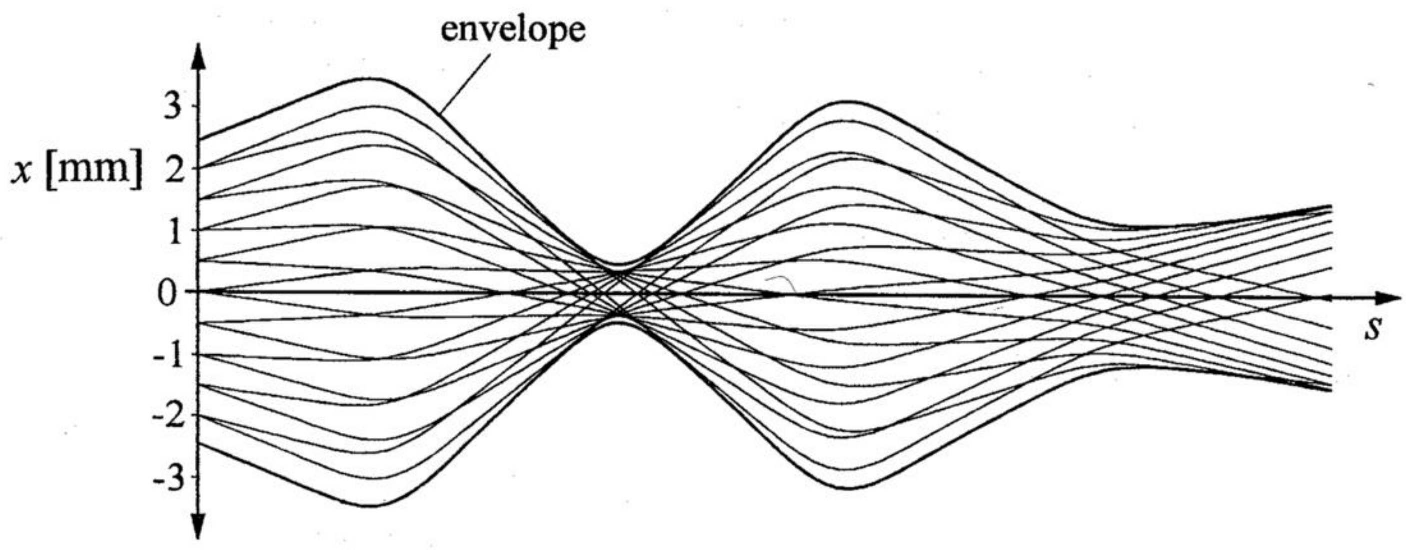
Outline

- **Particle accelerators**
  - Examples of applications
  - Working principles
- **Beam optics calculations**
  - An example: the LHC betatron squeeze
  - Numerical optimization techniques
- **Particle tracking**
  - Motivations
  - Need for symplectic methods
  - Experience with GPU computing

- In the presence of **dipole and quadrupole magnets alone** (linear regime) it is possible to **compute the envelope of the beam** without having to evaluate the trajectories of the single particles
  - Courant-Snyder formalism based on the Floquet theorem

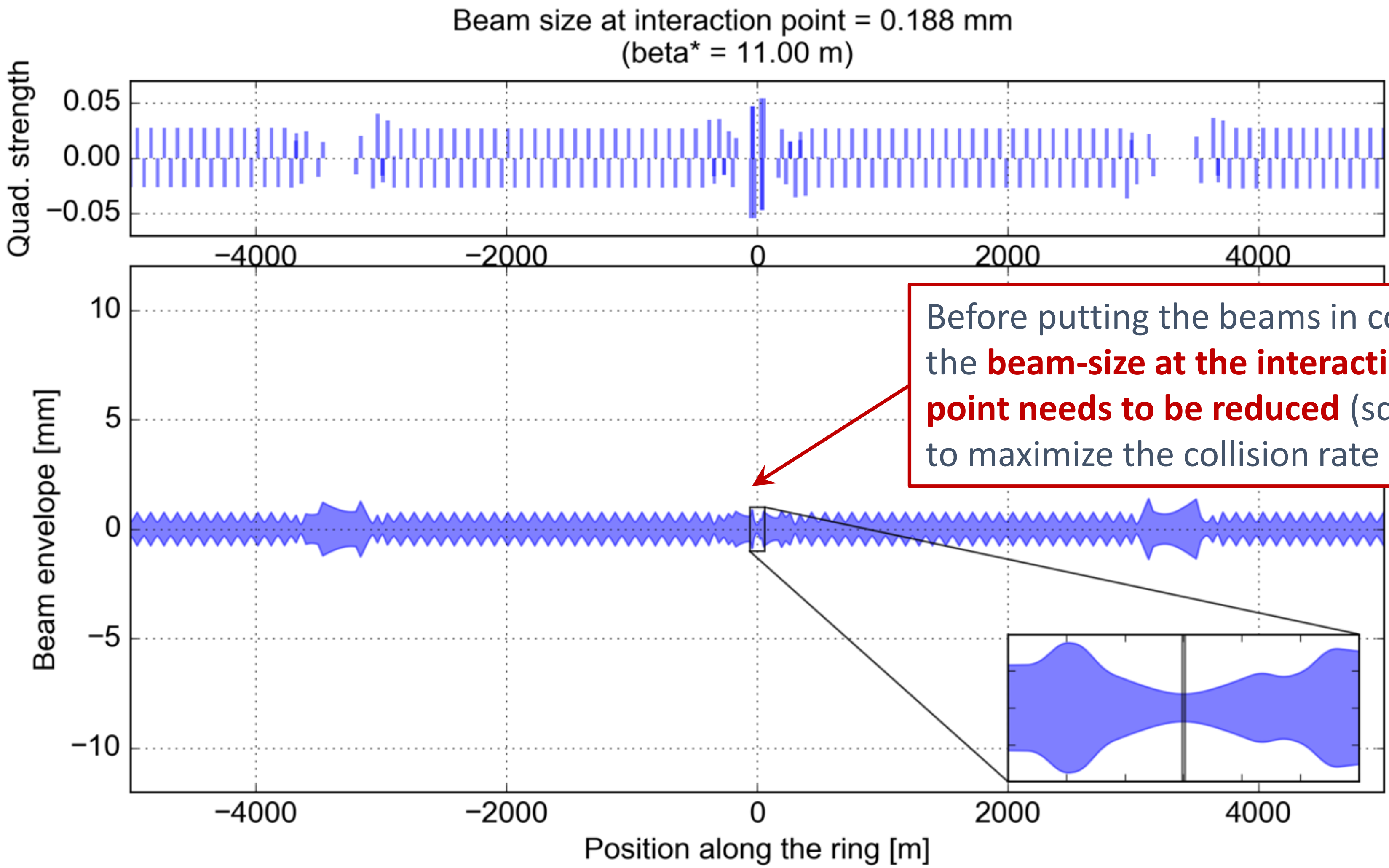
$$x(s) = \sqrt{\varepsilon}\sqrt{\beta(s)}\cos[\Psi(s) + \phi]$$

- These calculations are called in general **"linear optics"** calculations



- **Quadrupole strengths** can be **used to shape the particle beam envelope** (in the same way in which lenses can be used to shape a beam of light)

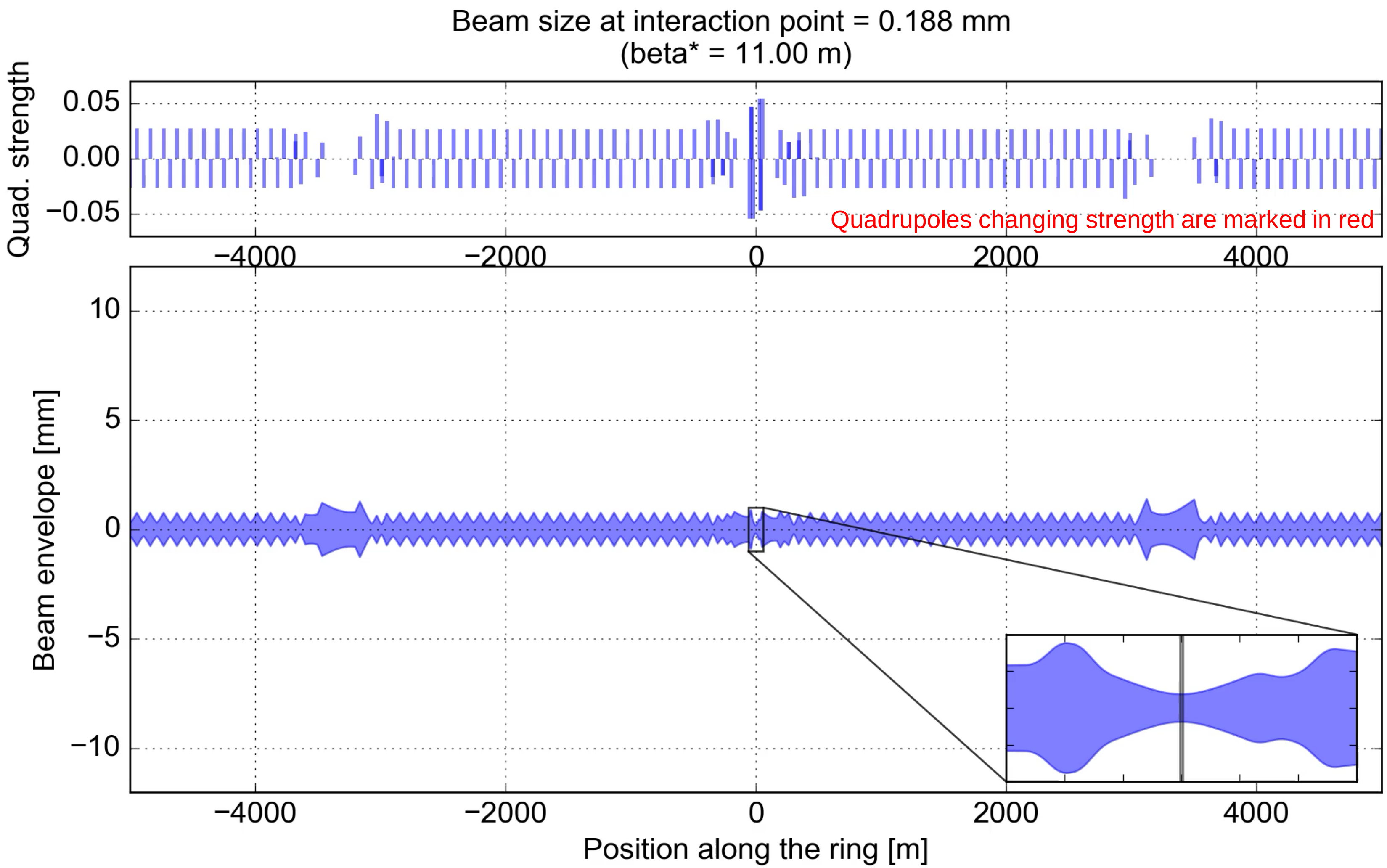
Example: LHC betatron squeeze



Before putting the beams in collisions, the **beam-size at the interaction point needs to be reduced** (squeezed) to maximize the collision rate

- **Quadrupole strengths** can be **used to shape the particle beam envelope** (in the same way in which lenses can be used to shape a beam of light)

Example: LHC betatron squeeze



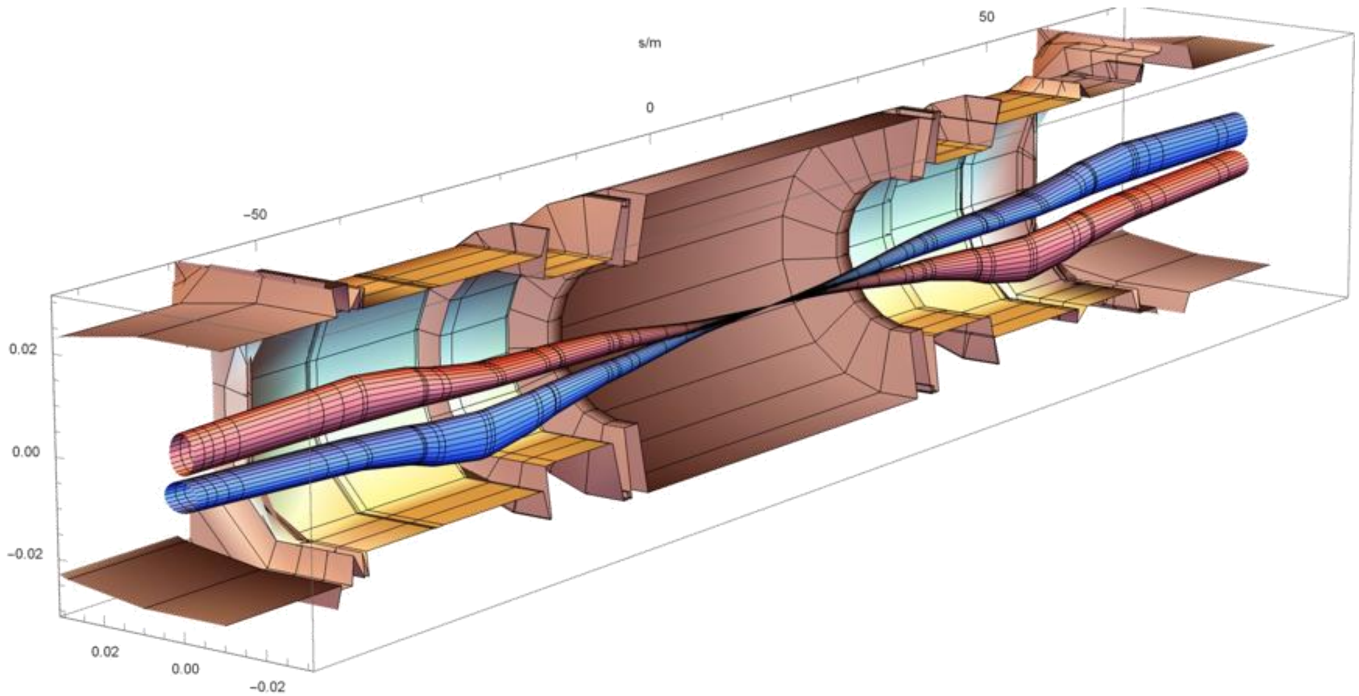
Outline

- **Particle accelerators**
  - Examples of applications
  - Working principles
- **Beam optics calculations**
  - An example: the LHC betatron squeeze
  - Numerical optimization techniques
- **Particle tracking**
  - Motivations
  - Need for symplectic methods
  - Experience with GPU computing

Shaping the "beam optics"

- The relation between the **quadrupole strengths and the beam size** is **non-trivial**
  - Several **technical constraints** also **need to be taken into account** (e.g. maximum quadrupole strength, current sign, size of the beam pipes, magnets in series that must have the same current)
  - **Numerical optimizers** need to be used to identify suitable quadrupole strengths as a function of given constraints on the beam envelope

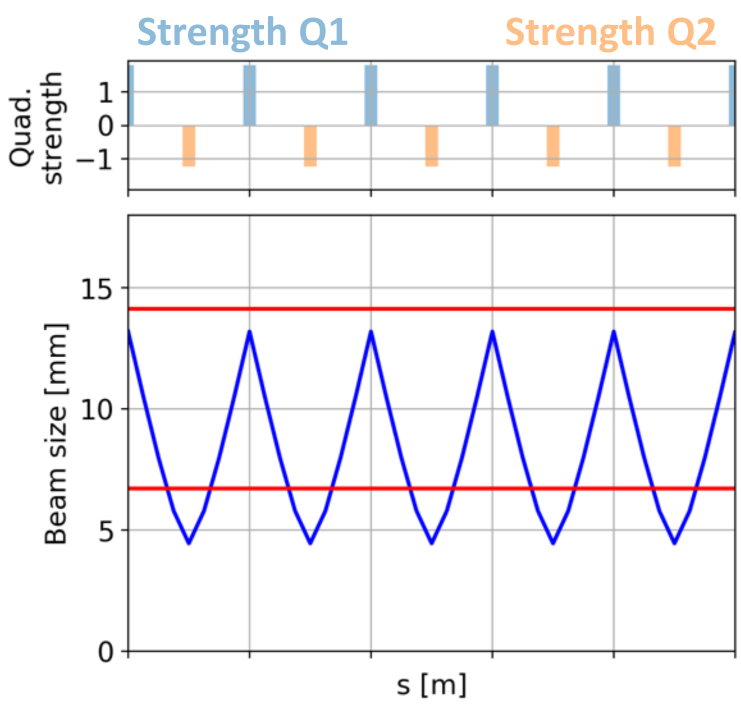
CERN's workhorse code for these calculations is the **MAD-X code**.



http://mad.web.cern.ch/mad/

Shaping the "beam optics"

- To illustrate how an optimization algorithm works we consider a **simple problem** with **2 constraints** and **2 degrees of freedom**:
  - We want the **maximum and the minimum of the beam envelope** σ(s) to assume specified values $\boldsymbol{\sigma_A}$ and $\boldsymbol{\sigma_B}$ (marked by the red lines in figure)
  - We can change the strength of **two families of quadrupoles** ($\mathbf{k_{Q1}}$ and $\mathbf{k_{Q2}}$)



We define a suitable **"cost function"**:

$$F\left(k_{\mathrm{Q1}}, k_{\mathrm{Q2}}\right) = \sqrt{\left(\frac{\sigma_{\mathrm{max}}\left(k_{\mathrm{Q1}}, k_{\mathrm{Q2}}\right) - \sigma_{\mathrm{A}}}{\sigma_{\mathrm{A}}}\right)^2 + \left(\frac{\sigma_{\mathrm{min}}\left(k_{\mathrm{Q1}}, k_{\mathrm{Q2}}\right) - \sigma_{\mathrm{B}}}{\sigma_{\mathrm{B}}}\right)^2}$$

To solve our problem **we need to search the minimum** of this quantity as function of k_{Q1} and k_{Q2}

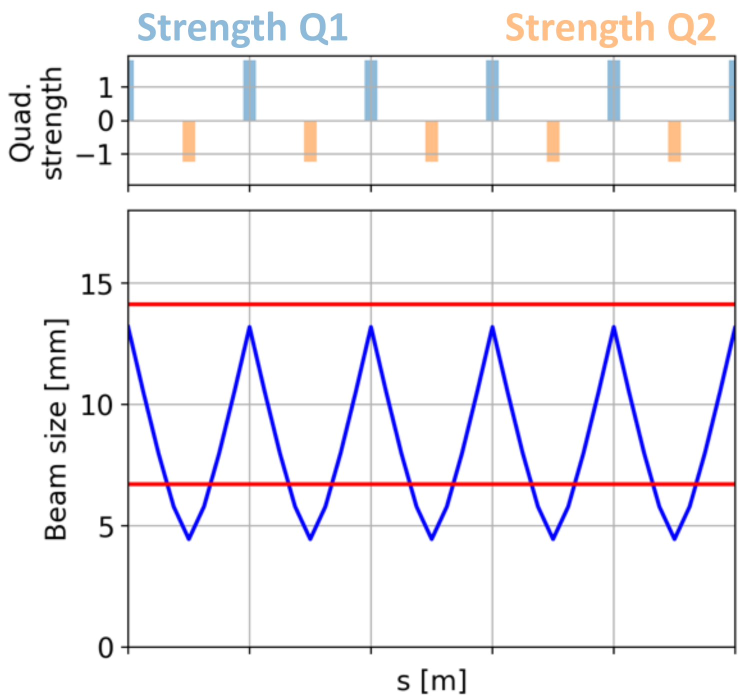
This is called an **"optimization problem"**

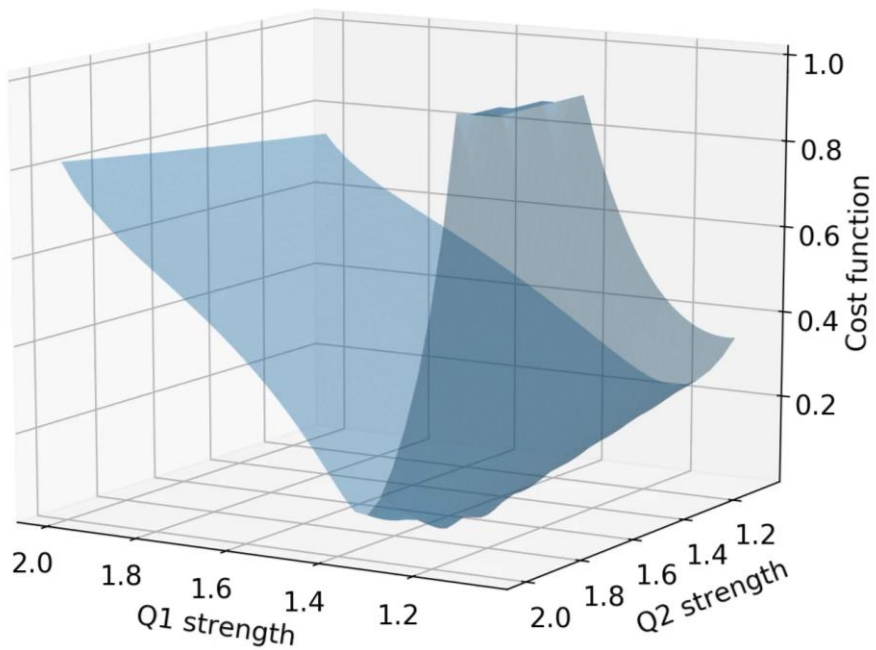
We define a suitable **"cost function"**:

$$F\left(k_{\mathrm{Q1}}, k_{\mathrm{Q2}}\right)=\sqrt{\left(\frac{\sigma_{\max }\left(k_{\mathrm{Q1}}, k_{\mathrm{Q2}}\right)-\sigma_{\mathrm{A}}}{\sigma_{\mathrm{A}}}\right)^2+\left(\frac{\sigma_{\min }\left(k_{\mathrm{Q1}}, k_{\mathrm{Q2}}\right)-\sigma_{\mathrm{B}}}{\sigma_{\mathrm{B}}}\right)^2}$$

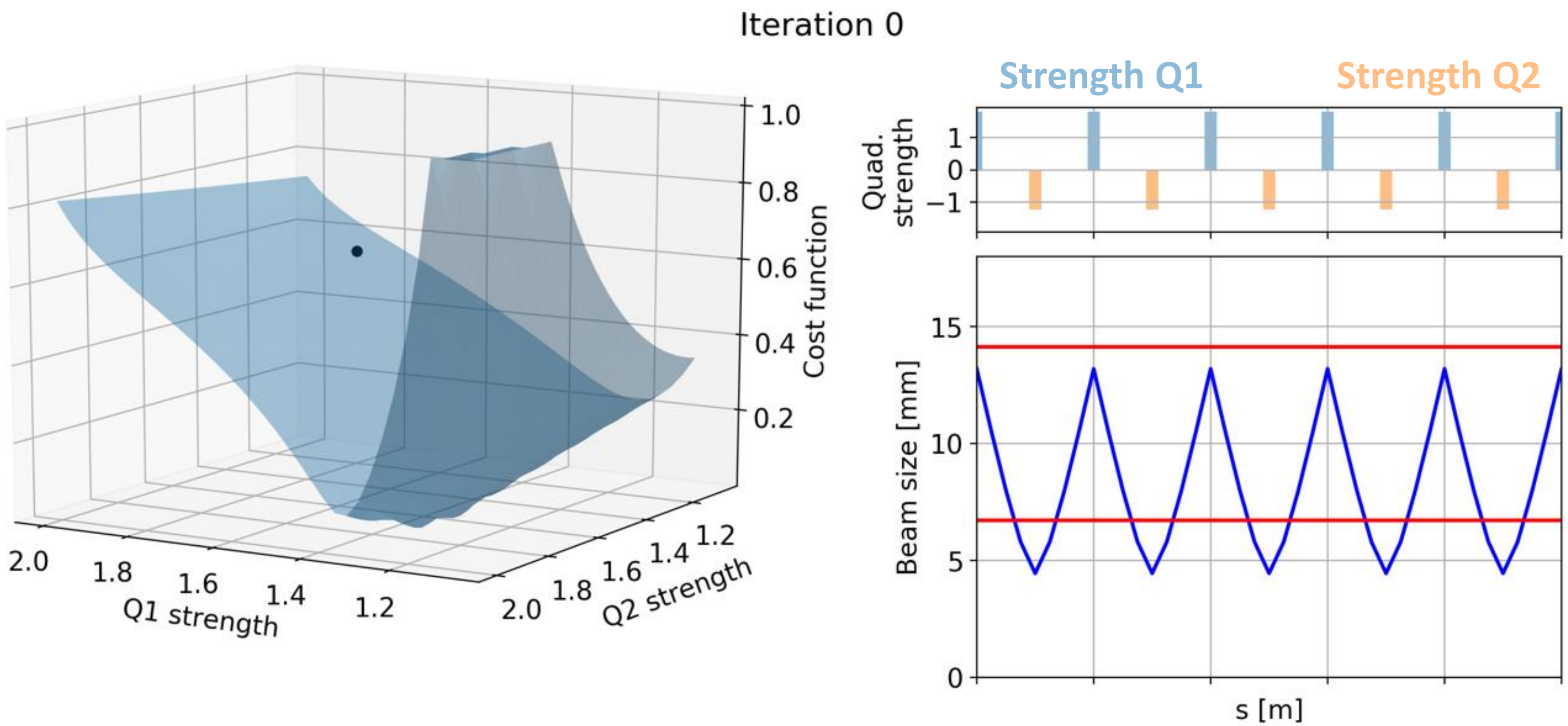
To solve our problem **we need to search the minimum** of this quantity as function of k_{Q1} and k_{Q2}

- With only two degrees of freedom we **can visualize the function as surface**
- With more than two degrees of freedom it can become **too expensive to map the whole parameter space** → we need to **search for the minimum "blindly"**





A simple **optimization technique** is the **"gradient method". It is based on the following iteration** (starting from an arbitrary "guess" of the solution) :

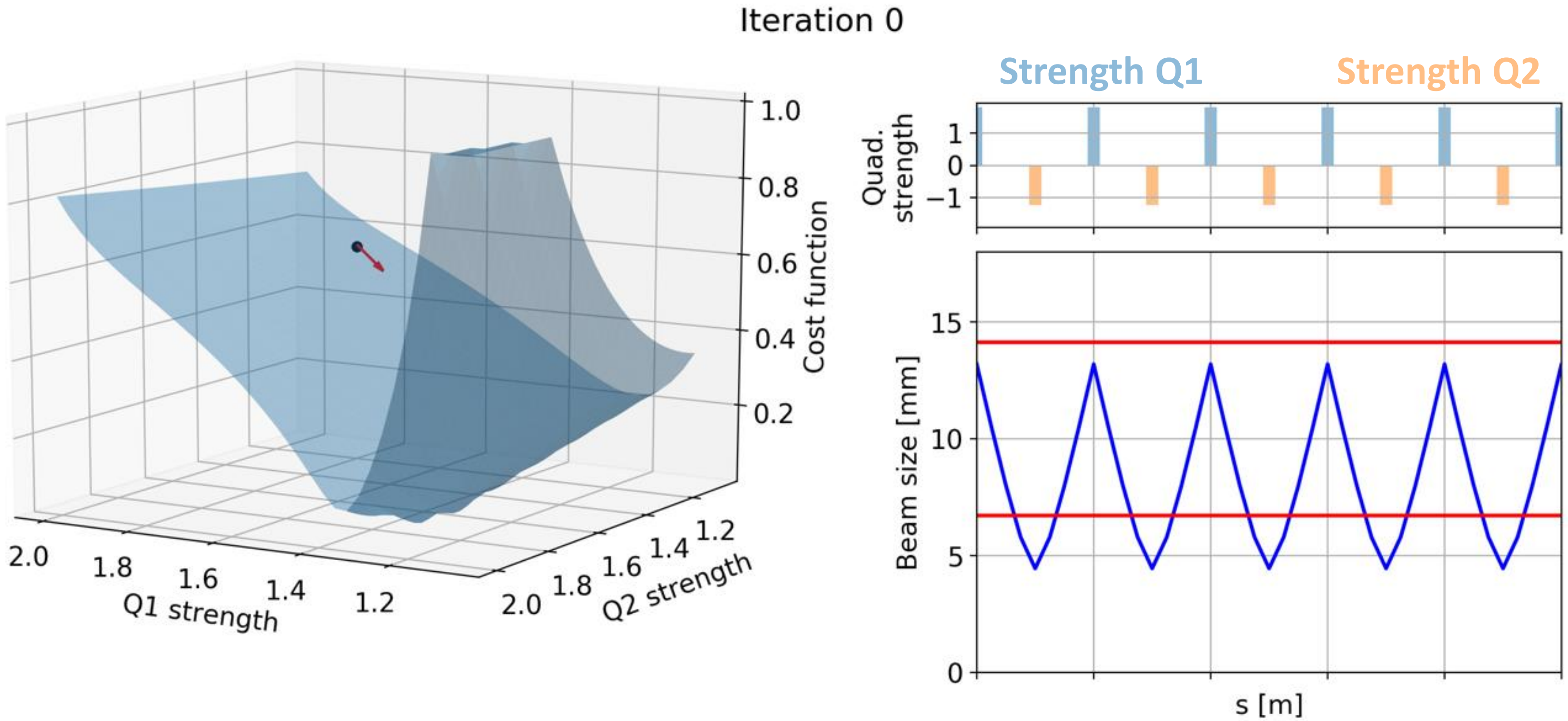


Gradient method

A simple **optimization technique** is the **"gradient method". It is based on the following iteration** (starting from an arbitrary "guess" of the solution) :

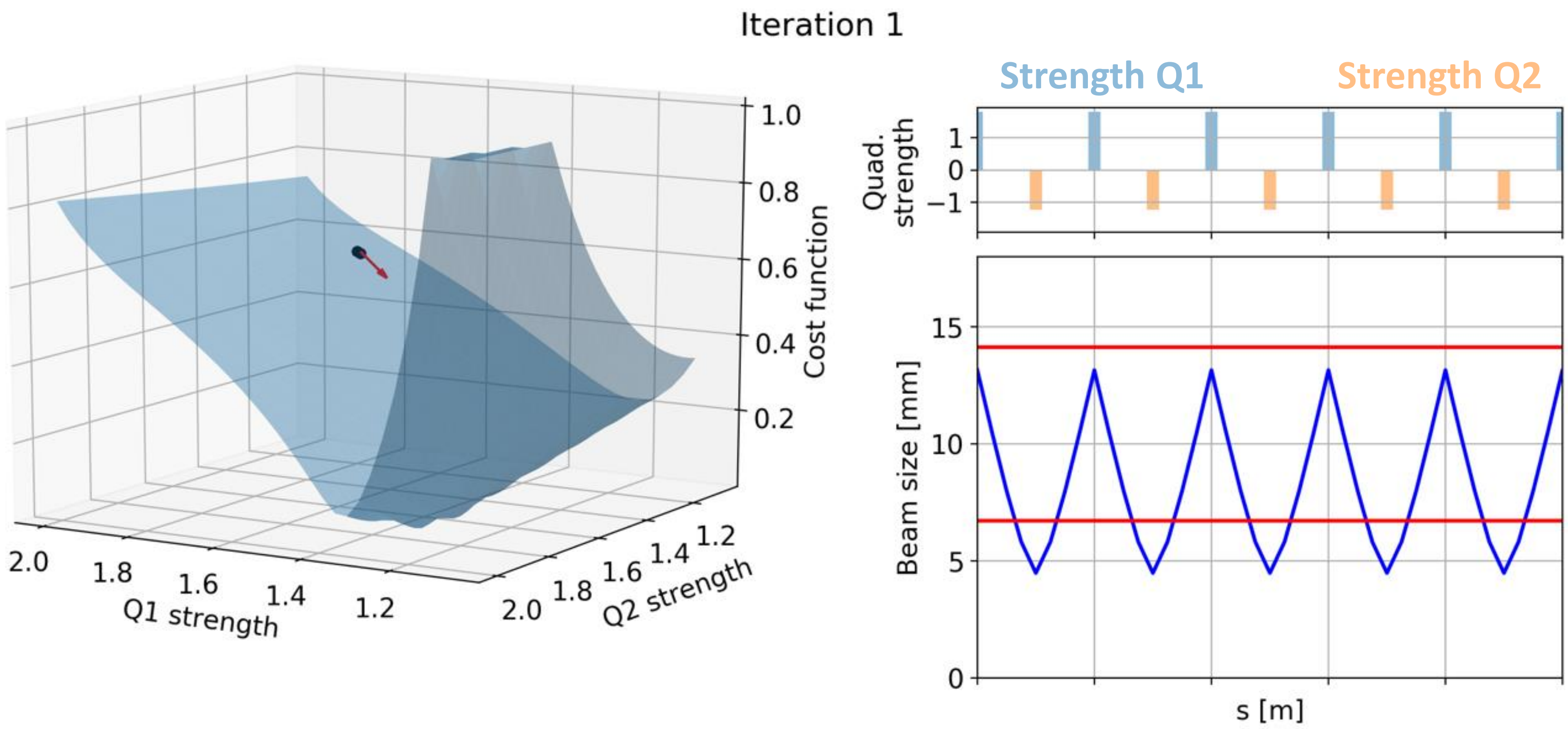
1. At the given point we evaluate the **gradient of the cost function**, tells us the **direction in which our surface is the steepest**

$$-\nabla F = -\left(\frac{\partial F}{\partial k_{\mathrm{Q1}}}, \frac{\partial F}{\partial k_{\mathrm{Q2}}}\right)$$



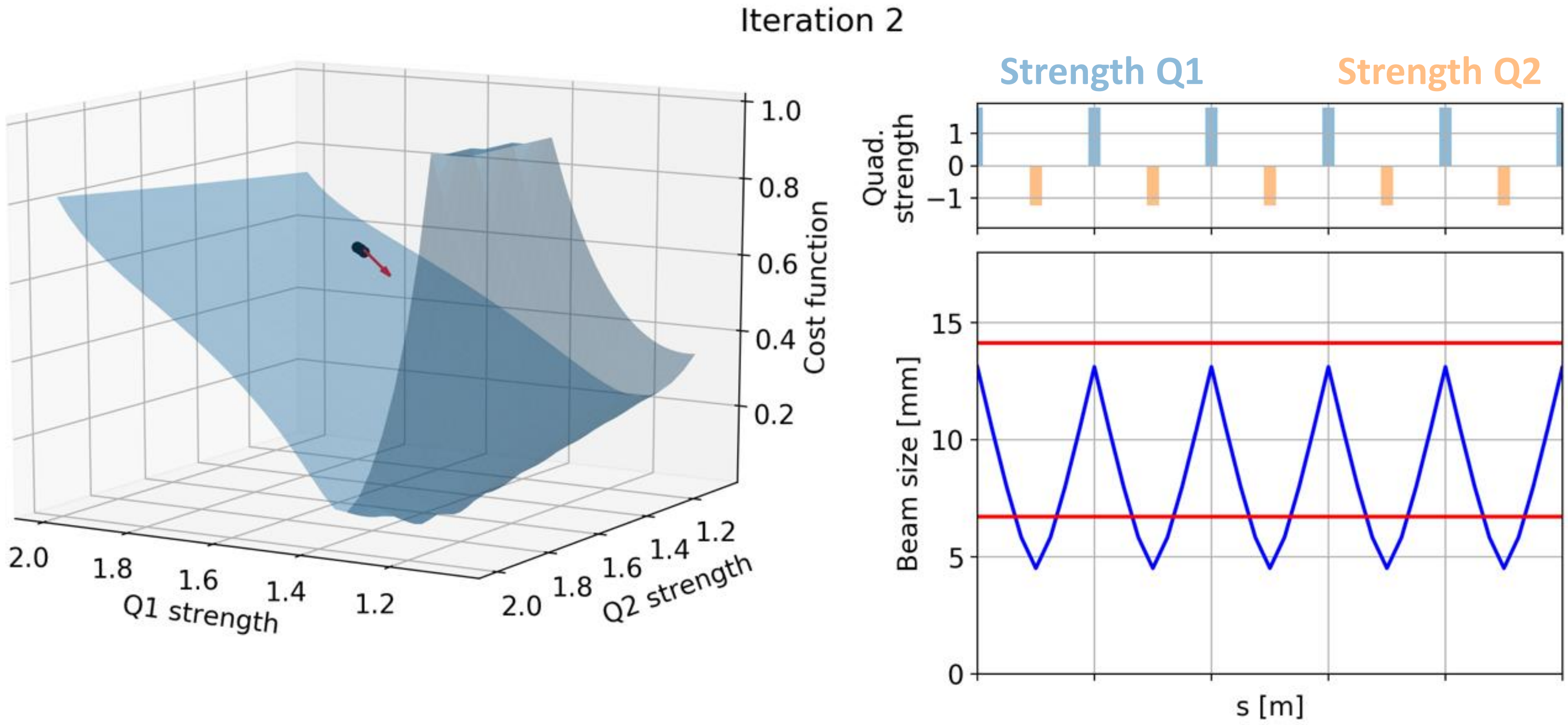
A simple **optimization technique** is the **"gradient method". It is based on the following iteration** (starting from an arbitrary "guess" of the solution) :

1. At the given point we evaluate the **gradient of the cost function**, tells us the **direction in which our surface is the steepest** $$-\nabla F = -\left(\frac{\partial F}{\partial k_{Q1}}, \frac{\partial F}{\partial k_{Q2}}\right)$$
2. We take a **new point in that direction** and we go back to 1.



A simple **optimization technique** is the **"gradient method". It is based on the following iteration** (starting from an arbitrary "guess" of the solution) :

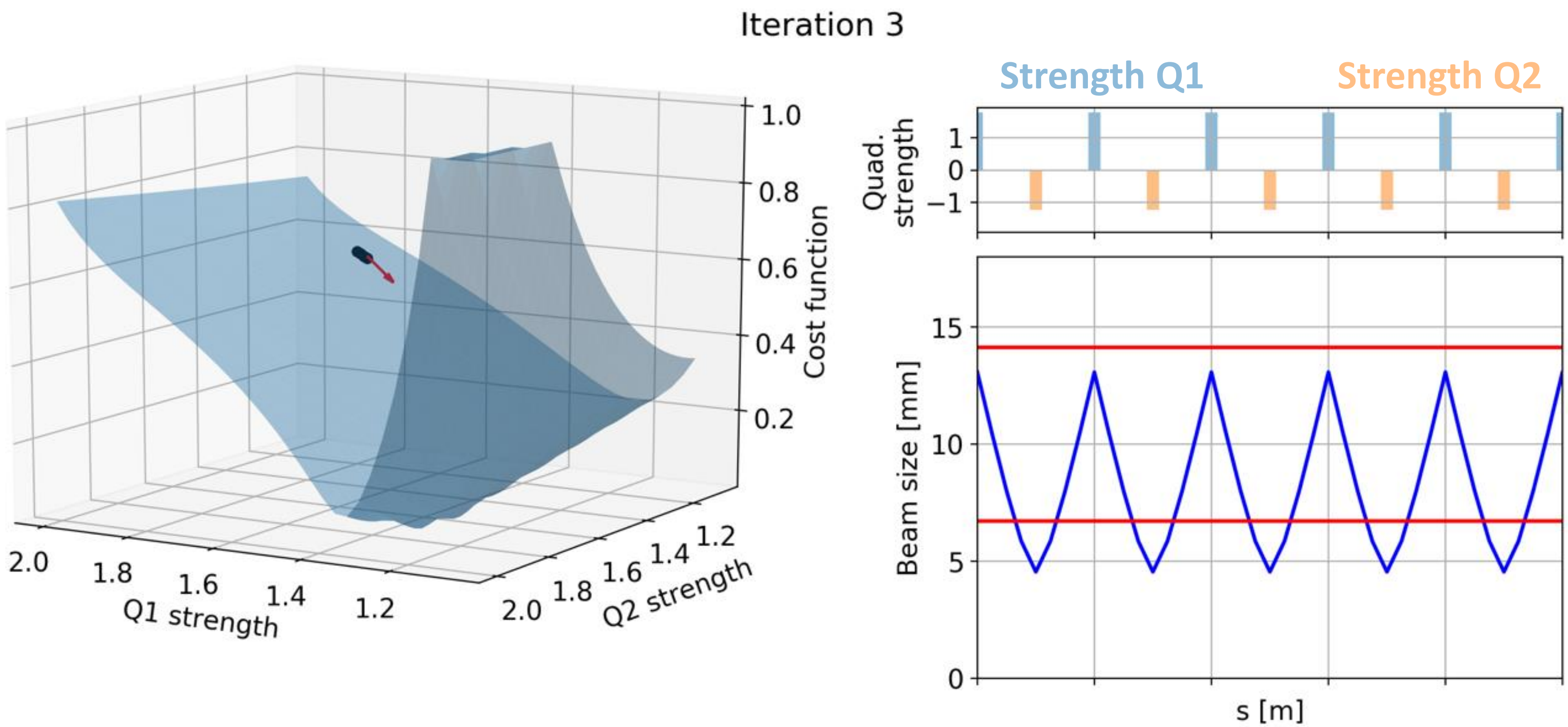
1. At the given point we evaluate the **gradient of the cost function**, tells us the **direction in which our surface is the steepest**
$$-\nabla F = -\left(\frac{\partial F}{\partial k_{Q1}}, \frac{\partial F}{\partial k_{Q2}}\right)$$
2. We take a **new point in that direction** and we go back to 1.



A simple **optimization technique** is the **"gradient method". It is based on the following iteration** (starting from an arbitrary "guess" of the solution) :

1. At the given point we evaluate the **gradient of the cost function**, tells us the **direction in which our surface is the steepest**
$$-\nabla F = -\left(\frac{\partial F}{\partial k_{Q1}}, \frac{\partial F}{\partial k_{Q2}}\right)$$
2. We take a **new point in that direction** and we go back to 1.

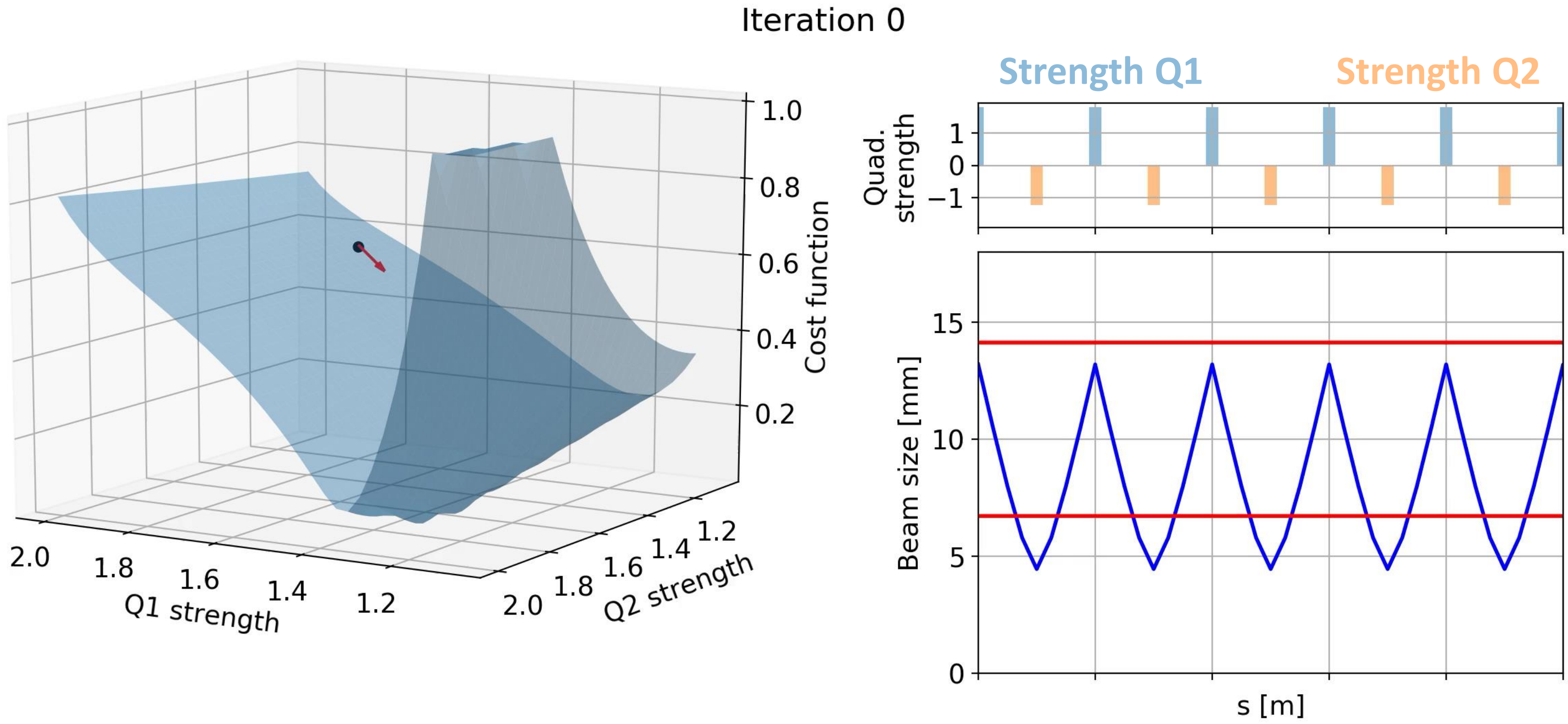
After a certain number of iterations the algorithm will **converge to a minimum of the cost function**



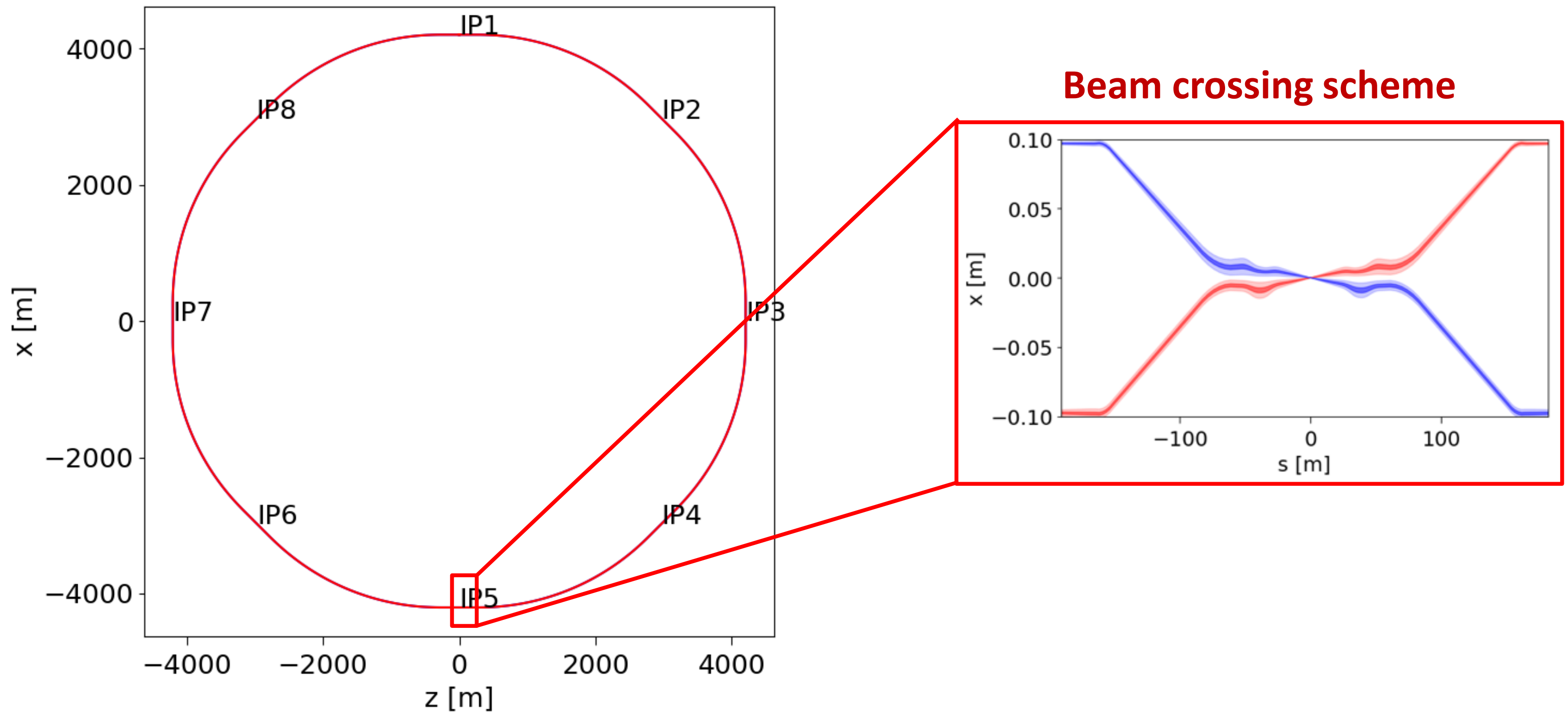
A simple **optimization technique** is the **"gradient method". It is based on the following iteration** (starting from an arbitrary "guess" of the solution) :

1. At the given point we evaluate the **gradient of the cost function**, tells us the **direction in which our surface is the steepest**
$$-\nabla F = -\left(\frac{\partial F}{\partial k_{Q1}}, \frac{\partial F}{\partial k_{Q2}}\right)$$
2. We take a **new point in that direction** and we go back to 1.

After a certain number of iterations the algorithm will **converge to a minimum of the cost function**

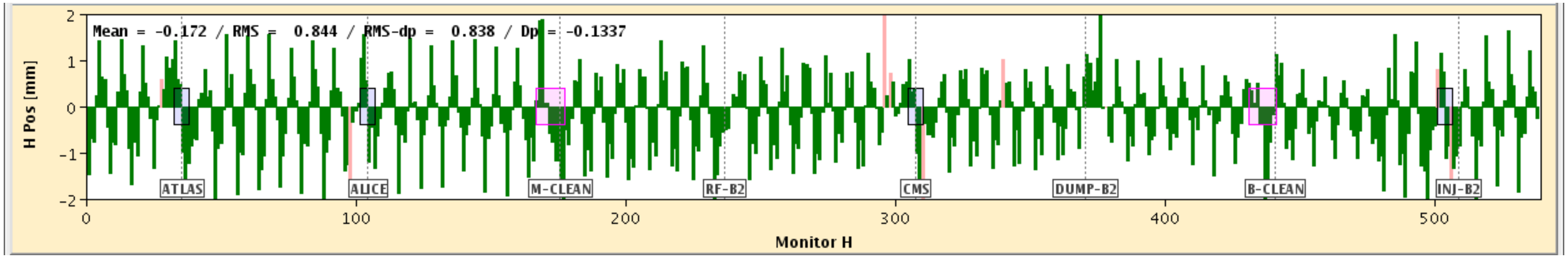


Similar techniques are user to **shape the beam trajectories** (closed orbit)

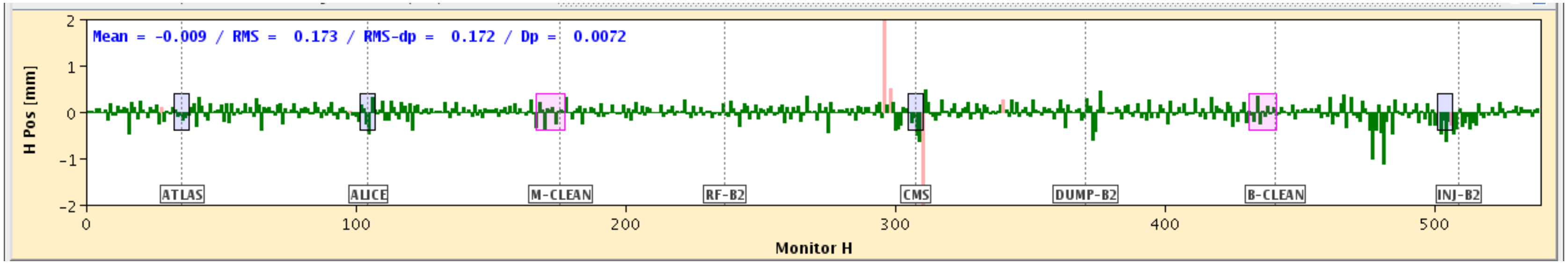


Iterative methods are used also **to correct the beam trajectory** (with respect to a known reference) due to daily small fluctuations → done **online on the circulating beams**

Before correction



After correction



Outline

- **Particle accelerators**
  - Examples of applications
  - Working principles

- **Beam optics calculations**
  - An example: the LHC betatron squeeze
  - Numerical optimization techniques

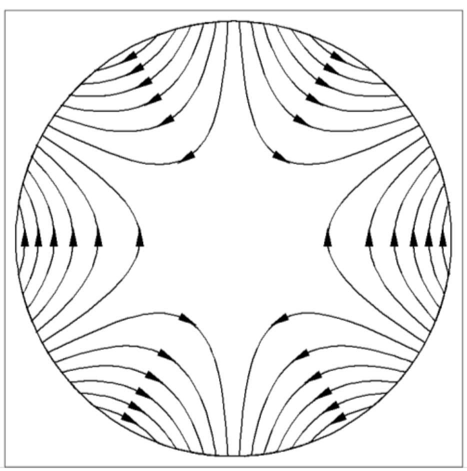
- **Particle tracking**
  - Motivations
  - Need for symplectic methods
  - Experience with GPU computing

Non-linearities

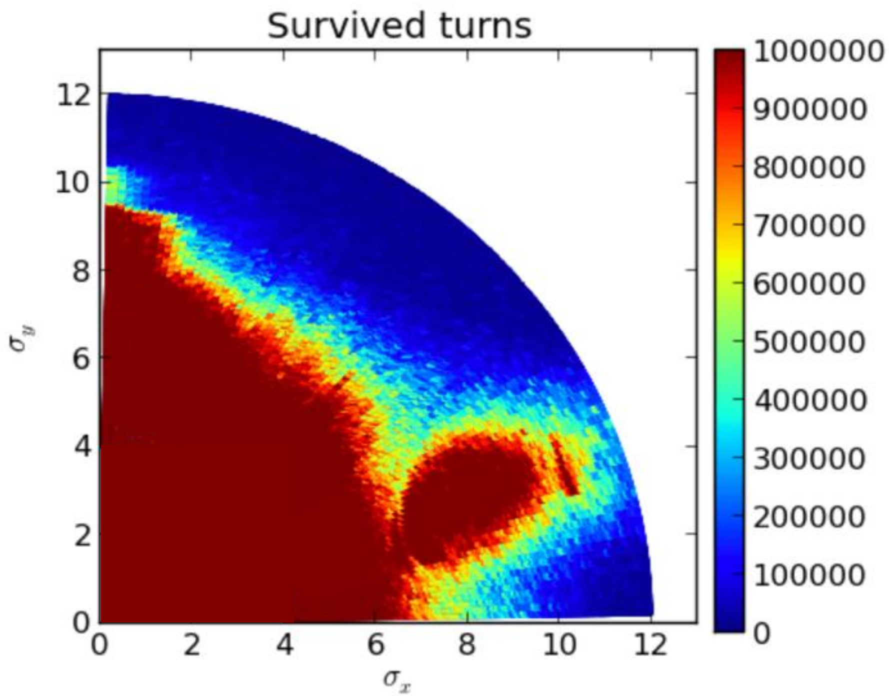
- **Dipolar and quadrupolar fields** are in principle **sufficient to keep the particles on a closed trajectory and keep them focused**.
- Nevertheless in a **realistic accelerator** the **situation is more complex**:
  - **Magnets are not perfect** (dipole and quadrupole magnets have unwanted deviation from the ideal field shapes)
  - Magnets **are not "exactly" where they are supposed to be** (alignment errors)
  - Particles **do not have all exactly the same energy** (typical relative spread ~10^{-3})
    - Need of **"chromatic corrections" using sextupole magnets**

→ A realistic machine has **unavoidable non-linearities**

Sextupole magnet



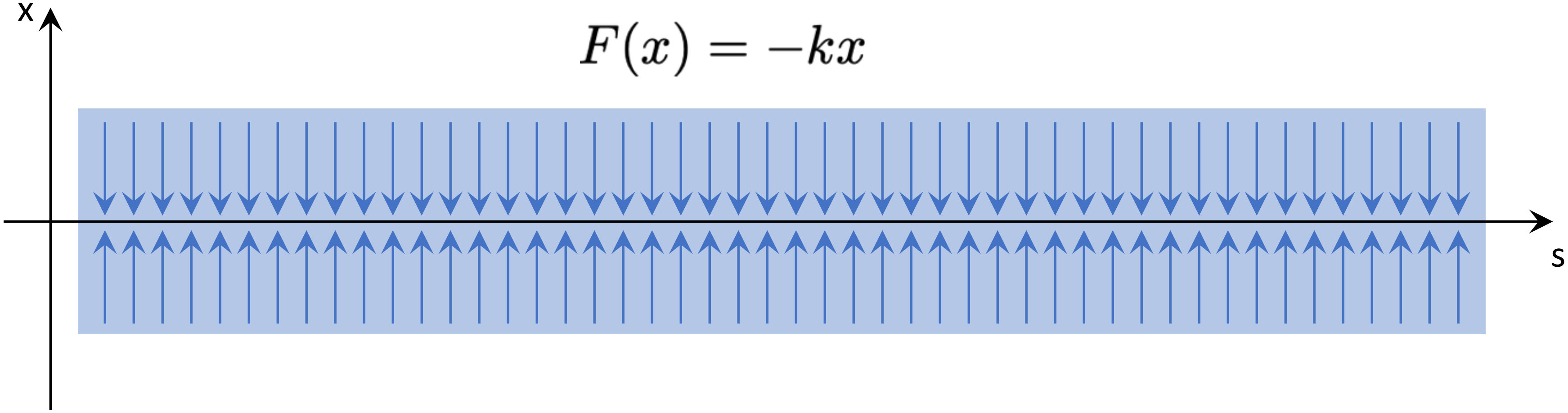
- In the presence of these effects, **the particle motion gets much more complex**:
  - The envelope equation is not anymore enough
  - Depending on the initial conditions **particles can be lost after a certain number of turns**
- We need to **numerically simulate the motion of the particle** in the accelerator:
  - We are interested in **quantifying how many particles will be lost over a realistic time** → for the LHC we **need to simulate ~millions of turns**!



- **Particle accelerators**
  - Examples of applications
  - Working principles
- **Beam optics calculations**
  - An example: the LHC betatron squeeze
  - Numerical optimization techniques
- **Particle tracking**
  - Motivations
  - Need for symplectic methods
  - Experience with GPU computing

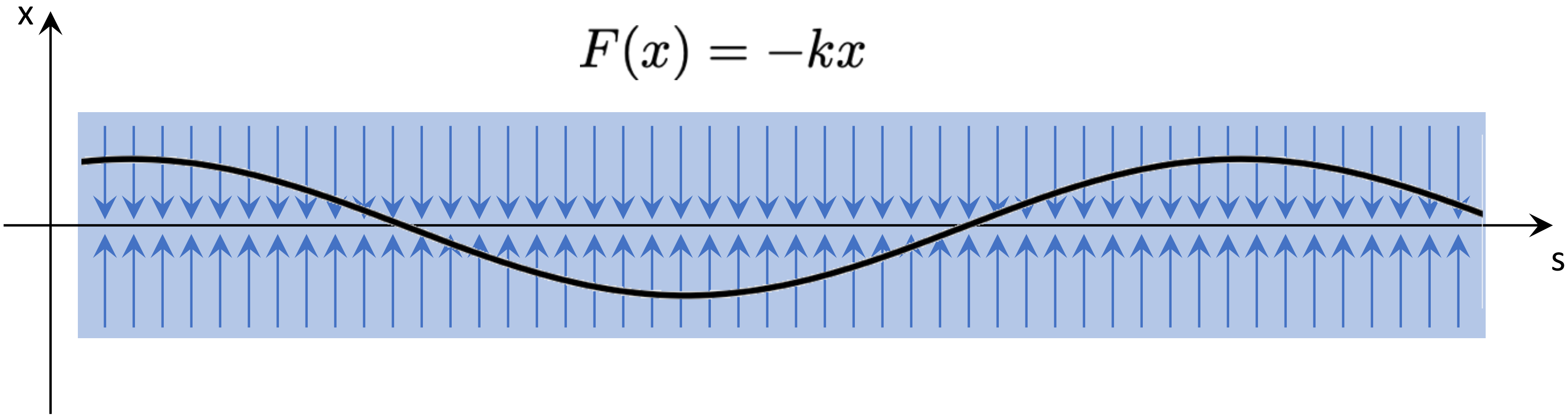
- Simulations on such long time scales are subject to **particular issues**, which we will illustrate using a **simple example**:
  - We assume **uniform focusing force**

$$F(x) = -kx$$



- Simulations on such long time scales are subject to **particular issues**, which we will illustrate using a **simple example**:
  - We assume **uniform focusing force**
  - In such a field the **particle oscillate** around the axis x = 0

$$F(x) = -kx$$



Equations of motion

$$F = ma \quad \Rightarrow \quad \frac{dv_x}{dt} = -\frac{k}{m}x$$

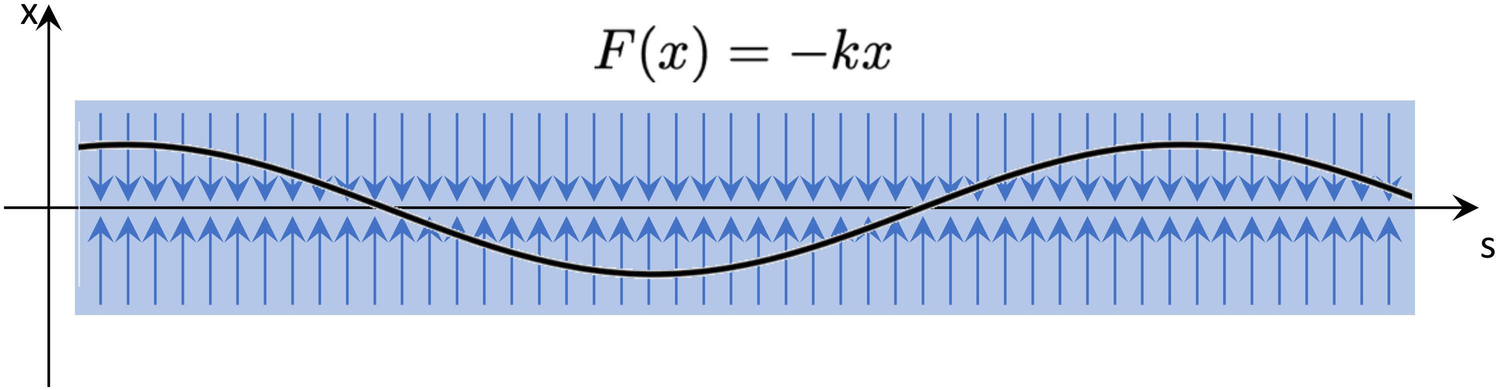
$$\frac{dx}{dt} = v_x$$

Eq. of motion

$$\frac{dv_x}{dt} = -\frac{k}{m}x$$

$$\frac{dx}{dt} = v_x$$

$$F(x) = -kx$$



Such a system **preserves the initial energy of the particle**, defined as:

$$E = \frac{1}{2}mv_x^2 + \frac{1}{2}kx^2 = \text{const.}$$

Kinetic energy

Potential energy

Proof:

$$\frac{dE}{dt} = mv_x\frac{dv_x}{dt} + kx\frac{dx}{dt}$$

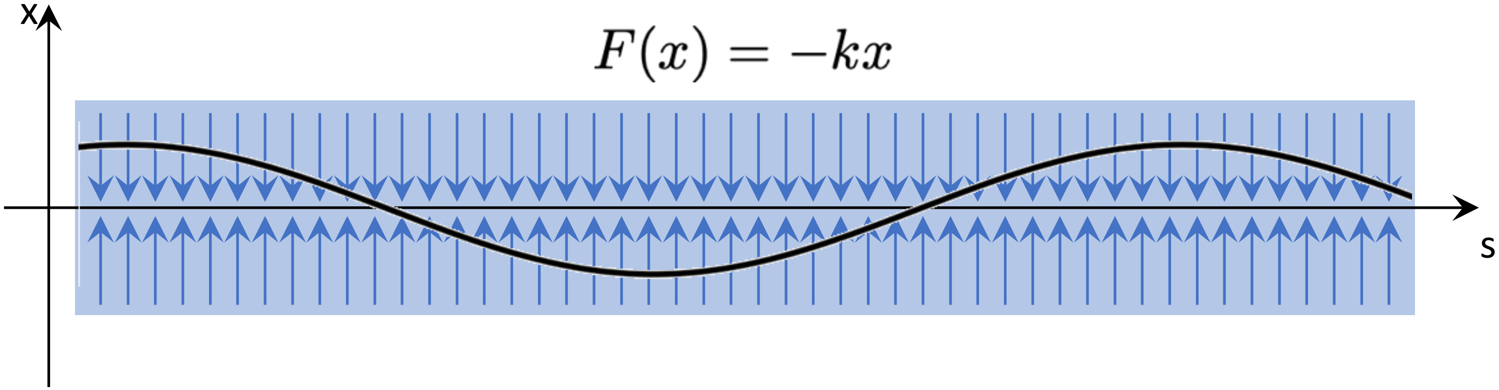
$$= mv_x\left(\frac{dv_x}{dt} + \frac{k}{m}x\right) = 0$$

Tracking simulations

Eq. of motion

$$\frac{dv_x}{dt} = -\frac{k}{m}x$$

$$\frac{dx}{dt} = v_x$$



We compare **two numerical methods** to compute x(t):

Method 1: We use a numerical integration method to find an **approximated solution** to the **exact problem**

Eq. of motion in vector form

$$\frac{d\mathbf{z}}{dt} = \mathbf{f}(\mathbf{z})$$

with: $\mathbf{z}(t) = \begin{pmatrix} x(t) \\ v_x(t) \end{pmatrix}$

We introduce a **discrete time-step** Δt

Runge-Kutta scheme:

$$\mathbf{z}_{n+1} = \mathbf{z}_n + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

with:

$$\mathbf{k}_1 = \Delta t\ \mathbf{f}(\mathbf{z}_n),$$

$$\mathbf{k}_2 = \Delta t\ \mathbf{f}\left(\mathbf{z}_n + \frac{\mathbf{k}_1}{2}\right)$$

$$\mathbf{k}_3 = \Delta t\ \mathbf{f}\left(\mathbf{z}_n + \frac{\mathbf{k}_2}{2}\right)$$

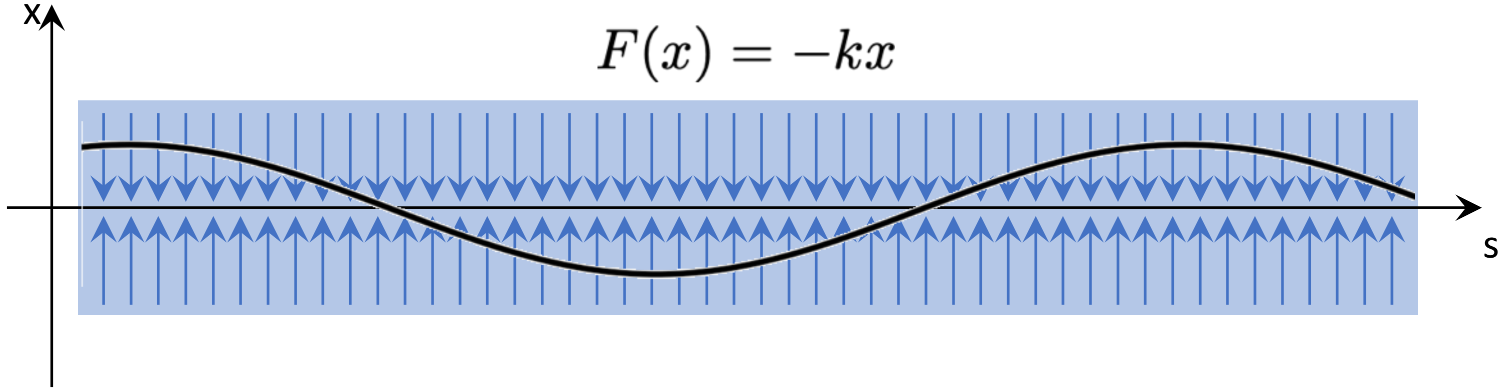
$$\mathbf{k}_4 = \Delta t\ \mathbf{f}(\mathbf{z}_n + \mathbf{k}_3)$$

Eq. of motion

$$\frac{dv_x}{dt} = -\frac{k}{m}x$$

$$\frac{dx}{dt} = v_x$$

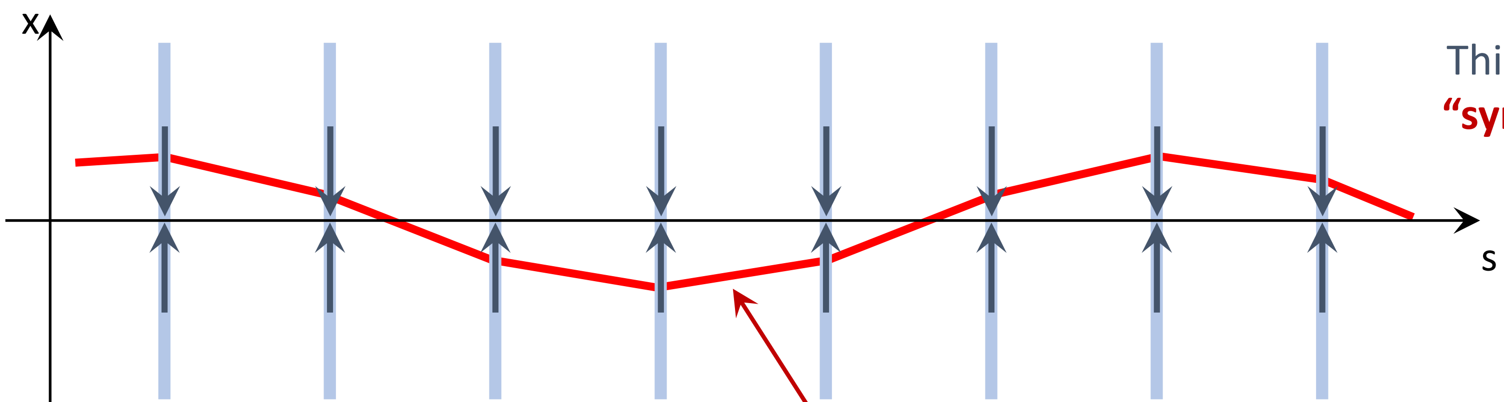
$$F(x) = -kx$$



We compare **two numerical methods** to compute x(t):

Method 2: We find an **approximated problem** for which we are able to compute the **exact solution**

We **concentrate our focusing force at discrete locations ("lenses")**



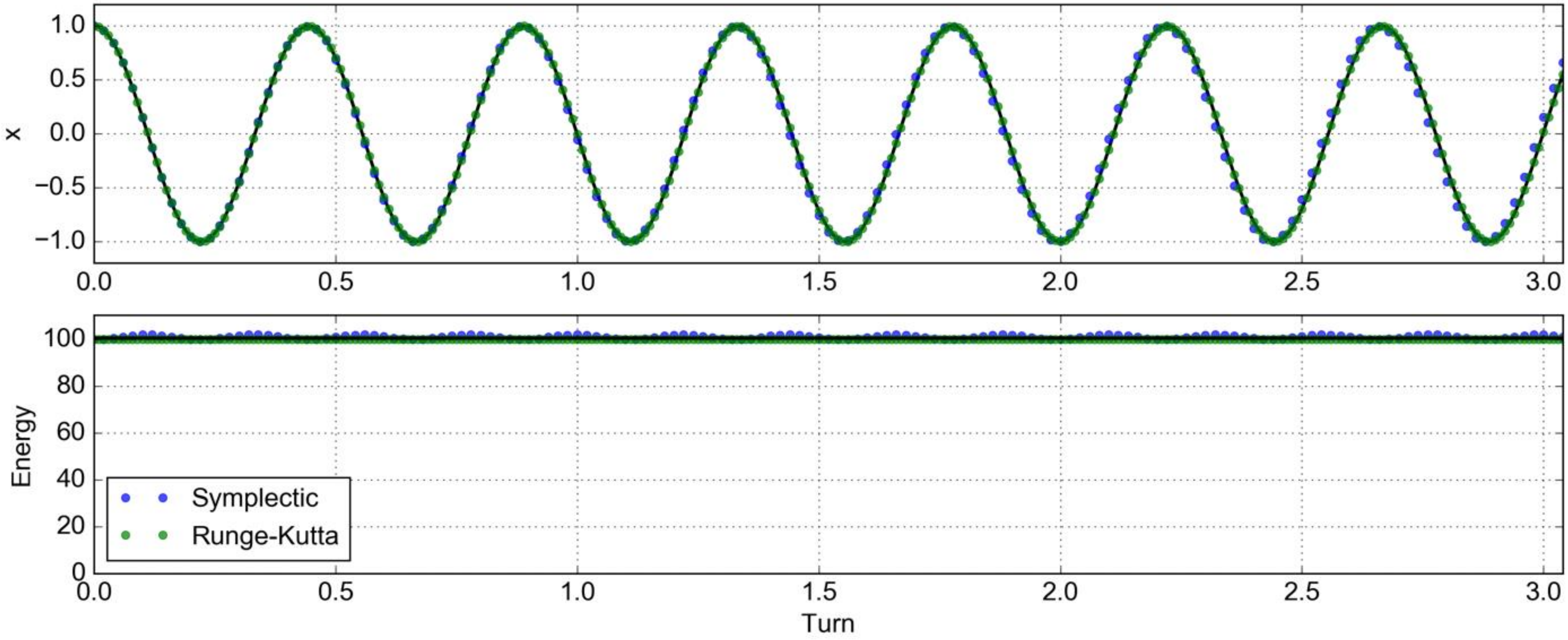
This is an example of **"symplectic scheme"**

$$\Delta v_x = -\frac{k}{m}x\Delta L$$

In between lenses the particle simple moves along a **straight line**

Tracking simulations: a numerical experiment

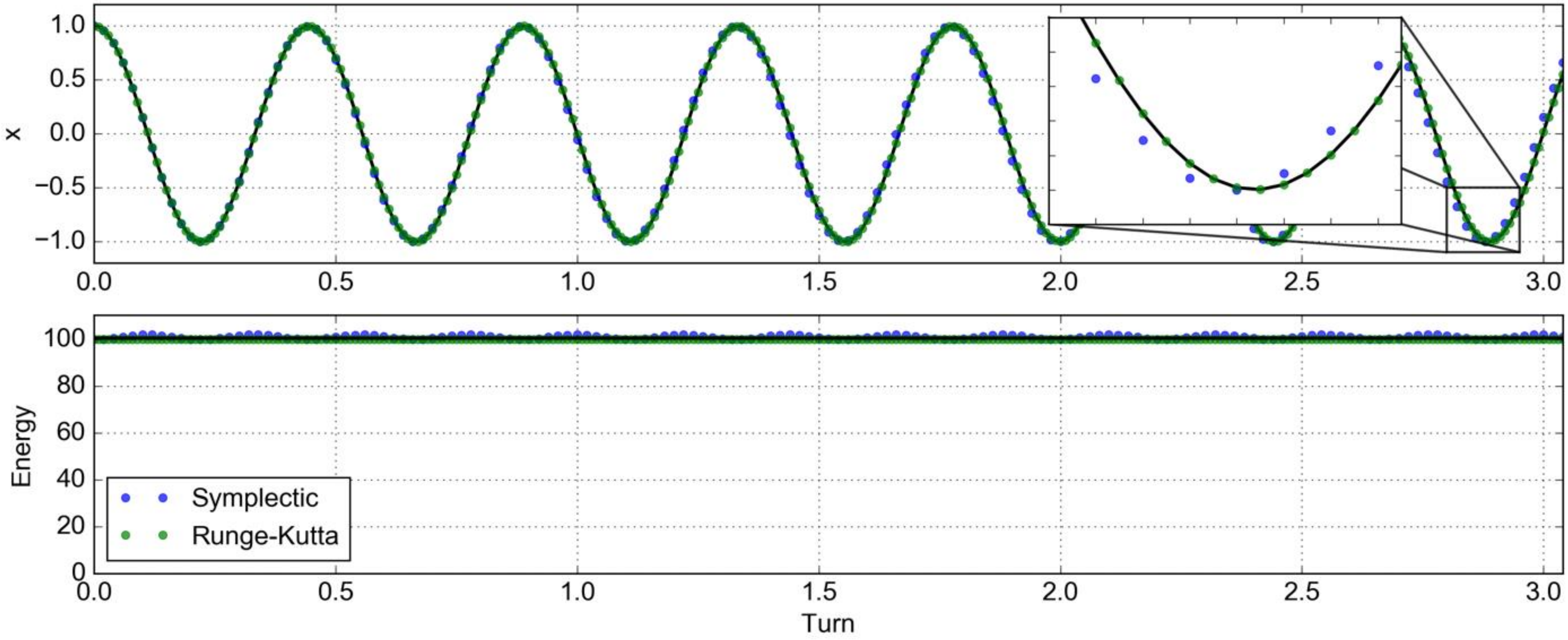
We perform a **numerical experiment** to compare the two methods:



Tracking simulations: a numerical experiment

We perform a **numerical experiment** to compare the two methods:

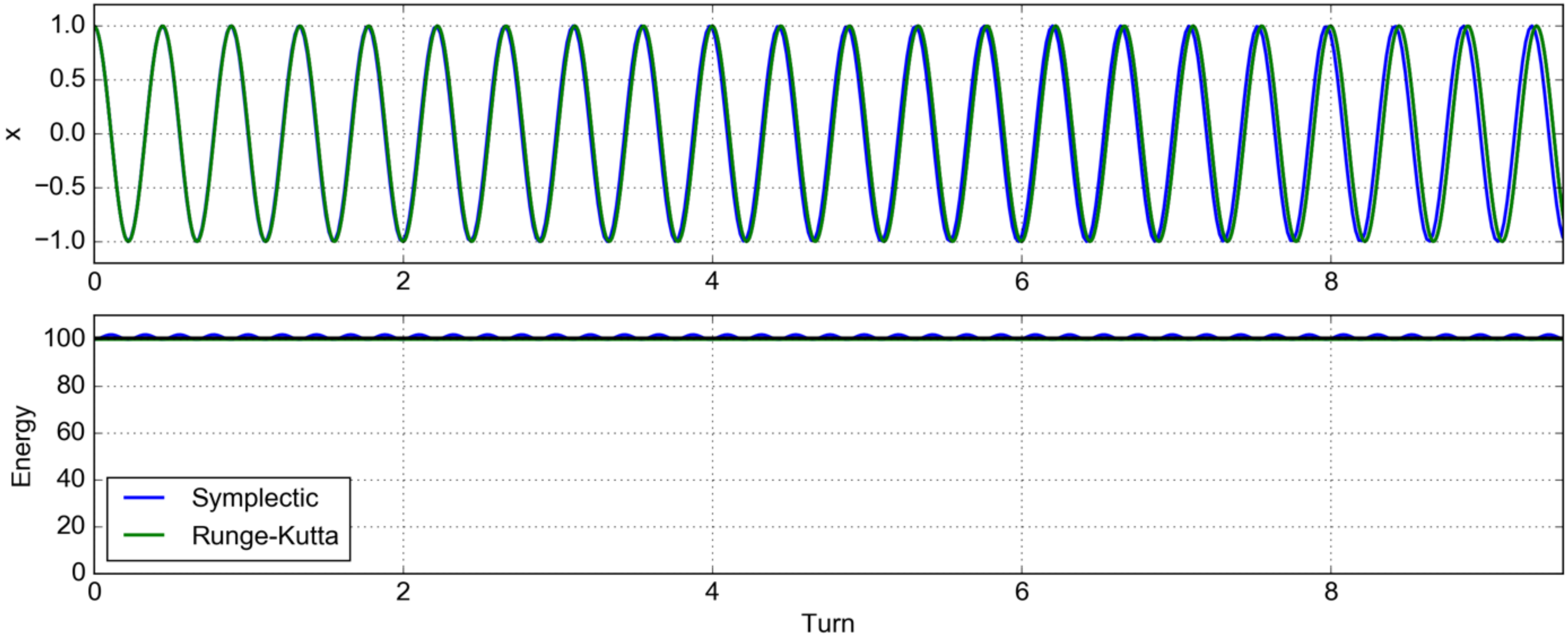
- The **Runge-Kutta** method is **more accurate** on a **short time interval**



Tracking simulations: a numerical experiment

We perform a **numerical experiment** to compare the two methods:

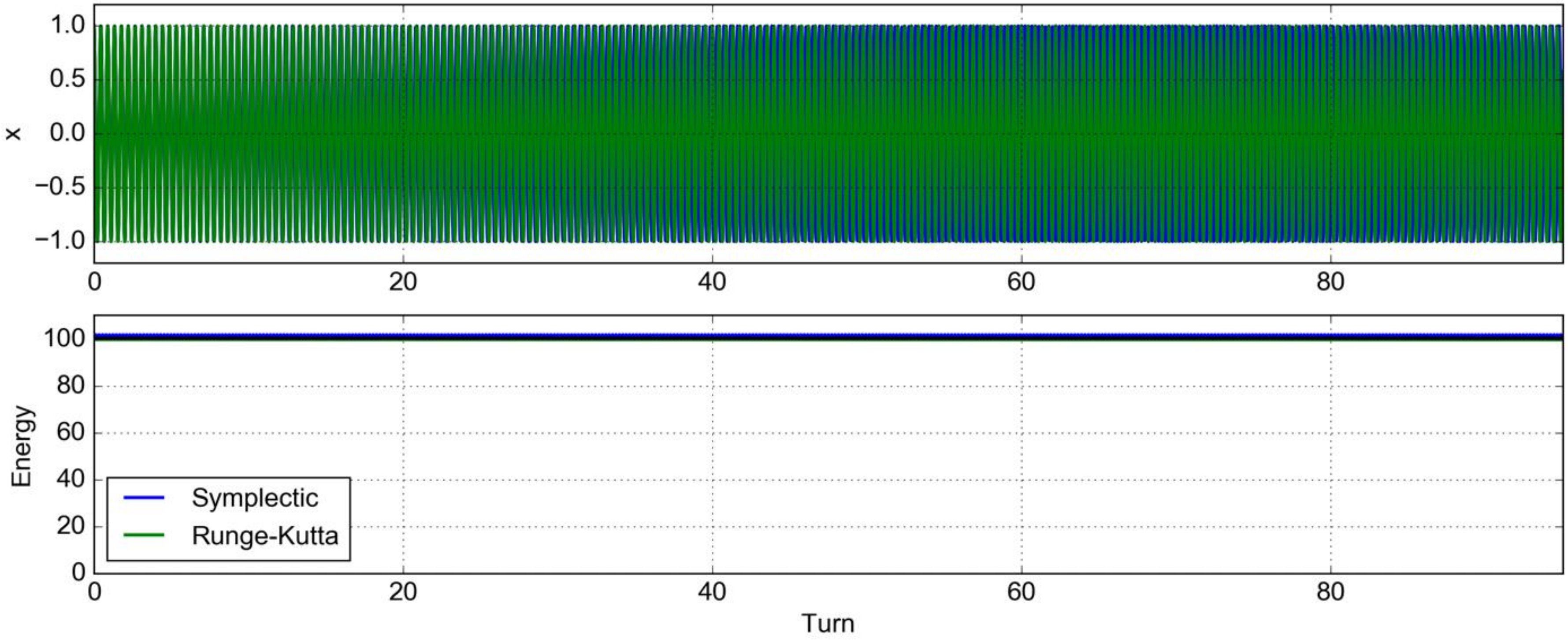
- The **Runge-Kutta** method is **more accurate** on a **short time interval**
- On **very long time-spans** …



Tracking simulations: a numerical experiment

We perform a **numerical experiment** to compare the two methods:

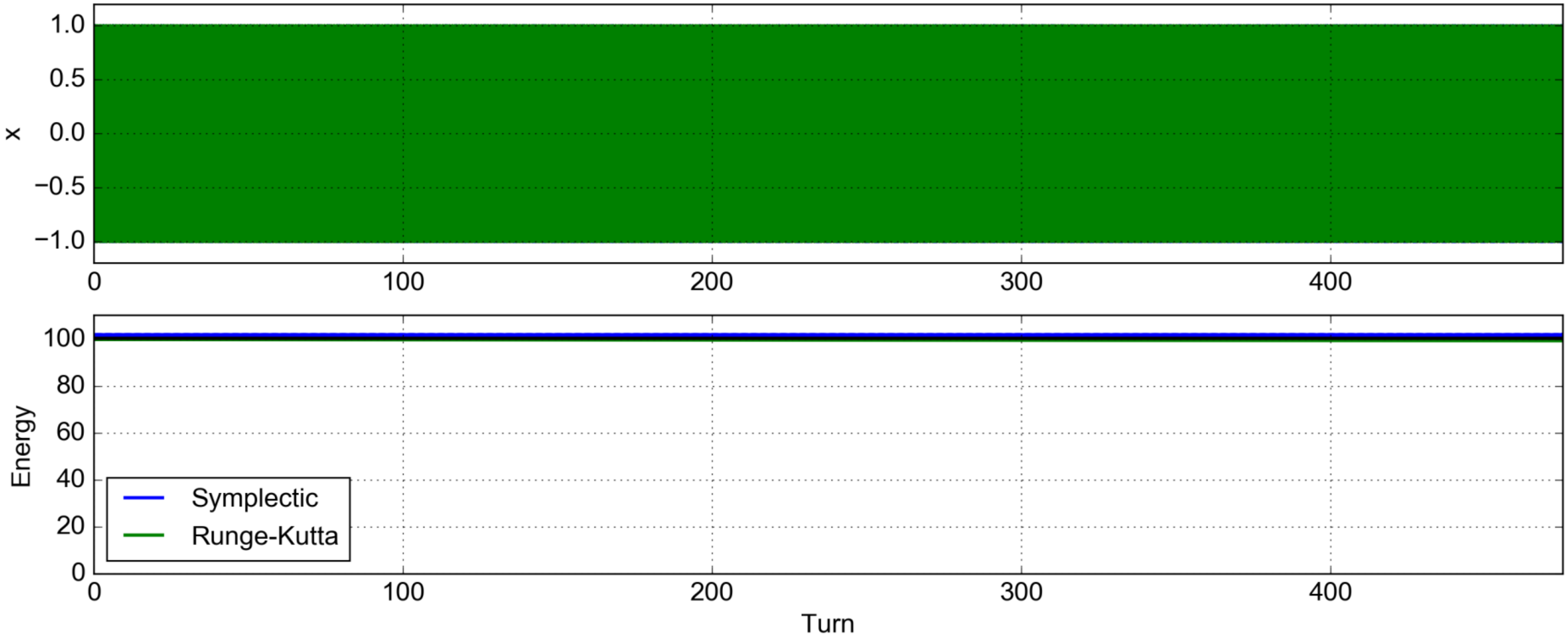
- The **Runge-Kutta** method is **more accurate** on a **short time interval**
- On **very long time-spans** …



Tracking simulations: a numerical experiment

We perform a **numerical experiment** to compare the two methods:

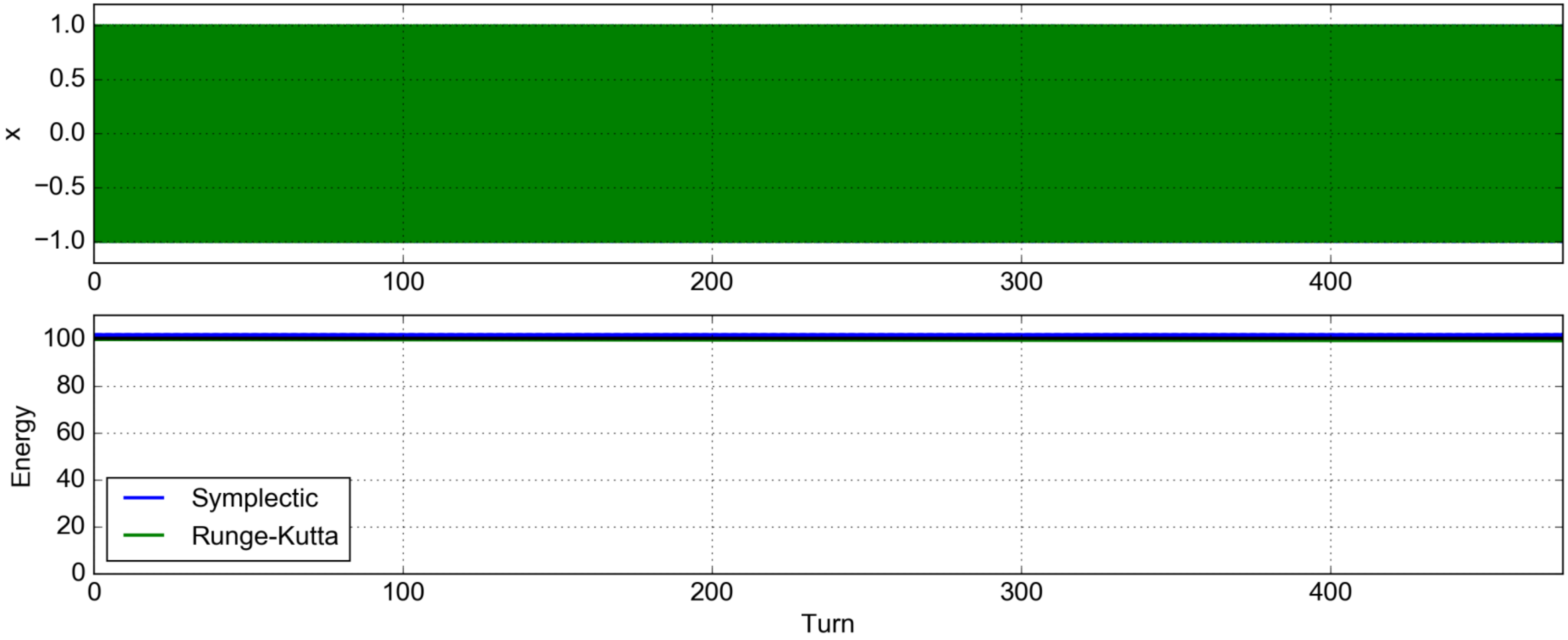
- The **Runge-Kutta** method is **more accurate** on a **short time interval**
- On **very long time-spans** …



Tracking simulations: a numerical experiment

We perform a **numerical experiment** to compare the two methods:

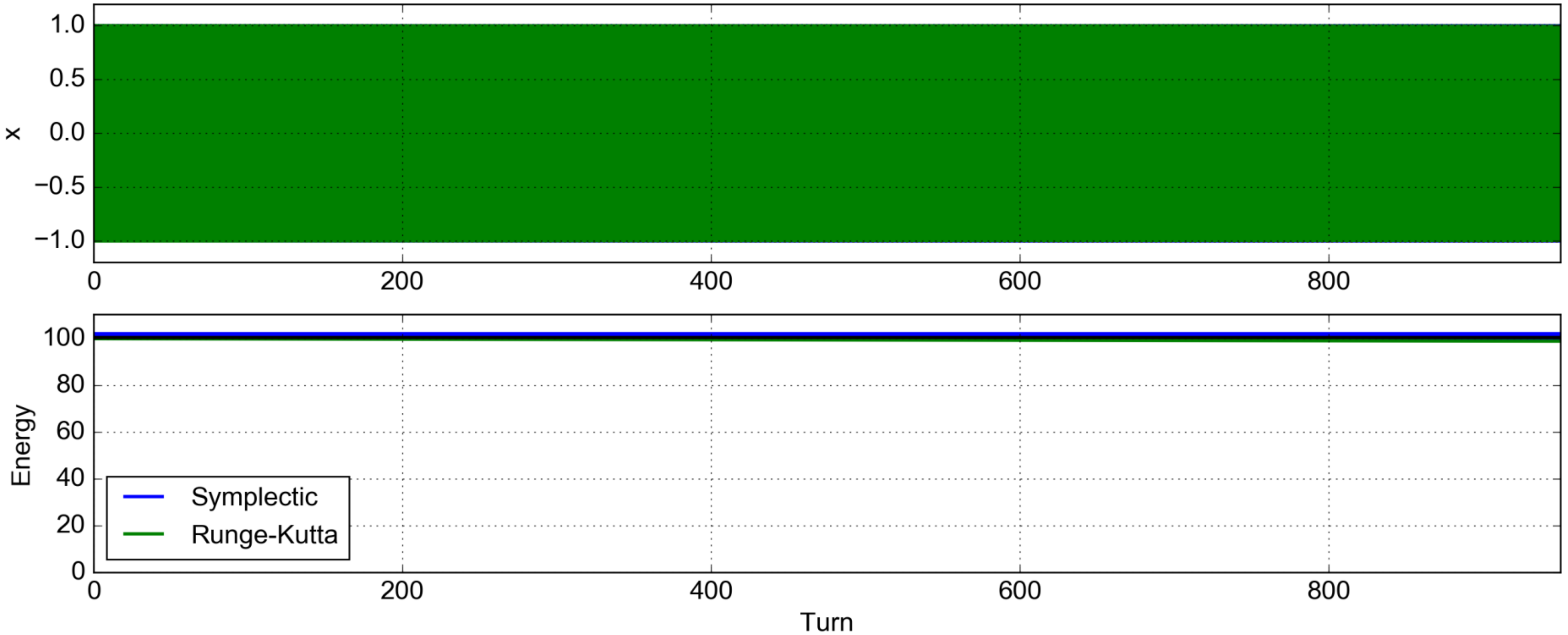
- The **Runge-Kutta** method is **more accurate** on a **short time interval**
- On **very long time-spans** …



Tracking simulations: a numerical experiment

We perform a **numerical experiment** to compare the two methods:

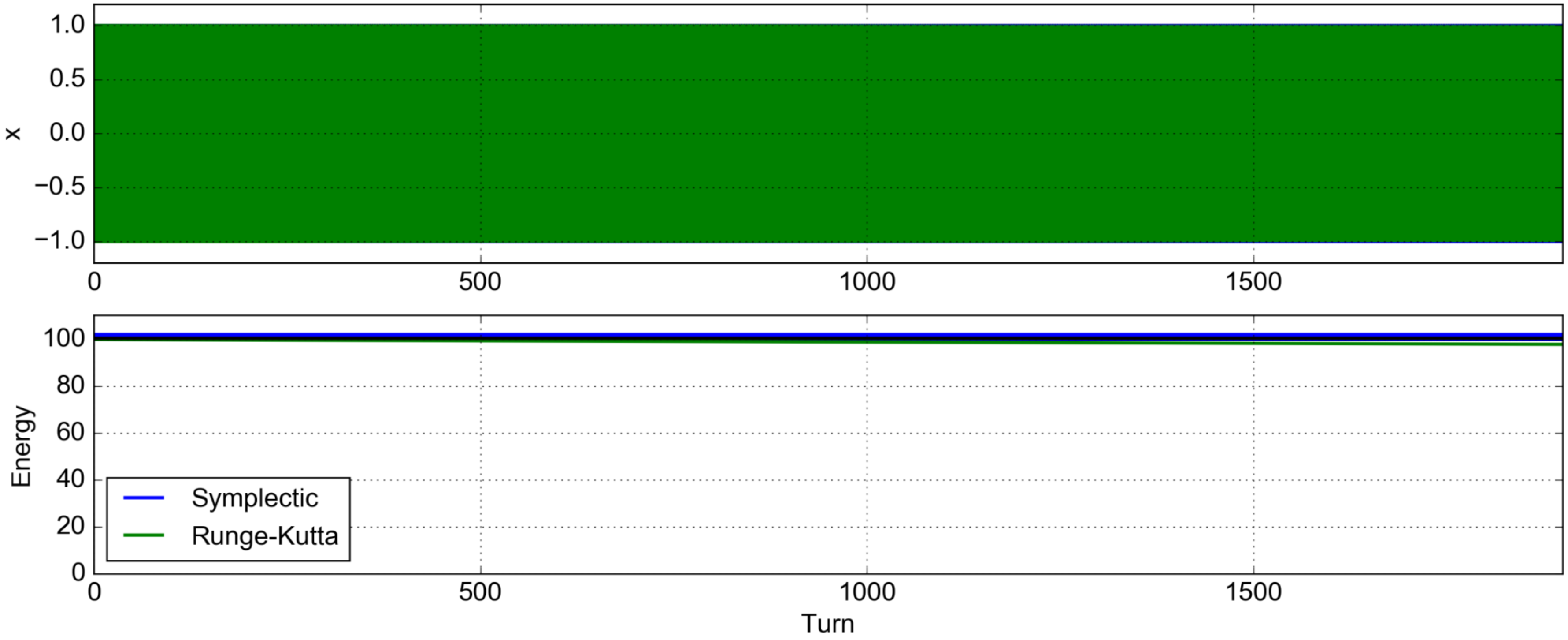
- The **Runge-Kutta** method is **more accurate** on a **short time interval**
- On **very long time-spans** …



Tracking simulations: a numerical experiment

We perform a **numerical experiment** to compare the two methods:

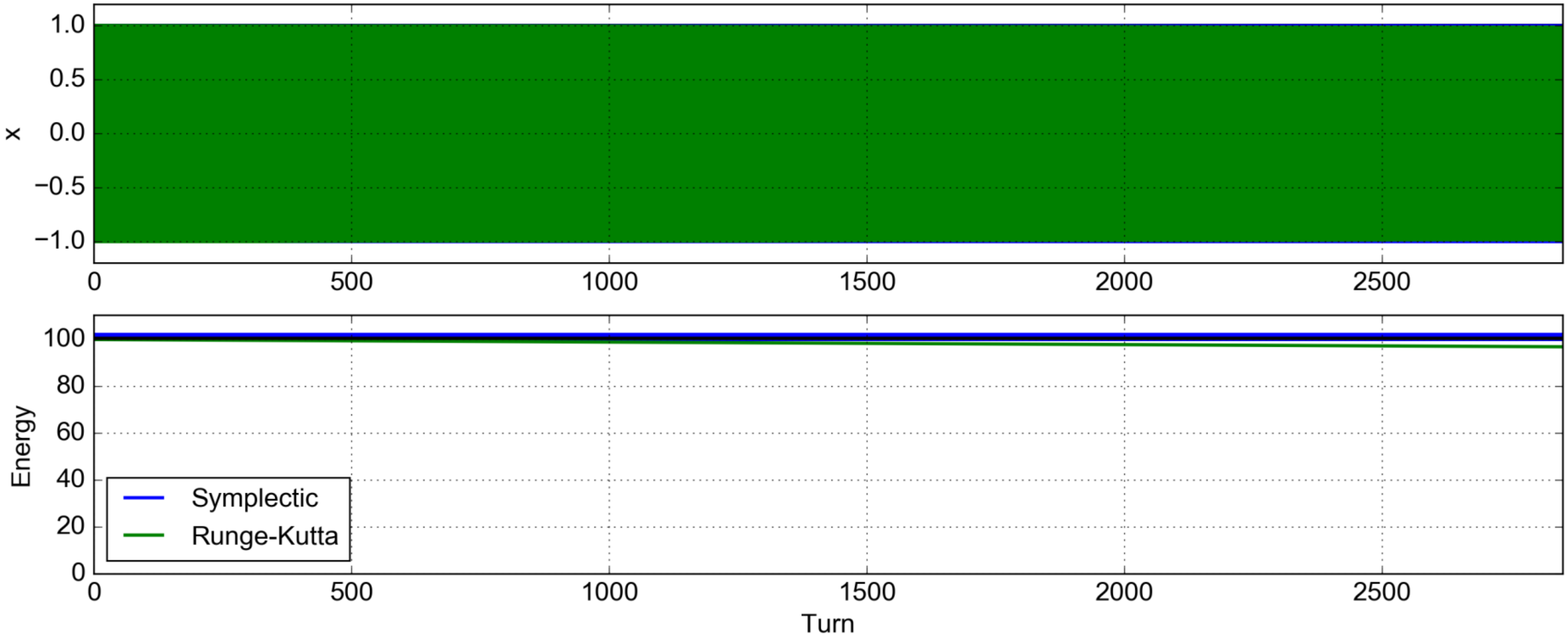
- The **Runge-Kutta** method is **more accurate** on a **short time interval**
- On **very long time-spans** …



Tracking simulations: a numerical experiment

We perform a **numerical experiment** to compare the two methods:

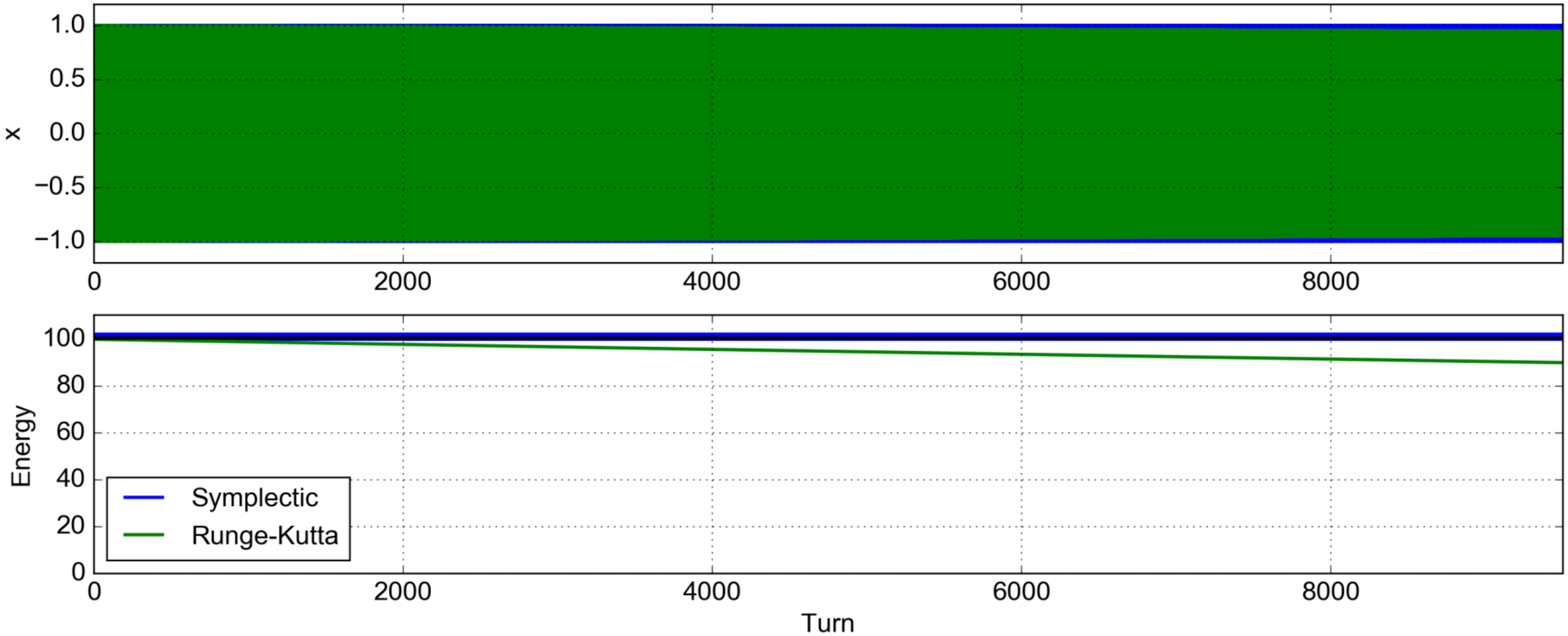
- The **Runge-Kutta** method is **more accurate** on a **short time interval**
- On **very long time-spans** …



Tracking simulations: a numerical experiment

We perform a **numerical experiment** to compare the two methods:

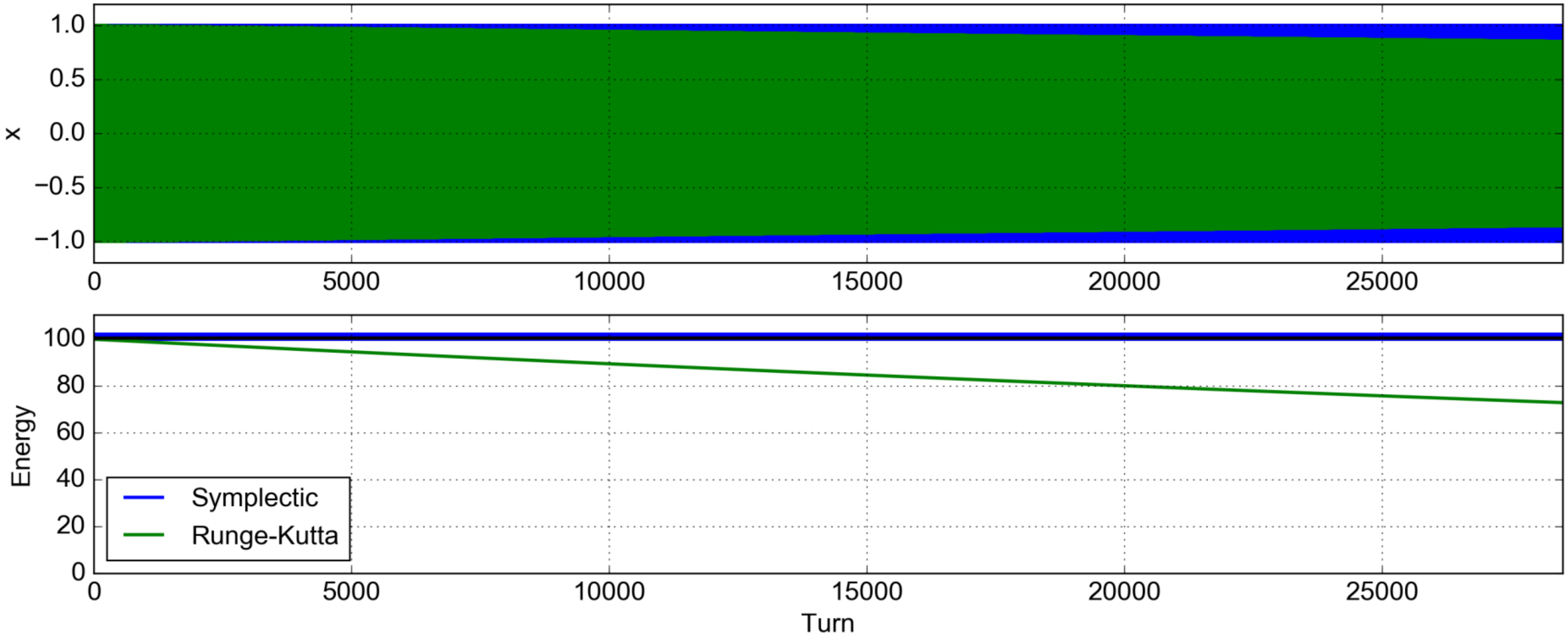
- The **Runge-Kutta** method is **more accurate** on a **short time interval**
- On **very long time-spans** the Runge-Kutta method **slowly “consumes” the energy of the particles**



Tracking simulations: a numerical experiment

We perform a **numerical experiment** to compare the two methods:

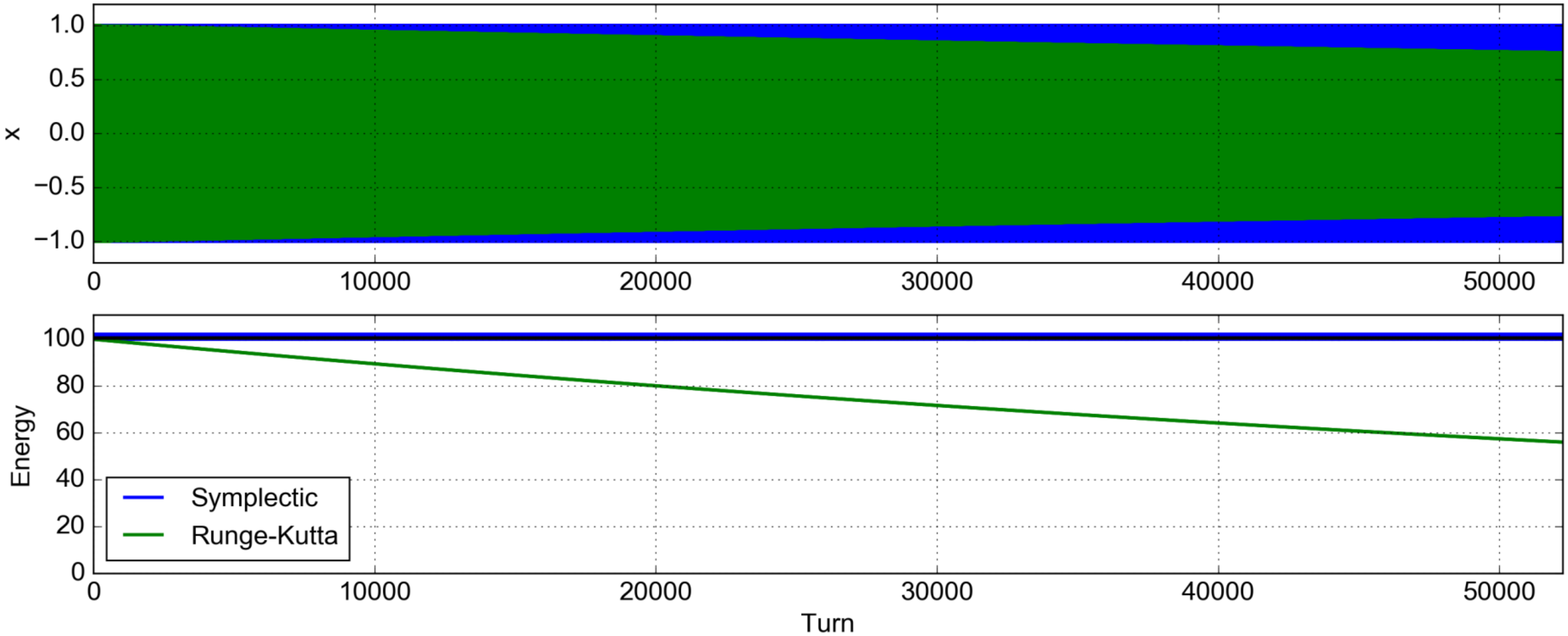
- The **Runge-Kutta** method is **more accurate** on a **short time interval**
- On **very long time-spans** the Runge-Kutta method **slowly "consumes" the energy of the particles**



Tracking simulations: a numerical experiment

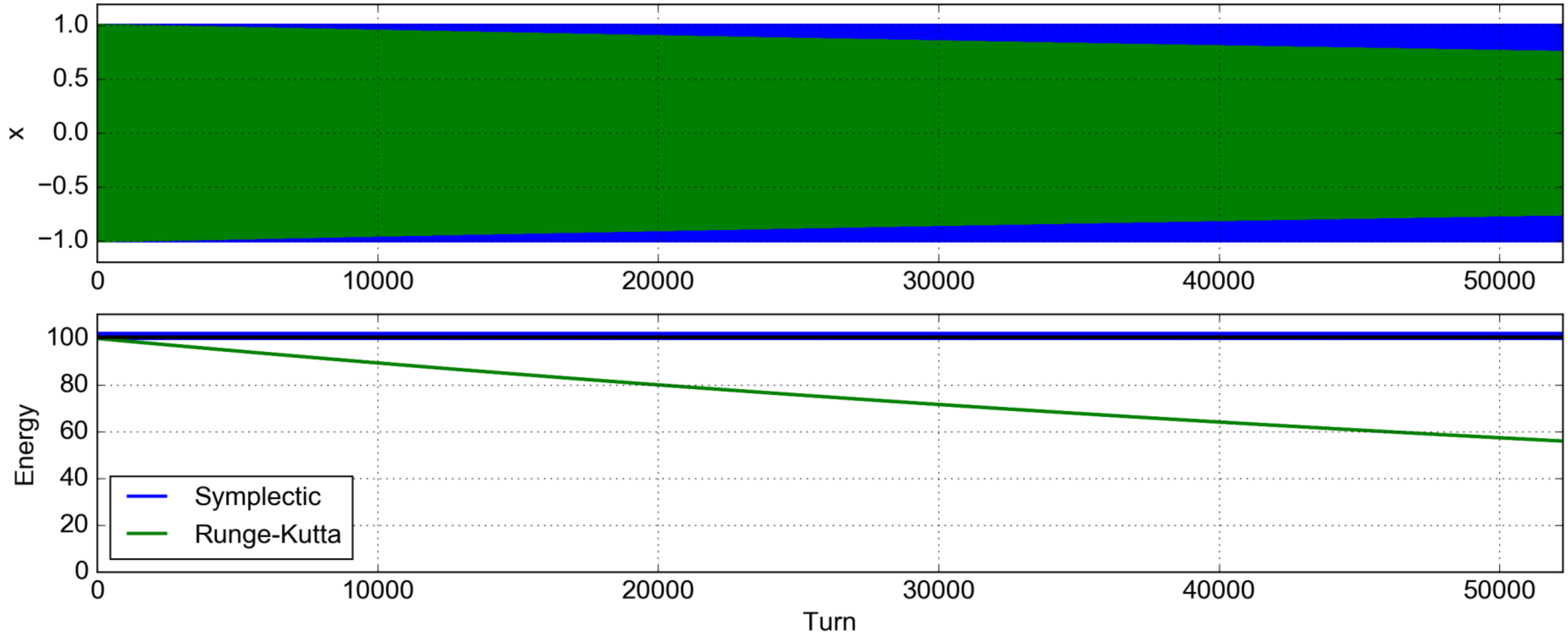
We perform a **numerical experiment** to compare the two methods:

- The **Runge-Kutta** method is **more accurate** on a **short time interval**
- On **very long time-spans** the Runge-Kutta method **slowly "consumes" the energy of the particles**
  - **"Fake" physical phenomena** are introduced
  - Runge-Kutta **cannot be used to predict slow effects** on the beam
- In spite of being less accurate on short times, **the symplectic scheme does not suffer from this issues**



Tracking simulations: a numerical experiment

- In general, for long-term tracking we do **need to use symplectic algorithms**:
  → The numerical solution needs to **preserve fundamental properties** of the physical system such as energy conservation

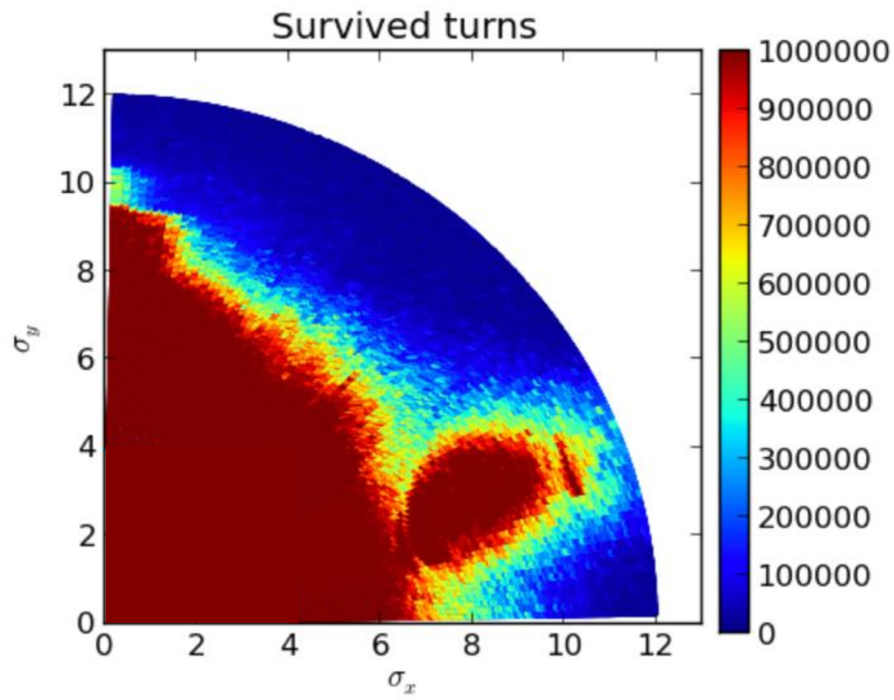


- **Particle accelerators**
  - Examples of applications
  - Working principles
- **Beam optics calculations**
  - An example: the LHC betatron squeeze
  - Numerical optimization techniques
- **Particle tracking**
  - Motivations
  - Need for symplectic methods
  - Experience with GPU computing

In accelerator studies we are interested **tracking a large number of particles** (~100 000) to probe different initial conditions:

- Simulation of each particle is independent
- We are facing an **"embarrassingly parallel problem"**

→ Particularly suited for **GPU acceleration**



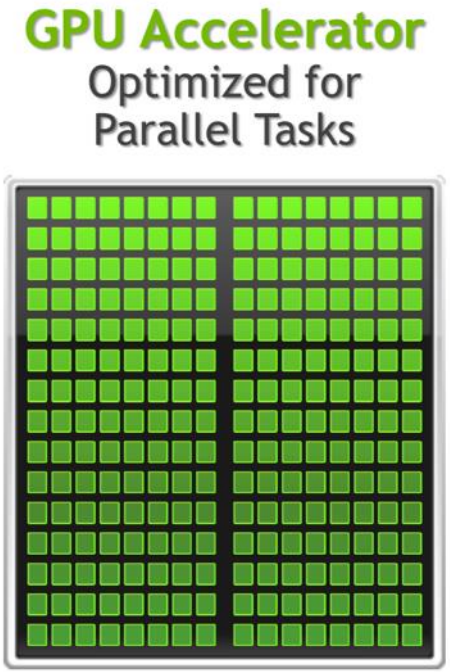
GPU computing

- **Graphics Processing Units (GPUs)** are chips developed since the 80s to perform **graphics calculations** (video rendering), typically installed on a **video-card**
    - Main applications are gaming and computer graphics in general
- Since the early 2000s GPU vendors provide tools to use the GPUs also for **general-purpose parallel computing:**
    - **Libraries** and **tools** to exploit these resources have flourished
    - **Cards dedicated to high performance computing** have been commercialized

Gaming card

HPC GPU card

GPU accelerated server



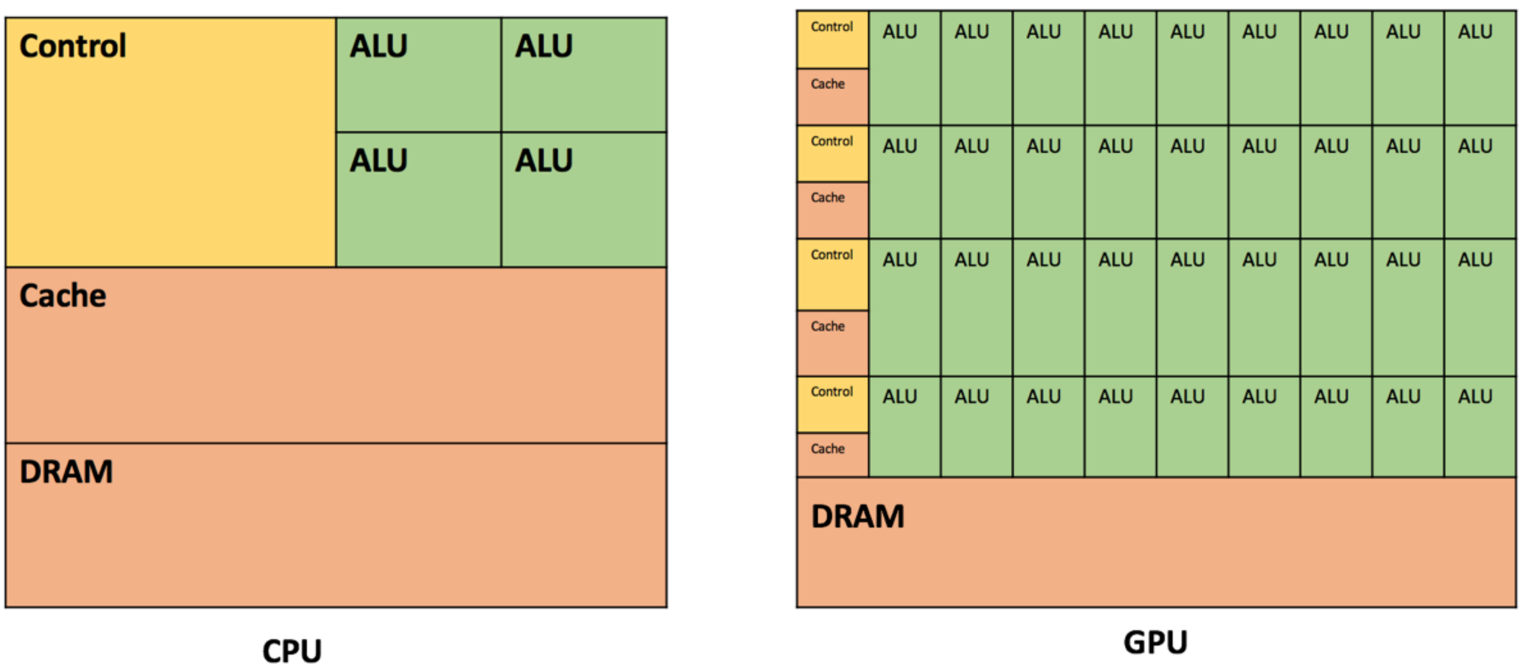
- Has a **small number of complex computing cores** (up to 8)
- Very **fast clock rates**
- Can access large memory (> 100 GB)

- Has a **large number of simpler computing cores (>1000)**
- **Slower clock rates**
- Can access relatively small memory (~16 GB)

https://www.olcf.ornl.gov/wp-content/uploads/2018/06/intro_to_HPC_gpu_computing.pdf

Resources allocation:

- A GPU has **more resources dedicated to Arithmetic-Logic operations** (ALUs) compared to a covariational CPU
- A GPU has **less resources dedicated to control and cache memory**



https://www.olcf.ornl.gov/wp-content/uploads/2018/06/intro_to_HPC_gpu_computing.pdf

GPU vs CPU: a simplified summary

CPU
optimized for speed
(but reduced capacity)

GPU
optimized for capacity
(but reduced speed)

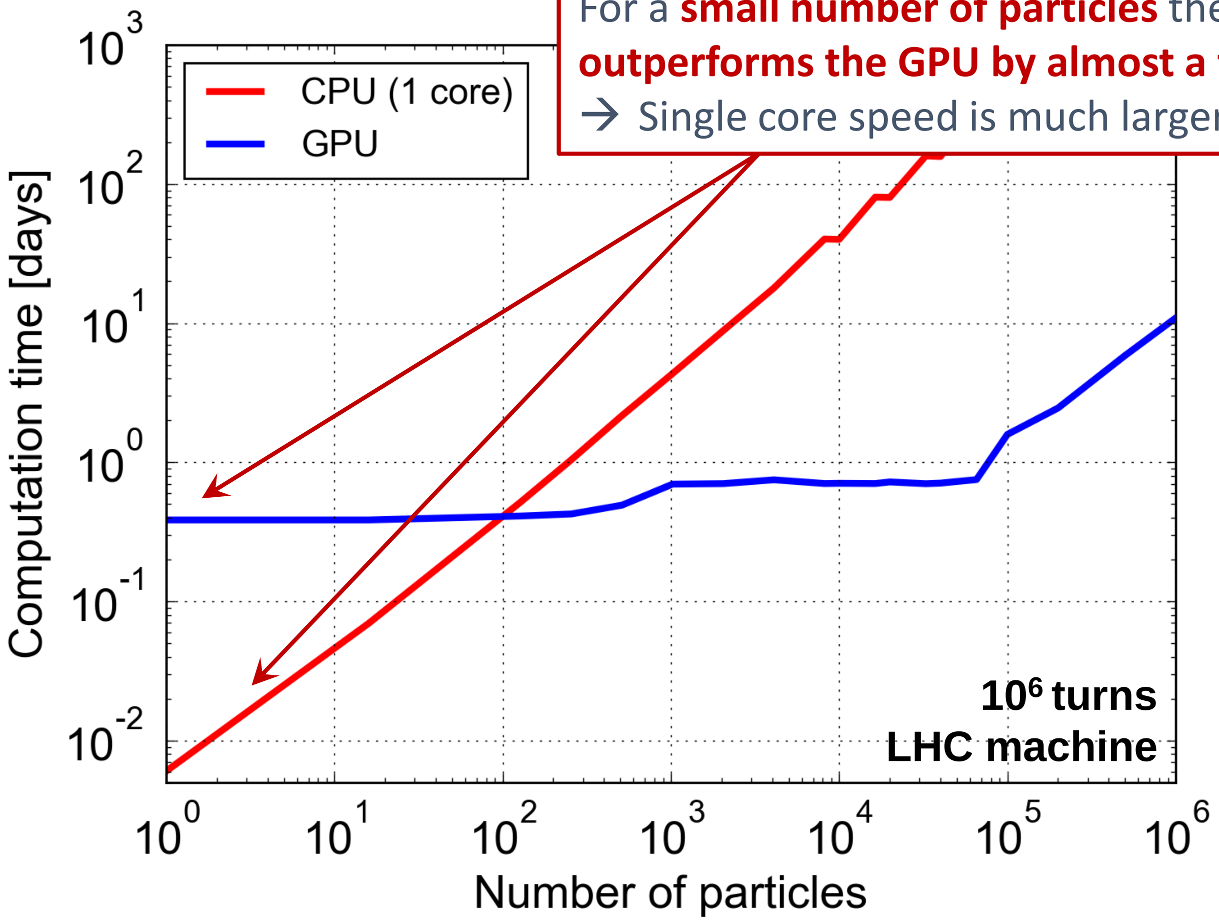
Which is better depends on your needs...

https://www.olcf.ornl.gov/wp-content/uploads/2018/06/intro_to_HPC_gpu_computing.pdf

GPU vs CPU: a real life example

The **sixtracklib tracking library** (recently developed at CERN) allows performing **tracking simulations both on CPU and on GPU**

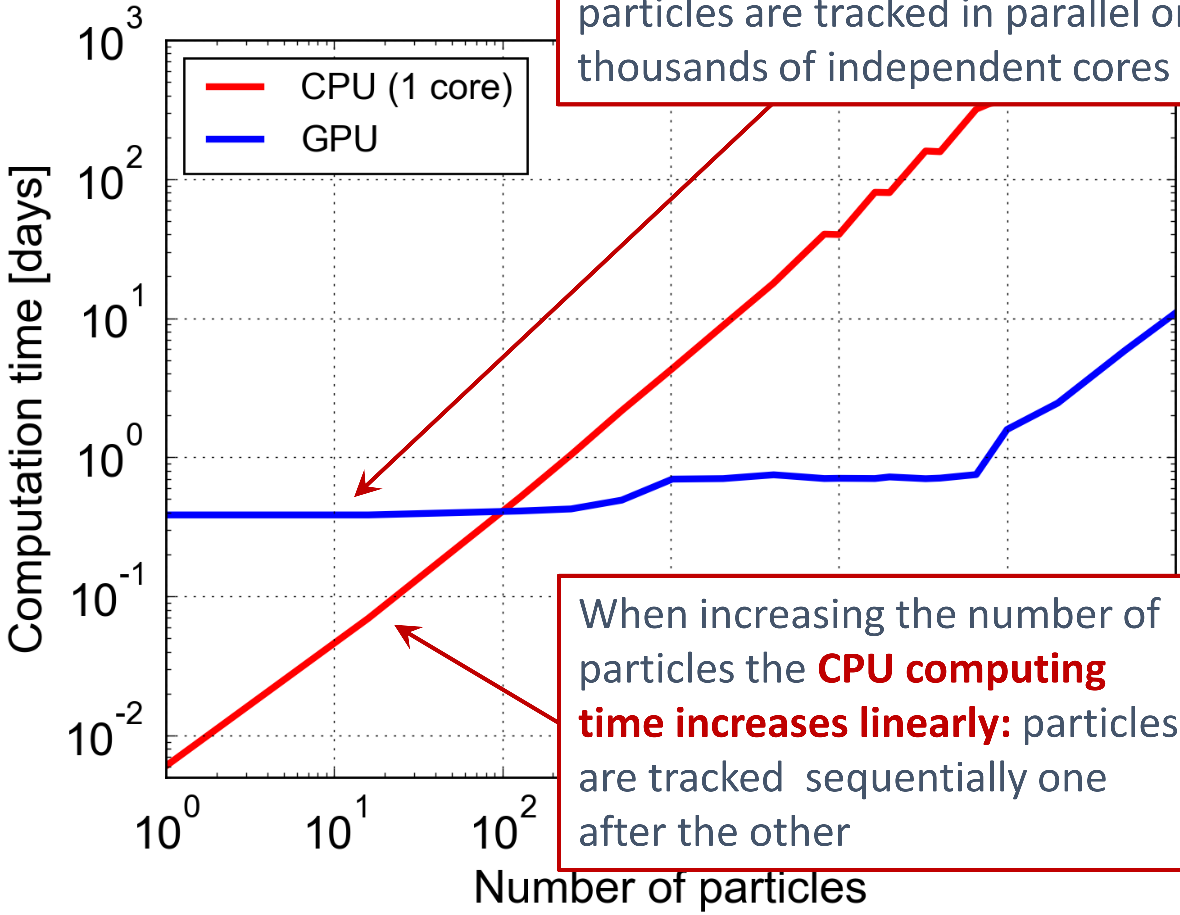
For a **small number of particles** the **CPU outperforms the GPU by almost a factor of 10**
→ Single core speed is much larger for the CPU



https://github.com/SixTrack/sixtracklib

GPU vs CPU: a real life example

The **sixtracklib tracking library** (recently developed at CERN) allows performing **tracking simulations both on CPU and on GPU**



The **GPU computing time remains constant:** particles are tracked in parallel on using thousands of independent cores

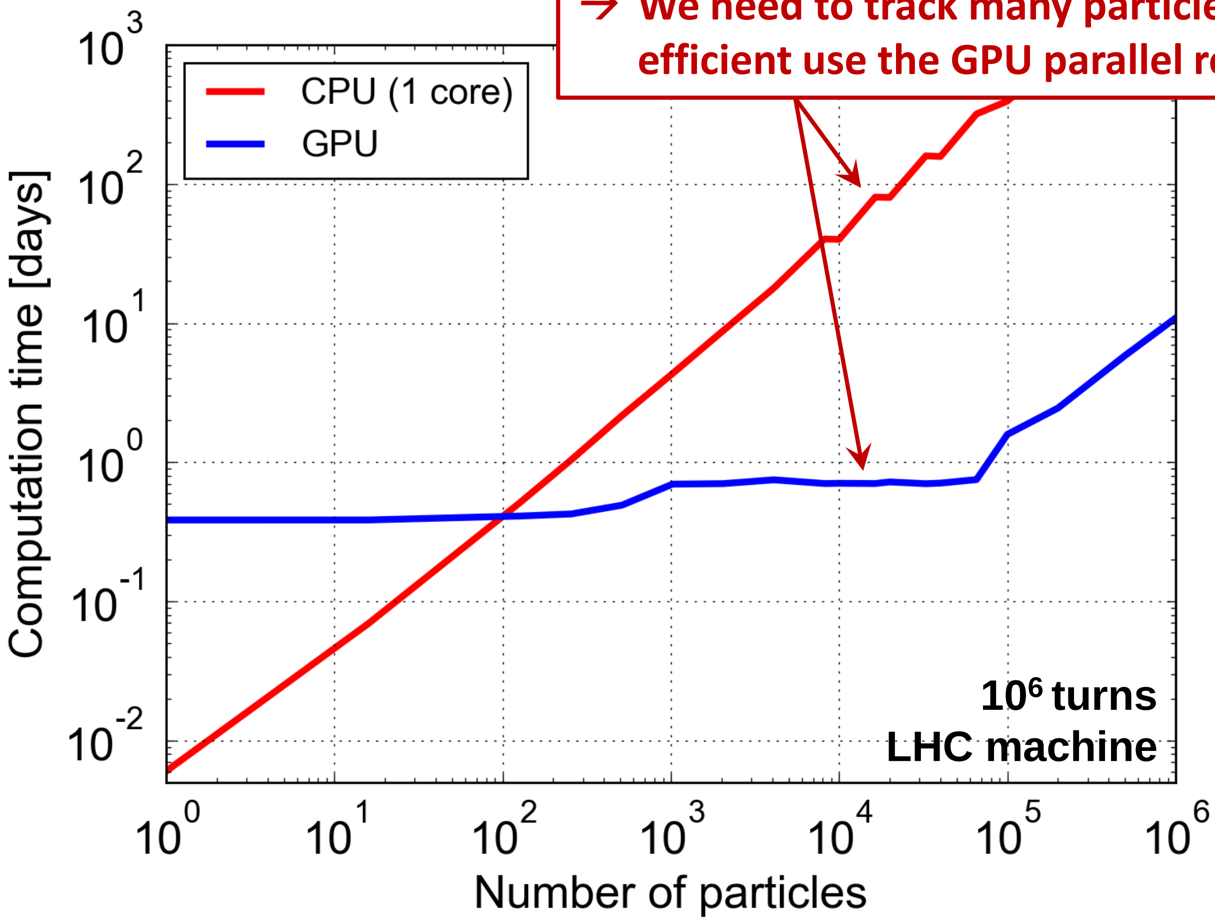
When increasing the number of particles the **CPU computing time increases linearly:** particles are tracked sequentially one after the other

https://github.com/SixTrack/sixtracklib

GPU vs CPU: a real life example

The **sixtracklib tracking library** (recently developed at CERN) allows performing **tracking simulations both on CPU and on GPU**

For a large number of particles **the GPU outperforms the CPU** by a factor of 500 **(1 day vs 1 year!)**
→ **We need to track many particles to make efficient use the GPU parallel resources**

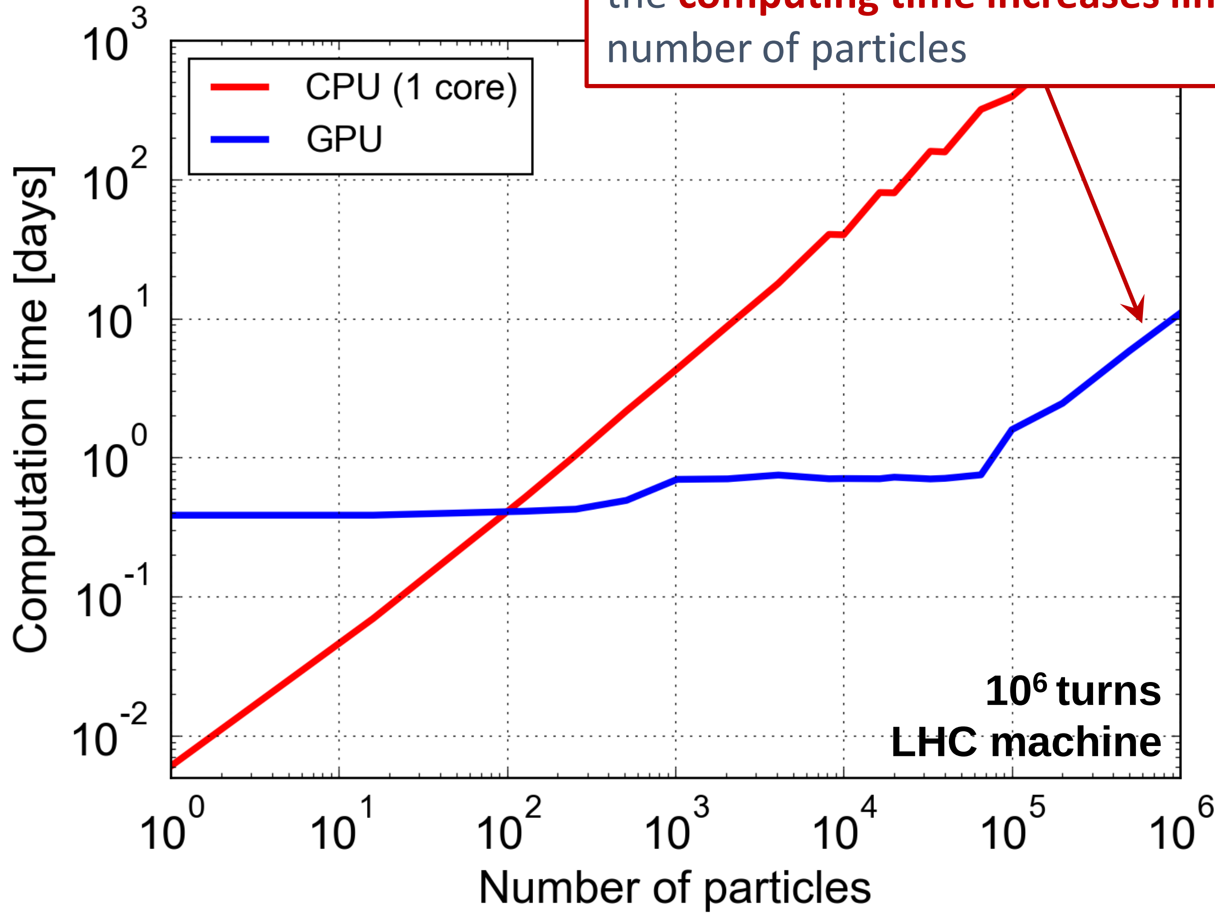


https://github.com/SixTrack/sixtracklib

GPU vs CPU: a real life example

The **sixtracklib tracking library** (recently developed at CERN) allows performing **tracking simulations both on CPU and on GPU**

Above a certain number of particles also the resources on the **GPU become saturated** and the **computing time increases linearly** with the number of particles

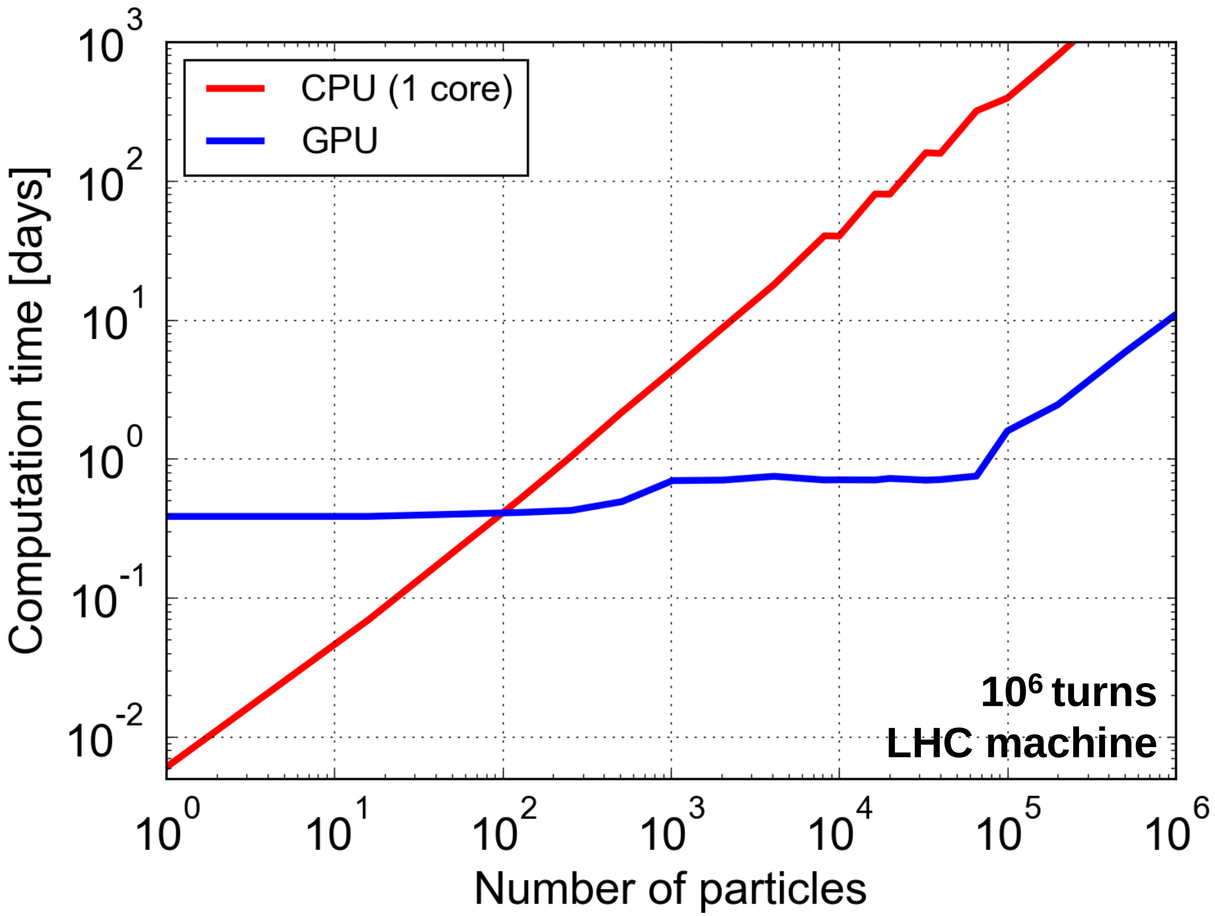


https://github.com/SixTrack/sixtracklib

GPU vs CPU: a real life example

The **sixtracklib tracking library** (recently developed at CERN) allows performing **tracking simulations both on CPU and on GPU**

When tracking a **large number of particles GPUs become very attractive** (still the price of the device can be more expensive…)



https://github.com/SixTrack/sixtracklib

Summary

- A **particle accelerator** uses **electromagnetic field** to accelerate and manipulate charged particles
  - **Accelerating structures** are used to increase the energy of the particles
  - **Dipole magnets** are used to keep the beams on a closed trajectory
  - **Quadrupole magnets** are used to confine (focus) the particles

- In the presence of **linear forces alone** it is possible to **compute the beam envelope** (optics) without computing the single particle trajectories
  - **Quadrupole strengths** can be **used to shape the particle beam envelope** (in the same way in which lenses can be used to shape a beam of light)
  - **Numerical optimizers (like the gradient method)** need to be used to identify suitable quadrupole strengths as a function of given constraints on the beam envelope

- **Particle tracking** is the simulation of individual particles in the accelerator over a very large number of turns:
  - **Symplectic algorithms** are required in order to preserve fundamental properties of the physical system
  - **GPU computers** are particularly suited for this kind of simulations

Next

- So far we have studied **"single-particle" methods**, which neglect the interactions among circulating particles

- In the second part we will focus on methods for **"collective effects"**, which are particularly relevant when the beam is very intense (large number of particles)

Thanks for your attention!