Proceedings of the CAS-CERN Accelerator School: Intensity Limitations in Particle Beams, Geneva, Switzerland, 2–11 November 2015, edited by W. Herr, CERN Yellow Reports: School Proceedings, Vol. 3/2017, CERN-2017-006-SP (CERN, Geneva, 2017)

Numerical Methods I & II

K. Li
CERN, Geneva, Switzerland

Abstract
Numerical methods are a fundamental field of research and application in accelerator physics and beam dynamics. They are widely used to study intensity effects and limitations in accelerators and become more important still in regimes where simple analytical models break down and experiments cannot be easily conducted. In this course we will introduce some basic concepts of numerical methods used to study collective effects in circular accelerators. The course covers macroparticle models, the implementation of the beam dynamics and applications relevant to intensity effects and limitations. This includes basic tracking in the transverse and longitudinal planes, modelling of impedance effects and finally two-stream effects such as electron clouds.

Keywords
Collective effects; instabilities; macroparticle models; numerical methods; simulation.

1 Introduction and concepts

Numerical methods are a fundamental field of research and application in accelerator physics and beam dynamics. They are a very powerful tool to overcome limitations from simplifications inherent to most analytical approaches or the complications connected to many experimental studies.

The advantage of numerical modelling over analytical solutions is that it can usually be done with far fewer approximations in order to study problems which include many of the complications present in the physical reality. For these, usually analytical models are so complicated that they cannot be solved by analytical means or they fail to correctly model the physical reality in the first place. In comparison with experiments, numerical modelling allows full flexibility in parameters and set-up since basically anything that can be modelled can also be studied even beyond practical or technical limitations. Moreover, the level of diagnostics can be arbitrarily accurate, since nearly any observable can be exposed.

The main drawback of numerical models is that they have to be executed on computers and, depending on the complexity of the problem, they need to run for a long time in order for them to produce meaningful results. Therefore, the goal of a good numerical model is to capture as much of the physical reality that is necessary to understand the problem under investigation. At the same time the algorithms should be sufficiently easy and fast to provide solutions within a reasonable amount of time. If implemented correctly, numerical models are able to capture the majority of physical effects and allow for a vast range of diagnostics and data analysis to understand the underlying mechanisms that pose intensity limitations.

There are several different ways of numerically modelling beam dynamics and intensity effects. These range from time-domain or frequency-domain Vlasov solvers (MOSES [1] and DELPHI [2]), circulant matrix models (BimBim [3]) or beam envelope tracking algorithms (HOMDYN [4]), for example. In these lectures, we will investigate and focus on macroparticle models. Examples of modern implementations of macroparticle models are ECLOUD [5] or HEADTAIL [6], which was among the first to treat two-stream effects in a strong–strong manner. As we will see, macroparticle models are the most natural way of mapping the model of a physical particle beam onto a computer system. They generally provide an easy and straightforward platform for implementing almost any known physical effect in beam dynamics without having to move to a great level of abstraction. In that sense, they are ideally suited to tackle most of the problems encountered in beam dynamics and collective effects and also to introduce some of the basic concepts of numerical methods.

1.1 Macroparticle models and computer systems

Macroparticle models, obviously, make use of macroparticles. A macroparticle is a numerical representation of an ensemble of physical particles clustered together within a representative region in space. This is sketched in Fig. 1. Typical systems of physical particles such as a particle beam in the Large Hadron Collider (LHC) at CERN, for example, consist of more than 1×10^{14} particles. These cannot be practically represented on modern computer systems due to memory limitations. To overcome this limitation, physical particles are grouped together into a single macroparticle in such a way that one is left with roughly of the order of 1×10^{7} macroparticles. This is the dimensionality of systems that can be easily solved on conventional computer systems. In clustering physical particles into a discrete set of macroparticles, one of course has to take care about noise issues. Many effects are now less averaged out and therefore enhanced (consider a single macroparticle representing a cluster of 1×10^{6} physical particles receiving a kick by some element or device—all particles now receive the same kick and move in exactly the same manner, where in reality there would be some finite spread in the motion of each individual physical particle). To make sure the results are still physically relevant and not dominated by numerical noise, one has to perform convergence studies. For this, a numerical parameter is varied, in our example the total number of macroparticles, which essentially is a measure of the granularity of our system, and the results are checked. Obviously, the results should not depend on the choice of the numerical parameter. In this case we say that the result is converged and we accept it as output from a numerical simulation. The results may still be wrong but the error should not originate from numerical noise issues but rather from a systematically wrong modelling of the physical reality.

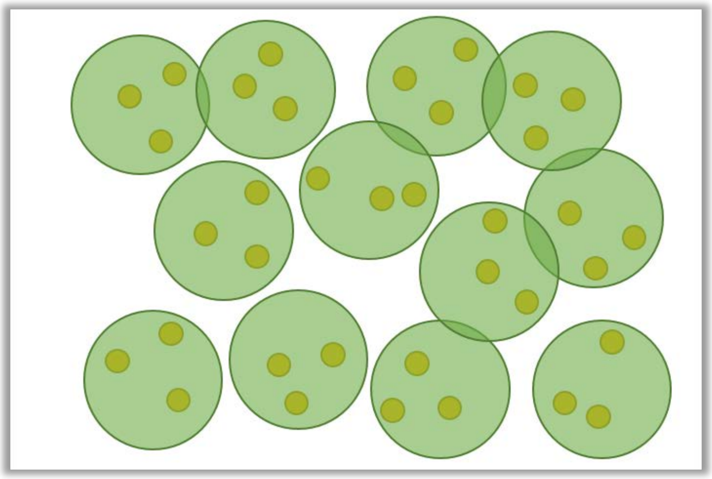


Fig. 1: Macroparticles (green) representing clusters of neighbouring physical particles (yellow)

Macroparticle models naturally map an accelerator–beam system onto modern computer systems. If we look at the main components of an accelerator–beam system we identify the physical particle beam and the accelerator with its elements and devices. If we think of some of the most fundamental components of computer hardware we can identify the main memory (RAM) and the CPU.

To fully describe the dynamics of a charged particle system, such as our physical particle beam, we need to know the generalized coordinates and the canonically conjugate momenta (i.e., the six phase-space variables) of each individual particle along with its charge and mass. In addition, we need to know the action of the different machine elements and devices on the particles, i.e., we need to know how they modify the coordinates and momenta upon interaction of the particles with the machine elements. Numerically, the physical particles are represented by macroparticles and the machine elements are described by functions that update the coordinates and momenta of the macroparticles in a specific manner. Mapping this to the computer hardware, we recognize that the macroparticle system, which basically consists of a set of coordinates and momenta along with charges and masses for each macroparticle, can be represented by some allocated memory block in the main memory which contains and stores these numbers. The machine elements can be represented by functions that pass instructions to the CPU to update the numbers stored in this allocated memory block, thus generating an evolution of the coordinates and momenta of the macroparticle system and, hence, finally, describing beam dynamics. This correspondence is conceptually illustrated in Fig. 2.

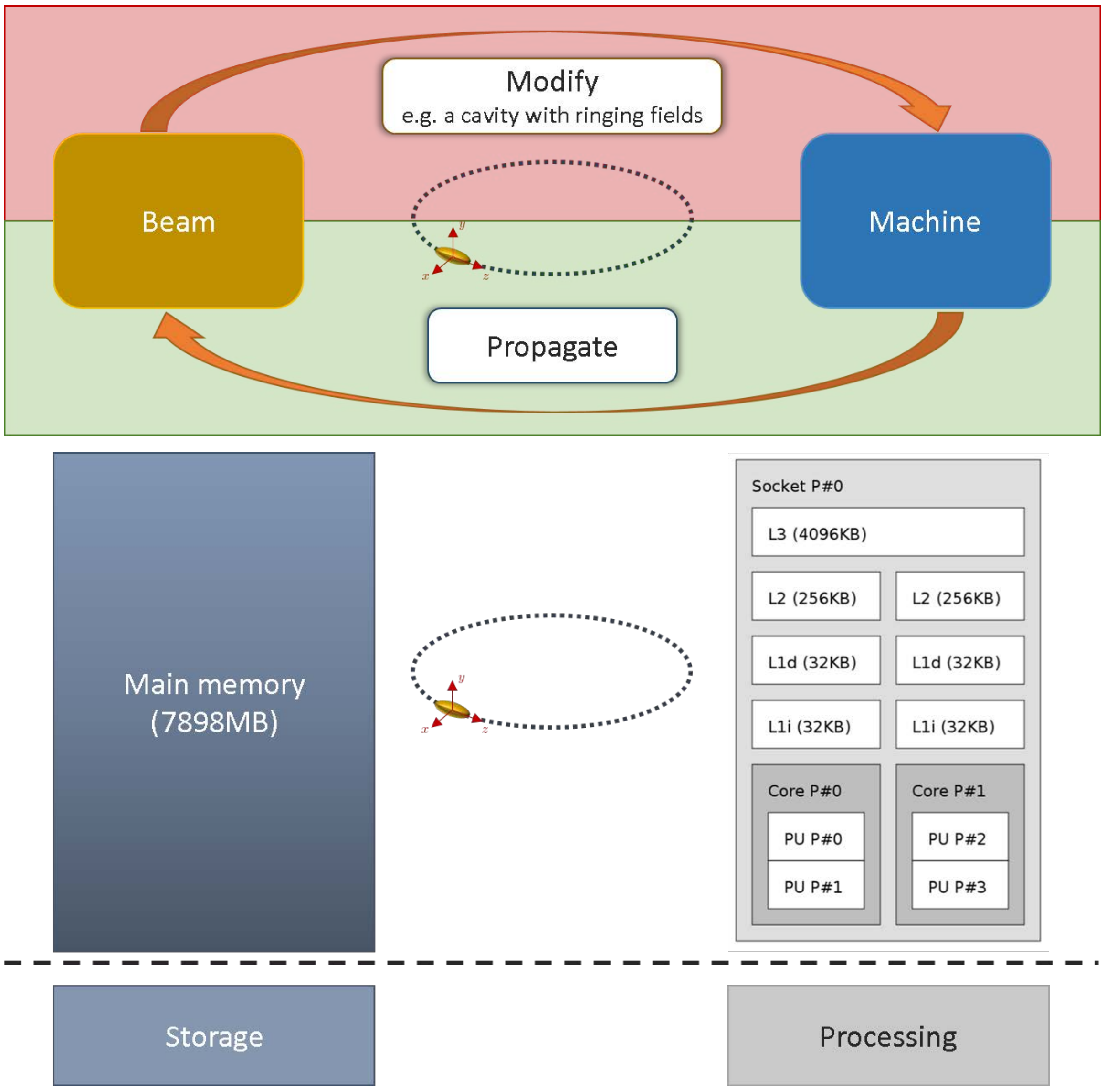


Fig. 2: The mapping of an accelerator–beam system onto the fundamental components of modern computer system hardware (display of the topological map of the system as obtained from *lstopo*).

1.2 Coordinate system and initialization

In a synchrotron the reference orbit curvature ρ and the reference momentum p_0 are related according to the synchronous condition via the dipole magnetic field B as

$$B\rho = \frac{p_0}{e}\,, \tag{1}$$

where e is the unit charge. In these lectures, we will choose generalized coordinates and canonically conjugate momenta to describe the dynamics of a charged particle system as illustrated in Fig. 3. These are, in the transverse plane, the horizontal and vertical positions x and y in metres with respect to the reference orbit of the machine and the respective angles x' and y'. In the longitudinal plane it is the position z in metres with respect to a position equivalent to the zero crossing of the RF fields (i.e., synchronous to an external RF clock) and the relative momentum offset δ with respect to the reference momentum. We name the phase space Γ; then this results in a set of six phase-space variables for every particle:

$$\left(\begin{pmatrix} x \\ x' \end{pmatrix}_i, \begin{pmatrix} y \\ y' \end{pmatrix}_i, \begin{pmatrix} z \\ \delta \end{pmatrix}_i\right) \in \Gamma, \quad i \in [1, \ldots, \text{particle number}]\,. \tag{2}$$

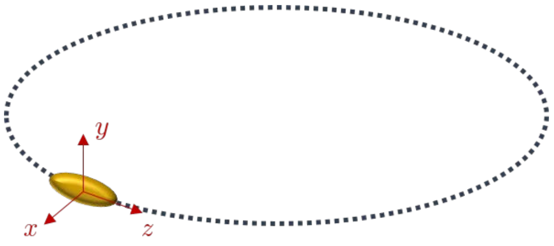


Fig. 3: The coordinate system used in these lectures

The first step in running a macroparticle simulation is to initialize a macroparticle system. For this we allocate a memory block in the main memory which is sufficiently large to accommodate all the relevant quantities of the macroparticle system, i.e., generalized coordinates and canonically conjugate momenta, charges and masses. Table 1 shows an example memory layout of a macroparticle system in the main memory where each of the six phase-space variables is allocated as an array of length equal to the macroparticle number which contains the corresponding values for every macroparticle. Then we fill this memory block randomly with numbers. Of course we do not do this entirely randomly. Typically, our macroparticle system will represent a particle bunch in a machine. As such, it should mimic the macroscopic statistical properties of the particle bunch such as the particle distribution function or the bunch size. A macroparticle system is therefore generated in a three-stage approach—first, we generate a distribution according to the particle distribution function, then we scale the distribution to match the bunch size and finally we distort the distribution without changing its r.m.s. size to match the local machine optics.

The particle bunch's intrinsic size is basically fully characterized by its emittance in the three planes, horizontal, vertical and longitudinal. The transverse and longitudinal emittances are defined as

$$\varepsilon_u = \gamma\beta\sqrt{\sigma_u^2\,\sigma_{u'}^2 - (\sigma_u\,\sigma_{u'})^2}\,, \quad u = x, y\,, \tag{3}$$

$$\varepsilon_z = 4\pi\frac{p_0}{e}\sqrt{\sigma_z^2\,\sigma_\delta^2 - (\sigma_z\sigma_\delta)^2}\,, \tag{4}$$

respectively, with

$$\sigma_b^2 = \left\langle (b - \langle b \rangle)^2 \right\rangle\,. \tag{5}$$

Table 1: Example memory layout of a macroparticle system in the main memory. Each of the six phase-space variables is allocated as an array of length equal to the macroparticle number N.

Count	0	1	2	$\dots$	N
x	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
x'	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
y	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
y'	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
z	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
δ	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

As a simple example we generate a Gaussian distribution in the horizontal plane with a given emittance. For this we simply need to create a Gaussian distribution using a Gaussian distribution random number generator, available in most standard libraries, in (x, x') with parameters [1]

$$\sigma_x = \sqrt{\frac{\varepsilon_x}{\gamma\beta}}, \tag{6}$$

$$\sigma_{x'} = \sqrt{\frac{\varepsilon_x}{\gamma\beta}}, \tag{7}$$

such that $\varepsilon_x = \gamma\beta\sigma_x\sigma_{x'}$. The resulting distribution is plotted in Fig. 4.

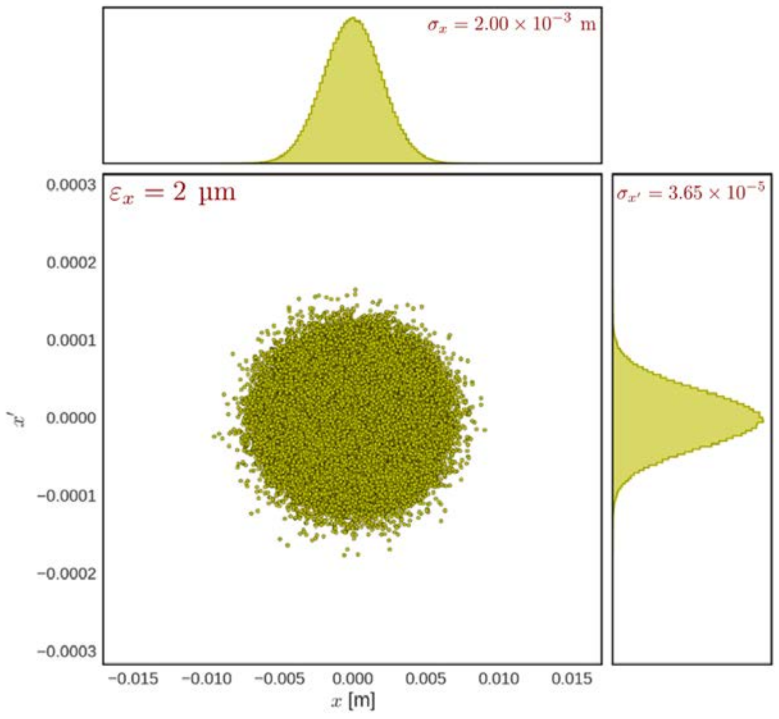


Fig. 4: Bi-Gaussian distribution in (x, x') with a normalized emittance of 2 µm

The distribution can now be matched to the local machine optics described by the optics functions (α_x, β_x) by scaling and correlating the coordinates and momenta according to

$$x_{\text{matched}} = \sqrt{\beta_x}\, x\,, \tag{8}$$

[1]The careful reader will realize an apparent mismatch in the dimensions in Eq. (8). In reality, this transformation emerges from the application of a matching section and there is another factor $\sqrt{1\ [\mathrm{m}]}$ involved, which solves the dimension mismatch.

$$x'_{\text{matched}} = \frac{1}{\sqrt{\beta_x}}\,x' - \frac{\alpha_x}{\sqrt{\beta_x}}\,x\,. \tag{9}$$

We will not go into further details of creating non-linearly matched distributions such as for an RF bucket here.

Once all numbers have been generated, we are left with a numerical representation of the particle bunch as a macroparticle system. This manifests in the main memory as six individual arrays each with the length equal to the number of macroparticles in the macroparticle system. We will assume the charge and mass of all macroparticles to be identical such that they can each be described by two single floating point numbers. The particle bunch is now fully described by $(6 \times \text{macroparticle number} + 2)$ floating point numbers.

So far, we are able to generate a numerical representation of a particle bunch as a macroparticle system residing in the main memory in six arrays and two floating point numbers. No dynamics has been included yet. For this, we need to provide functions that modify the entries in each array in accordance with the machine elements that these functions are supposed to represent. This will be discussed in the following sections.

2 Modelling beam dynamics in absence of collective effects

The goal in the previous section was to abstract a physical particle system such that it can be numerically modelled. We implemented this by moving to a macroparticle system and representing it in the main memory as a set of arrays and numbers. Now we need to propagate this macroparticle system through the machine. For this, we need to identify machine elements and assign to them actions that they are to apply to the macroparticle system. This will be implemented as functions that update the entries of each of the respective arrays in accordance with the machine element that this function is supposed to represent.

As a first step we need to formalize this action so that we understand the beam dynamics and can come up with an algorithm that can be implemented on a computer. We can work in the framework of classical mechanics where we know that the entire dynamics of a particle system is fully contained in a single function, the Hamiltonian H, which is a function of the generalized coordinates and canonically conjugate momenta $[(x, x'), (y, y'), (z, \delta)]$ and an independent variable s along which the evolution is generated:

$$\left(\begin{pmatrix} x \\ x' \end{pmatrix}, \begin{pmatrix} y \\ y' \end{pmatrix}, \begin{pmatrix} z \\ \delta \end{pmatrix}\right) \in \Gamma, \quad H(x, x', y, y', z, \delta; s) \in f : \Gamma \to \mathbb{R}. \tag{10}$$

This takes place according to the Hamilton equations of motion

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}s} &= \frac{\partial H}{\partial x'}, & \frac{\mathrm{d}x'}{\mathrm{d}s} &= -\frac{\partial H}{\partial x}, \\ \frac{\mathrm{d}y}{\mathrm{d}s} &= \frac{\partial H}{\partial y'}, & \frac{\mathrm{d}y'}{\mathrm{d}s} &= -\frac{\partial H}{\partial y}, \\ \frac{\mathrm{d}z}{\mathrm{d}s} &= \frac{\partial H}{\partial \delta}, & \frac{\mathrm{d}\delta}{\mathrm{d}s} &= -\frac{\partial H}{\partial z}. \end{aligned} \tag{11}$$

We will now treat two simple Hamiltonians that generate betatron and synchrotron motion, which will allow us to propagate our macroparticle system along a circular accelerator in the absence of collective effects.

2.1 Transverse tracking

The basic elements in a circular accelerator that control the transverse motion are the dipoles to keep the beam on a circular orbit and the quadrupoles that provide focusing for off-orbit particles. The orbit defined by the dipoles is given by the synchronous condition in Eq. (1). The focusing of the quadrupoles is described by the Hamiltonian

$$H(x,x') = \frac{1}{2}x'^2 + \frac{1}{2}K(s)x^2\,, \quad K(s) = K(s+C)\,. \tag{12}$$

The corresponding equation of motion is Hill's equation given as

$$x'' + K(s)x^2 = 0\,. \tag{13}$$

Equation (13) has the complication that it contains an s-dependent coefficient $K(s)$. Fortunately, this coefficient comes with the periodicity condition given in Eq. (12), which makes Eq. (13) actually solvable. An excellent treatment of this problem together with non-linear terms can be found in Ref. [7]. Here, we will focus on solving the linear problem.

The solution of any linear second order differential equation of the form (13) is uniquely determined by the initial values of x and its derivative x':

$$\begin{aligned} x(s) &= a\,x(s_0) + b\,x'(s_0), \\ x'(s) &= c\,x(s_0) + d\,x'(s_0). \end{aligned} \tag{14}$$

In matrix notation, this can be written as

$$\begin{pmatrix} x \\ x' \end{pmatrix}\Bigg|_s = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ x' \end{pmatrix}\Bigg|_{s_0} = M(s|s_0) \begin{pmatrix} x \\ x' \end{pmatrix}\Bigg|_{s_0}. \tag{15}$$

The matrix formulation is useful because it separates the properties of the general solution from those due to a specific initial condition. The matrix depends only on $K(s)$ and the length of the interval $(s - s_0)$. In addition, the matrix for any interval made up of subintervals is just the product of the matrices for the subintervals, that is,

$$M(s_2|s_0) = M(s_2|s_1)M(s_1|s_0). \tag{16}$$

Using the transfer matrix method together with the conditions of periodicity, stability and boundedness, one discovers that the eigenvalues of M are given as

$$\lambda = \exp\left(\pm \mathrm{i}\mu\right). \tag{17}$$

The matrix M satisfies (by construction, as it ultimately emerged from the Hamiltonian in Eq. (12))

$$M^T J M = J, \tag{18}$$

where J is the symplectic structure matrix

$$J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \tag{19}$$

Hence, the matrix M may be written in a form which exhibits the eigenvalues explicitly:

$$M = I\cos(\mu) + JA\sin(\mu), \tag{20}$$

where I is the identity matrix and A is a symmetric parameterized matrix

$$A = \begin{pmatrix} \gamma & \alpha \\ \alpha & \beta \end{pmatrix}. \tag{21}$$

The parameters (α, β, γ) are the Courant–Snyder parameters [8] and correspond to the optics functions. They are s-dependent in general and play a major role in determining the details of the motion. In particular, β determines the maximum local amplitude of transverse oscillations.

In applying Floquet's theorem, we insert a solution

$$x(s) = w(s) \exp\left(\mathrm{i}\psi(s)\right) \equiv \sqrt{2J_x\beta_x(s)} \cos(\psi_x(s) + \psi_{x,0}), \tag{22}$$

which again exhibits the eigenvalues of M explicitly, into (13) and find that the transfer matrix that propagates the coordinates and momenta (x, x') from one point s_0 with local optics function $(\alpha_0, \beta_0, \gamma_0)$ to another point s_1 with local optics function $(\alpha_1, \beta_1, \gamma_1)$ in the ring is written as

$$M(s_1|s_0) = \begin{pmatrix} \sqrt{\beta_1} & 0 \\ -\frac{\alpha_1}{\sqrt{\beta_1}} & \frac{1}{\sqrt{\beta_1}} \end{pmatrix} \begin{pmatrix} \cos(\Delta\mu_{0\to1}) & \sin(\Delta\mu_{0\to1}) \\ -\sin(\Delta\mu_{0\to1}) & \cos(\Delta\mu_{0\to1}) \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{\beta_0}} & 0 \\ \frac{\alpha_0}{\sqrt{\beta_0}} & \sqrt{\beta_0} \end{pmatrix}. \tag{23}$$

Coming back to our original problem of numerically implementing linear transverse (betatron) motion, we can now see that this reduces basically to implementing a simple matrix multiplication function. The betatron motion is fully characterized by the machine optics functions. These need to be provided as an external input, either explicitly as a table obtained from an optics calculation (e.g. MAD-X) or, which is often done for simplification, using the smooth approximation where

$$\alpha = 0, \quad \beta = \frac{R}{Q} = \text{constant}. \tag{24}$$

Here, R is the ring radius and Q is the tune defined as the number of betatron oscillations per ring revolution:

$$Q = \frac{1}{2\pi} \oint \mu(s)\, \mathrm{d}s = \frac{1}{2\pi} \oint \frac{\mathrm{d}s}{\beta(s)}. \tag{25}$$

Once the optics functions are known, we can split the ring into segments and build the set of transfer matrices $M(s_j|s_i)$ that propagate macroparticles through subsequent segments (s_i, s_j) along the ring, as indicated for the two points s_0 and s_1 in Fig. 5. The numerical propagation of the macroparticle system through the machine is then performed by executing

$$\begin{aligned} \left.\begin{pmatrix} x_k \\ x'_k \end{pmatrix}\right|_{s_{i+1}} &= M_x(s_{i+1}|s_i) \left.\begin{pmatrix} x_k \\ x'_k \end{pmatrix}\right|_{s_i}, \\ \left.\begin{pmatrix} y_k \\ y'_k \end{pmatrix}\right|_{s_{i+1}} &= M_y(s_{i+1}|s_i) \left.\begin{pmatrix} y_k \\ y'_k \end{pmatrix}\right|_{s_i}, \\ i &\in [0, \dots, \text{segment number} - 1], \\ k &\in [0, \dots, \text{macroparticle number} - 1], \end{aligned} \tag{26}$$

repetitively for every segment and for every macroparticle in both the horizontal and the vertical planes.

In implementing linear, periodic and uncoupled betatron motion we simply need to generate and store the transfer matrices for every segment in the horizontal and the vertical planes, which amounts to a total of $(4 \times \text{segment number} \times 2)$ constant floating point numbers. The betatron propagation of the macroparticle system is computationally cheap and fast.

2.2 Longitudinal tracking

Longitudinal motion is a little more involved than the linear betatron motion described in the previous subsection. Longitudinal motion in circular accelerators, or synchrotron motion, has some peculiarities such as the presence of transition or the intrinsic non-linearities important particularly for hadron

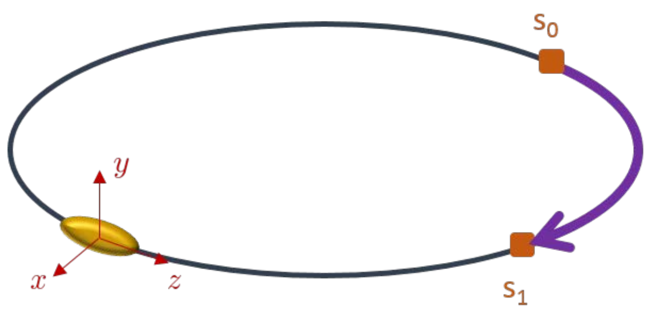


Fig. 5: A segment from a point s_0 to another point s_1 along the ring for which the betatron motion is described by the transfer matrix in Eq. (23).

machines where bunch lengths are comparable to the RF period of the accelerating cavities. A good introduction into the longitudinal dynamics can be found in Ref. [9].

Transition is a phenomenon that occurs in circular accelerators as a result of two competing effects. At low energies the relation between energy and momentum is nearly linear such that particles with slightly higher energy travel faster and hence for these particles the revolution period decreases, as one would naively expect. For high energies another effect becomes important, which results from the momentum compaction present in any circular machine with dispersion. What happens is that particles at high energies no longer gain in speed. However, due to dispersion along the machine they take longer paths and hence for these particles the revolution period increases. This effect is governed by the momentum compaction factor of the machine

$$\alpha_\text{c} = \frac{1}{C}\oint \frac{D(s)}{\rho}\,\mathrm{d}s\,, \tag{27}$$

where C is the machine circumference, $D(s)$ the dispersion function and ρ the orbit curvature. The energy at which this behaviour switches, from lower to higher revolution periods with an increase in energy, is called the transition energy, defined by

$$\gamma_\text{transition} = \frac{1}{\sqrt{\alpha_\text{c}}}\,. \tag{28}$$

At transition itself all synchrotron motion is frozen.

Since a particle bunch in a circular machine naturally has a spread in energy it consequently also has a spread in revolution period. If simply left to drift, the particle bunch will eventually de-bunch along the machine. What is needed to keep particles bunched is some mechanism of focusing similar to the one that is provided in the transverse plane by the quadrupoles. This is accomplished in the longitudinal plane by the RF cavities. At each revolution, particles experience a kick from the integrated fields in the RF cavities. Depending on the delay of particles arriving at the RF cavities, they will gain more or less energy from the RF cavities. In that way the RF cavities keep the revolution periods close and thus provide focusing around what is called the synchronous phase φ_s.

Finally, the RF cavities also provide the necessary energy for acceleration. When ramping up the magnetic dipole fields, the revolution period of the entire particle bunch changes resulting in a shift of the synchronous phase. This leads to a net energy gain of the entire particle bunch and thus to acceleration.

The synchrotron motion together with all the resulting effects mentioned above are described by the Hamiltonian

$$H = -\frac{1}{2}\eta\beta c\,\delta^2 + \frac{e}{p_0 C}V_\text{RF}(z)\,, \tag{29}$$

and V_{RF} usually takes the form

$$V_{\mathrm{RF}} = \sum_i \frac{V_i R}{h_i} \left(\frac{h_i z}{R} + \varphi_i \right) + \frac{\Delta E}{e} z \,. \tag{30}$$

Here, η is the slippage factor

$$\eta = \frac{1}{\gamma_{\mathrm{tr}}^2} - \frac{1}{\gamma^2} \,, \tag{31}$$

which we will discuss further below, $\beta = v/c$ is the velocity in units of the speed of light, p_0 is the reference momentum (1) and C is the ring circumference. Also, V is the integrated voltage along the RF cavities (this includes the transit-time factor), $h = \omega_{\mathrm{RF}}/\omega_0$ the harmonic number, which is the RF frequency in units of the revolution frequency, and ΔE the mean energy gain per turn.

If we assume a single harmonic RF system, from Eqs. (29) and (30) together with Eq. (11) the longitudinal equations of motion are derived as

$$\frac{\mathrm{d}z}{\mathrm{d}s} = -\eta\beta c\,\delta, \tag{32}$$

$$\frac{\mathrm{d}\delta}{\mathrm{d}s} = \frac{eV}{p_0 C} \left(\sin\left(\frac{hz}{R} \right) - \frac{\Delta E}{eV} \right) . \tag{33}$$

Equation (32) contains the slippage factor from Eq. (31) and exhibits the phenomenon of transition. Particles below the transition energy ($\eta < 0$) with positive momentum offset have a positive velocity, whereas above the transition energy ($\eta > 0$) a positive momentum offset leads to a negative velocity with respect to the direction of flight.

For small phases close to the synchronous phase, we may linearize the kick in Eq. (33) around the synchronous phase φ_{s} such that we obtain the linearized longitudinal equations of motion [2]

$$z'' + \underbrace{\frac{eV\eta h}{p_0 \beta c C R} \cos(\varphi_{\mathrm{s}})}_{\left(\frac{\omega_{\mathrm{s}}}{\beta c}\right)^2} \, z = 0 \,. \tag{34}$$

From Eq. (34), we identify the synchrotron tune as

$$Q_{\mathrm{s}} = \frac{\omega_{\mathrm{s}}}{\omega_0} = \sqrt{\frac{eV\eta h}{2\pi E_0 \beta^2} \cos(\varphi_{\mathrm{s}})} \,. \tag{35}$$

Defining

$$\alpha_{\mathrm{s}} = 0 \,, \quad \beta_z = \frac{|\eta| R}{Q_{\mathrm{s}}} = \text{constant} \,, \tag{36}$$

we can apply the same concept as for the linear betatron motion in the transverse plane, obtaining the one-turn transfer matrix

$$M = \begin{pmatrix} \sqrt{\beta_z} & 0 \\ 0 & \frac{1}{\sqrt{\beta_z}} \end{pmatrix} \begin{pmatrix} \cos(2\pi Q_{\mathrm{s}}) & \sin(2\pi Q_{\mathrm{s}}) \\ -\sin(2\pi Q_{\mathrm{s}}) & \cos(2\pi Q_{\mathrm{s}}) \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{\beta_z}} & 0 \\ 0 & \sqrt{\beta_z} \end{pmatrix} , \tag{37}$$

and propagating the longitudinal phase-space variables (z, δ) of the macroparticle system by applying a simple matrix multiplication as in Eq. (26).

For modelling the general non-linear longitudinal motion we need to solve the system of equations (32) and (33) via a numerical integration algorithm. There are several numerical integration schemes

[2]Note that via this linearization, z is now the distance from the synchronous phase.

available (e.g., Runge–Kutta methods). In order to respect the Hamiltonian nature of the problem, it is important that we choose a symplectic integrator. We will not go into the details of symplectic integrators here but rather refer the reader to Refs. [10, 11]. An example of a reasonably simple and fast symplectic integrator of second order in the time step is the velocity-Verlet algorithm

$$\begin{aligned} z_{k,i+1/2} &= z_{k,i} - \frac{\eta C}{2}\,\delta_{k,i}, \\ \delta_{k,i+1} &= \delta_{k,i} + \frac{eV_{\mathrm{RF}}}{m\gamma\beta^2c^2}\sin\left(\frac{2\pi h}{C}\,z_{k,i+1/2}\right), \\ z_{k,i+1} &= z_{k,i+1/2} - \frac{\eta C}{2}\,\delta_{k,i+1}, \\ i &\in [0,\ldots,\text{turn number}-1], \\ k &\in [0,\ldots,\text{macroparticle number}-1]. \end{aligned} \tag{38}$$

The integration of the longitudinal equations of motion (32) and (33) for modelling the synchrotron motion is then applied once per turn for the full revolution as depicted in Fig. 6.

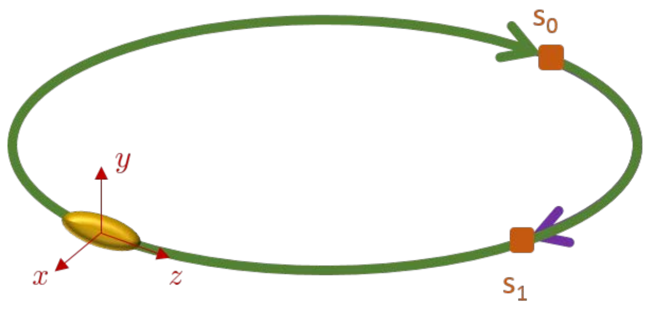


Fig. 6: The integration for the synchrotron motion according to Eqs. (38) is applied once per turn for the full ring

Once this algorithm has been implemented as a function, the numerical propagation of the macroparticle system through the machine is then performed by an iterative turn-by-turn application of Eqs. (38) for every macroparticle.

Figure 7 shows a simulation of a macroparticle bunch which is propagated through a ring applying both betatron and synchrotron motion. The horizontal (top) and longitudinal (bottom) phase spaces are shown. The non-linearity of the synchrotron motion leads to different tunes for different macroparticles depending on their longitudinal action. As a result, a filamentation of the longitudinal phase space takes place, which can be observed as the originally aligned colours spiral around the synchronous phase.

2.3 Modelling chromaticity and detuning with amplitude

We are now able to advance a macroparticle system through the machine by transverse and longitudinal tracking of the individual single macroparticles. The transverse tracking implemented so far models the linear betatron motion. For collective effects certain aspects of the non-linear transverse motion also have an important impact. These can appear in the form of chromaticity or detuning with amplitude, for example.

Chromaticity leads to a shift of the coherent beam spectrum and modifies the way the beam interacts with the machine impedance. This significantly impacts the behaviour of head–tail modes and instabilities as well as the transverse mode coupling instability (TMCI) (see Ref. [12]). Chromaticity results from different focusing of off-momentum particles. Thus, the quadrupole magnets, needed to provide the betatron focusing, introduce a natural chromaticity in the machine. Sextupole magnets

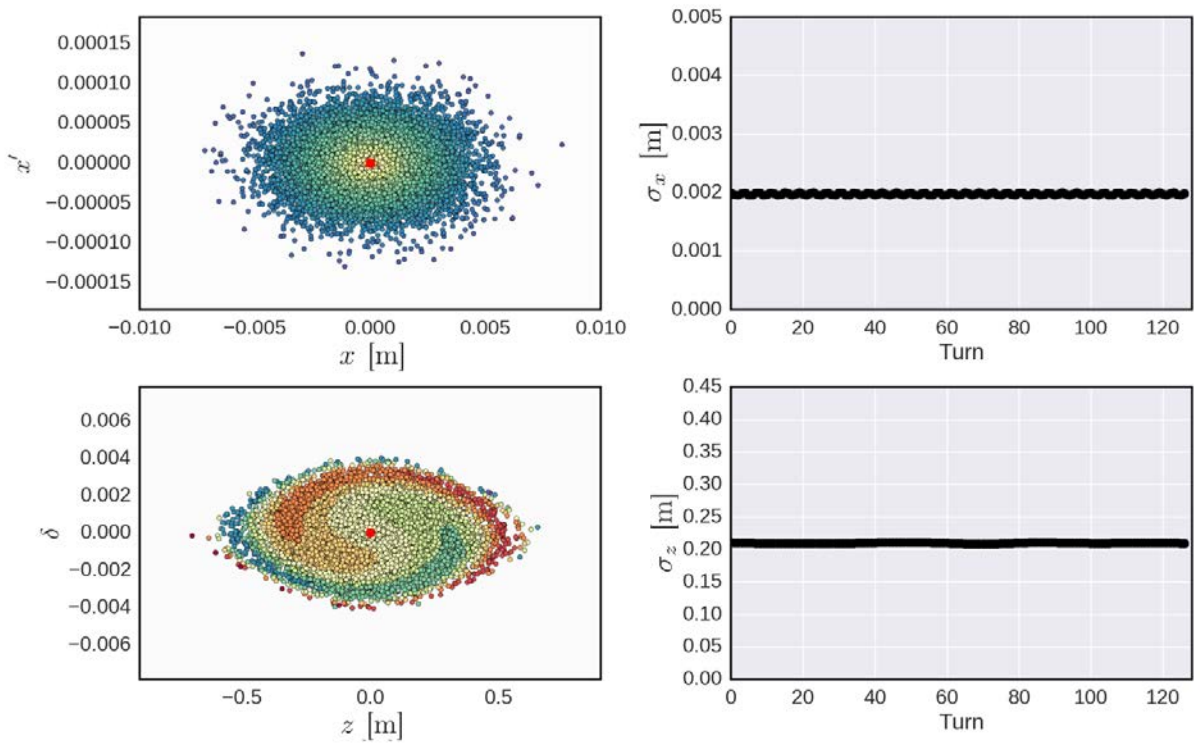


Fig. 7: A macroparticle bunch tracked through a ring applying betatron and synchrotron motion. The figure shows the horizontal (top left) and longitudinal (bottom left) phase spaces after several turns. Filamentation of the longitudinal phase space is clearly observable as the originally aligned colours spiral around the synchronous phase. The horizontal and longitudinal bunch sizes remain constant, as can be seen in the plots on the right, indicating that the bunch was well matched upon initialization to the machine optics.

placed in dispersive regions can then be used to control the chromaticity. Chromaticity is defined as

$$Q' = \frac{\mathrm{d}(\Delta Q)}{\mathrm{d}\delta}, \tag{39}$$

and can be easily calculated by writing down the (non-linear) Hamiltonian in action–angle variables as done in Ref. [7], for example. It may be written as

$$\begin{aligned} Q'_x &= -\frac{1}{4\pi}\oint \beta_x(s)\,(K_1(s) - K_2(s)D(s))\;\mathrm{d}s, \\ Q'_y &= +\frac{1}{4\pi}\oint \beta_y(s)\,(K_1(s) - K_2(s)D(s))\;\mathrm{d}s. \end{aligned} \tag{40}$$

Here, $K_i = B_i/(B\rho)$ are the effective (normalized) magnetic field strengths of the quadrupole $(i = 1)$ and sextupole $(i = 2)$ magnets, respectively.

Detuning with amplitude leads to a transverse tune spread within the beam, which can result in the suppression of instabilities via Landau damping. This is achieved for example with octupole magnets which provide a different focusing depending on the transverse offset of particles from the centre of the octupoles (which are usually placed on the reference orbit). Detuning with amplitude is described by the anharmonicities of the machine, which are defined as

$$\begin{pmatrix} \Delta Q_x \\ \Delta Q_y \end{pmatrix} = \begin{pmatrix} \alpha_{xx} & \alpha_{xy} \\ \alpha_{yx} & \alpha_{yy} \end{pmatrix} \begin{pmatrix} J_x \\ J_y \end{pmatrix}, \tag{41}$$

or consequently

$$\begin{aligned} \alpha_{xx} &= \frac{\mathrm{d}\Delta Q_x}{\mathrm{d}J_x}, \qquad \alpha_{xy} = \frac{\mathrm{d}\Delta Q_x}{\mathrm{d}J_y}, \\ \alpha_{yy} &= \frac{\mathrm{d}\Delta Q_y}{\mathrm{d}J_y}, \qquad \alpha_{yx} = \frac{\mathrm{d}\Delta Q_y}{\mathrm{d}J_x} \end{aligned} . \tag{42}$$

As for chromaticity, the anharmonicities can be easily calculated by writing down the (non-linear) Hamiltonian, now also containing octupole fields, in action–angle variables. They may be written as

$$\begin{aligned} \alpha_{xx} &= +\frac{3}{8\pi} \oint K_3(s)\beta_x(s)^2 \,\mathrm{d}s, \\ \alpha_{xy} &= -\frac{3}{4\pi} \oint K_3(s)\beta_x(s)\beta_y(s) \,\mathrm{d}s = \alpha_{yx}, \\ \alpha_{yy} &= +\frac{3}{8\pi} \oint K_3(s)\beta_y(s)^2 \,\mathrm{d}s. \end{aligned} \tag{43}$$

Here, $K_3 = B_3/(B\rho)$ is the effective (normalized) octupole field strength.

Chromaticity and detuning with amplitude can be implemented rather easily if we recollect that for a certain particle with momentum offset δ and transverse actions J_x and J_y, the detuning resulting from chromaticity and anharmonicities of the machine integrated over one turn is calculated as

$$\begin{aligned} \Delta Q_x &= Q'_x\delta + \alpha_{xx}J_x + \alpha_{xy}J_y \,, \\ \Delta Q_y &= Q'_y\delta + \alpha_{yx}J_x + \alpha_{yy}J_y \,. \end{aligned} \tag{44}$$

Hence, for a given segment (i, j) along the ring, the proportional detuning becomes

$$\begin{aligned} \Delta Q_{x,i\to j} &= \left(Q'_x\delta + \alpha_{xx}J_x + \alpha_{xy}J_y\right) \frac{\Delta\mu_{x,i\to j}}{2\pi\, Q_x}, \\ \Delta Q_{y,i\to j} &= \left(Q'_y\delta + \alpha_{yx}J_x + \alpha_{yy}J_y\right) \frac{\Delta\mu_{y,i\to j}}{2\pi\, Q_y}. \end{aligned} \tag{45}$$

We can pick up the transfer matrix in Eq. (23), which, when we include detuning effects from chromaticity and detuning with amplitude, becomes a separate transfer matrix for every individual single macroparticle:

$$\begin{aligned} M_k(s_1|s_0) &= \begin{pmatrix} \sqrt{\beta_1} & 0 \\ -\frac{\alpha_1}{\sqrt{\beta_1}} & \frac{1}{\sqrt{\beta_1}} \end{pmatrix} \begin{pmatrix} \cos(\Delta\mu_{k,0\to 1}) & \sin(\Delta\mu_{k,0\to 1}) \\ -\sin(\Delta\mu_{k,0\to 1}) & \cos(\Delta\mu_{k,0\to 1}) \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{\beta_0}} & 0 \\ \frac{\alpha_0}{\sqrt{\beta_0}} & \sqrt{\beta_0} \end{pmatrix}, \\ \Delta\mu_{k,0\to 1} &= \Delta\mu_{k,0\to 1}(\delta_k, J_k) = \left(Q'\delta_k + \alpha J_k\right) \frac{\Delta\mu_{i\to j}}{Q}, \\ & k \in [0, \ldots, \text{macroparticle number} - 1]. \end{aligned} \tag{46}$$

Chromaticity and detuning with amplitude already lead to some macroscopic effects which are not yet collective effects (the potential does not depend on the particle distribution) but are nevertheless distinct for multiparticle systems. Examples are the decoherence from chromaticity or emittance blow-up due to filamentation from detuning with amplitude. These effects can be easily simulated with the tools we have implemented so far. An example of a simulation of decoherence and emittance blow-up in the presence of anharmonicities is shown in Fig. 8. It can be clearly seen how macroparticles with larger transverse actions have a smaller tune and thus start lagging behind in transverse phase space. This results in the characteristic spiralling of the macroparticle distribution in phase space, which translates into the observed decoherence and emittance blow-up. From the signal of the mean position in Fig. 8, we can compute the coherent spectrum, which is plotted in Fig. 9. The spectrum clearly features a finite spread around the tune, which in this case was at 20.13. Because we are running a macroparticle simulation we have full access to all six phase-space variables of every macroparticle at any time. Equipped with the data for each individual macroparticle we are now even able to produce the incoherent spectrum. For this we perform a fast Fourier transform (FFT) analysis for each macroparticle to obtain their individual tunes in the horizontal and the vertical planes. We plot the obtained points in tune space (Q_x, Q_y) for every macroparticle to obtain the tune footprint of the macroparticle system. This is shown in Fig. 10. The tune footprint is an essential piece of information for the characterization of Landau damping, which will not be treated in further detail here. The interested reader is referred to Refs. [13–15].

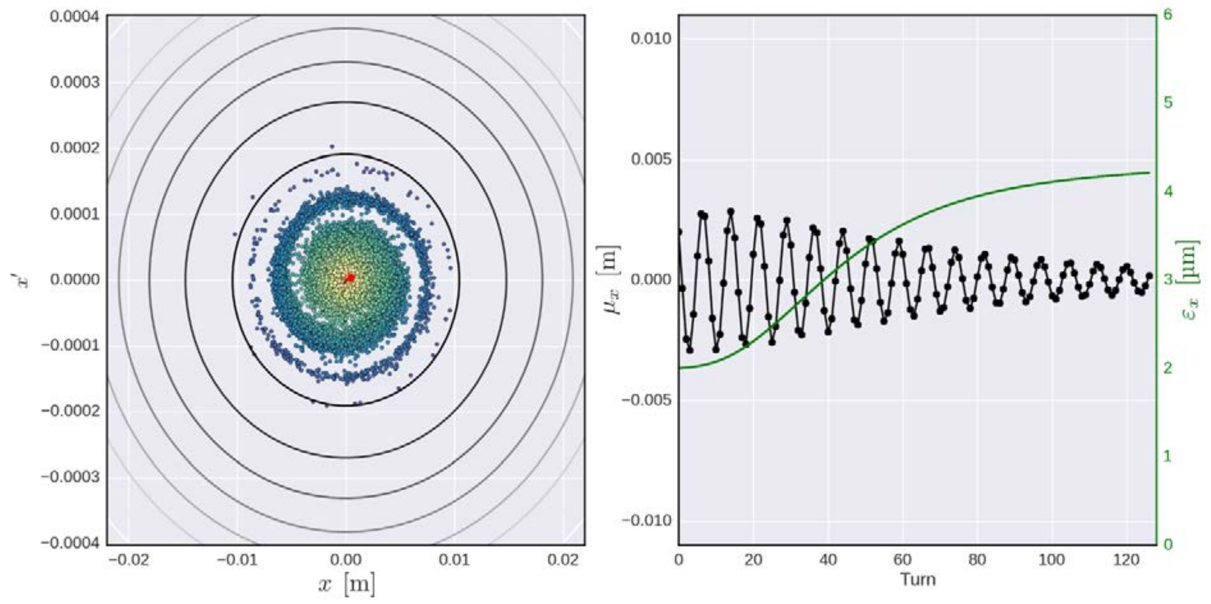


Fig. 8: Filamentation in the transverse phase space as a result of anharmonicities when the bunch enters with an initial offset (left). The mean position decoheres accompanied by an emittance blow-up, as can be seen in the plot on the right.

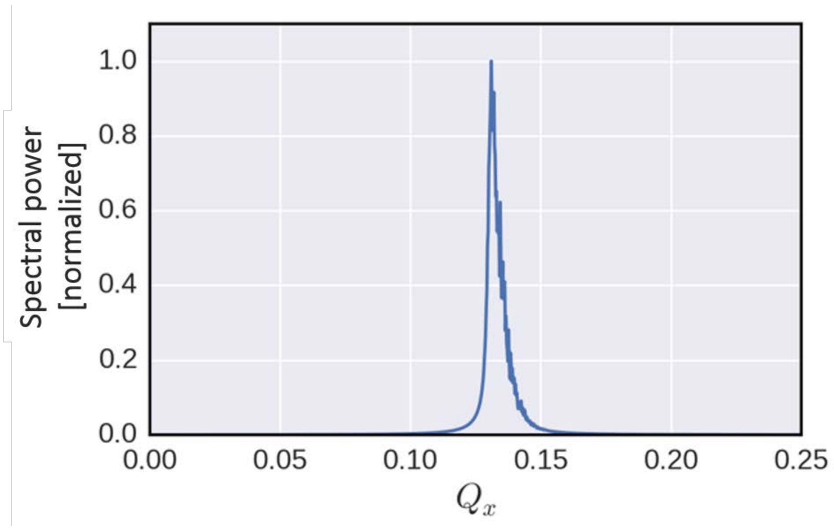


Fig. 9: The coherent spectrum obtained from the signal of the mean position. The spectrum features a finite spread around the tune (20.13).

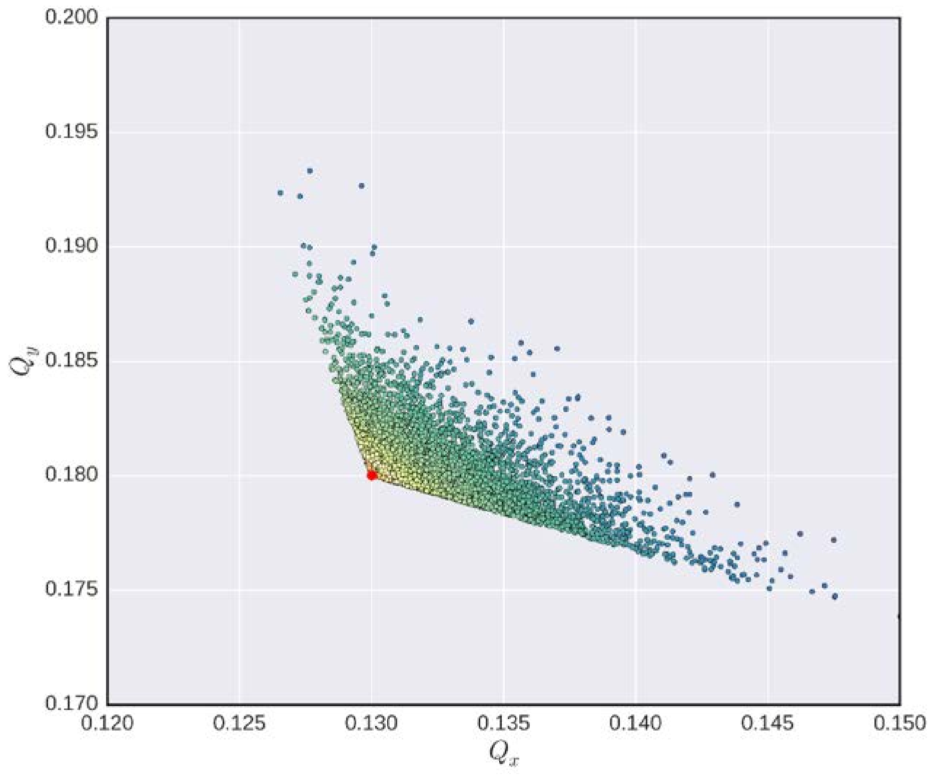


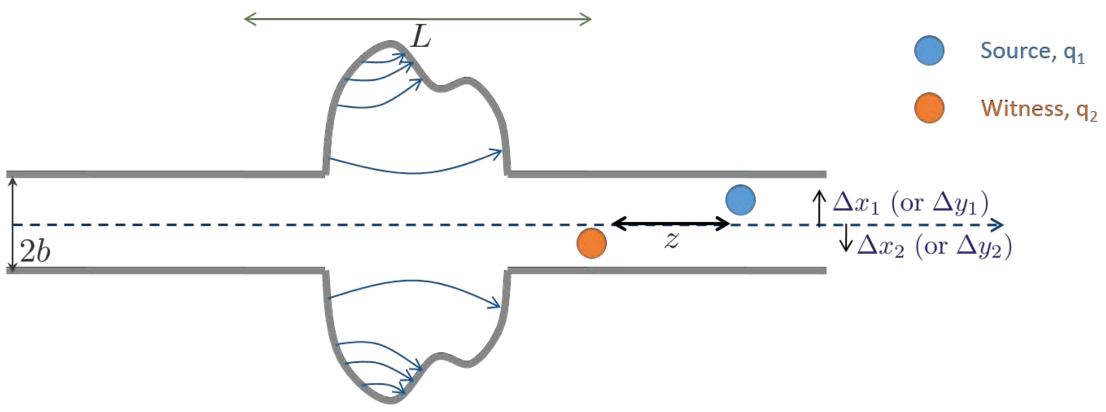
Fig. 10: The incoherent (fractional tune) spectrum obtained from the signal of each macroparticle. The nominal tune in this case was at $(20.13, 20.18)$. The darker points correspond to macroparticles with larger actions. The triangular shape of the tune footprint is characteristic for the detuning generated by octupoles, which were used for this simulation.

3 Modelling beam dynamics in the presence of wake fields

We have learned how to initialize a macroparticle system in accordance with the macroscopic statistical properties of a physical particle bunch and how to track this macroparticle system both transversely and longitudinally through a circular machine in absence of collective effects. So far, all propagation of the macroparticle system took place independently, i.e., macroparticles were tracked individually and independently from other macroparticles. Adding collective effects, tracking can no longer be performed in this manner. Collective effects add dependences to the macroparticle system such that macroparticles have to be tracked taking into account all other macroparticles present in the macroparticle system.

Physically, collective effects originate from machine elements and devices that store electromagnetic fields which have been excited by particles travelling through these elements or devices. The net electromagnetic field excited will depend on the exact distribution of the particles travelling through an element or device. Examples of these types of collective effects are the resistive wall effect coming from the resistivity of the walls of a vacuum chamber or trapped modes excited in a cavity-like structure. In all cases, particles passing through these objects leave back electromagnetic fields which then affect trailing particles. To model these effects one employs the concept of wake fields (or impedances).

Wake fields are the electromagnetic fields left behind a source particle travelling through an object and felt by a trailing or target particle. A such, they are functions of the distance between the source and the target particle. The concept of wake fields is illustrated in Fig. 11. Formally, a wake field is the electromagnetic response function of an object. This can be computed independently for any individual object either by analytical means for simple objects or by numerical means for more sophisticated structures. The wake function is an intrinsic property of any such object.



L
Source, q_1
Witness, q_2
Δx_1 (or Δy_1)
Δx_2 (or Δy_2)
z
$2b$

Fig. 11: The concept of wake fields. Wake fields are the integrated force from the electromagnetic fields left by a source particle (blue) and felt by a trailing or target particle (orange).

In the formalism to follow, we will restrict ourselves to the horizontal plane, but it is equally valid for both the horizontal and the vertical planes. Both planes must be included of course when one wishes to add coupling effects. The wake function w is defined via the integrated force along the effective length of a given object:

$$\int_{-L/2}^{L/2} F(z,s)\,\mathrm{d}s = qq_{\mathrm{t}}w(z). \tag{47}$$

Here, q and q_{t} are the charges of the source and target particles, respectively. The wake function in general is a function of the transverse positions of both the source and the target particles as well as of the distance between the two, so that

$$w(z) \to w(x, x_{\mathrm{s}}, z - z_{\mathrm{s}}). \tag{48}$$

Here, x and z denote the positions of the target particle. With the linearity of Maxwell's equations in the fields, the superposition principle holds and the integrated force generated not by a single particle but by a particle distribution travelling through an object is given as the convolution

$$\begin{aligned}\int_{-L/2}^{L/2} F(x,z,s)\,\mathrm{d}s &= qq_{\mathrm{t}} \int \rho(x_{\mathrm{s}}, z_{\mathrm{s}})\, w(x, x_{\mathrm{s}}, z - z_{\mathrm{s}})\,\mathrm{d}x_{\mathrm{s}}\,\mathrm{d}z_{\mathrm{s}} \\ &\equiv -\nabla V(x,z).\end{aligned} \tag{49}$$

We are now ready to write down the Hamiltonian for the simple betatron motion together with wake fields:

$$H = \frac{1}{2}x'^2 + \frac{1}{2}K(s)x^2 + \sum_k \frac{e^2}{m\gamma\beta^2c^2\,C} \iint \rho(x_{\mathrm{s}}, z_{\mathrm{s}})\, w(x, x_{\mathrm{s}}, z - z_{\mathrm{s}} - kC)\,\mathrm{d}x_{\mathrm{s}}\,\mathrm{d}z_{\mathrm{s}} + H_s(z,\delta) \tag{50a}$$

$$= \ldots + \sum_k \frac{e^2}{m\gamma\beta^2c^2\,C} \iint \rho(x_{\mathrm{s}}, z_{\mathrm{s}}) \sum_{mn} x^n x_{\mathrm{s}}^m\, W_{mn}(z - z_{\mathrm{s}} - kC)\,\mathrm{d}x_{\mathrm{s}}\,\mathrm{d}z_{\mathrm{s}} + H_{\mathrm{s}}(z,\delta) \tag{50b}$$

$$= \ldots + \sum_k \frac{e^2}{m\gamma\beta^2c^2\,C} \sum_{mn} x^n \int \lambda_m(z_{\mathrm{s}})\, W_{mn}(z - z_{\mathrm{s}} - kC)\,\mathrm{d}z_{\mathrm{s}} + H_{\mathrm{s}}(z,\delta), \tag{50c}$$

where

$$\lambda_m(z_{\mathrm{s}}) = \int \rho(x_{\mathrm{s}}, z_{\mathrm{s}}) x_{\mathrm{s}}^m\,\mathrm{d}x_{\mathrm{s}} \tag{51}$$

is the transverse moment of order m of the source distribution at z_s. We included multiturn wakes in the Hamiltonian above, where k is the turn number and C the ring circumference. Since the wake function is a function of the longitudinal coordinates we of course needed to include the longitudinal dynamics via the Hamiltonian $H_\mathrm{s}(z, \delta)$ to obtain a complete and self-consistent picture. In Eq. (50b), we made a Taylor expansion around the reference orbit and obtained the wake function of order mn. Here, m gives the order of the transverse moment of the source distribution and n gives the order of the resulting kick from the wake applied to the target particles. For most objects in practice, the wake functions that play a role are

$$\begin{aligned} W_{01} &: \text{constant wake,} \\ W_{11} &: \text{dipole (driving) wake,} \\ W_{02} &: \text{quadrupole (detuning) wake.} \end{aligned} \tag{52}$$

For these wakes, and omitting multiturn effects, the Hamiltonian simplifies to

$$H = \frac{1}{2}x'^2 \tag{53a}$$

$$+ A\left(\int [\rho(z_\mathrm{s})]\, W_{01}(z - z_\mathrm{s})\,\mathrm{d}z_\mathrm{s} + \int [\rho(z_\mathrm{s})\,\langle x_\mathrm{s}\rangle\,(z_\mathrm{s})]\, W_{11}(z - z_\mathrm{s})\,\mathrm{d}z_\mathrm{s}\right) x \tag{53b}$$

$$+ \left(\frac{1}{2}K(s) + A\int [\rho(z_\mathrm{s})]\, W_{02}(z - z_\mathrm{s})\right) x^2 + H_\mathrm{s}(z, \delta), \tag{53c}$$

where we set $A = e^2/(m\gamma\beta^2c^2C)$ and $\langle x_\mathrm{s}\rangle\,(z_\mathrm{s})$ is the mean horizontal position of the source distribution located at z_s. From this set of equations we can readily identify the effect of the different types of wakes on the beam dynamics. The term (53b) is linear in x and thus describes dipolar kicks changing the particle orbit. We can see that a constant wake produces a change of orbit depending on the source charge, whereas a dipolar wake creates an orbit distortion that depends on the mean source orbit offset. This feature of the dipolar wake is actually what enables it to drive beam instabilities, which is why it is sometimes also called the driving wake. The term (53c) produces a change in tune depending on the source charge. For this reason quadrupolar wakes are sometimes also called detuning wakes. We will discuss the different wakes and their impact on beam dynamics further below in parallel with simulations. But first we will move to the numerical implementation of wake field effects.

Going back to the definition of the wake fields in Eq. (49), we see that for a macroparticle system, which is essentially a collection of discrete individual macroparticles, the wake field kick, i.e., the change in momentum, from a constant, dipole or quadrupole wake field for an individual macroparticle is easily calculated via the sum over all macroparticles:

$$\Delta x'_k = \begin{cases} -\dfrac{e^2}{m\gamma\beta^2c^2}\displaystyle\sum_j W_{01}(z_k - z_j), \\ -\dfrac{e^2}{m\gamma\beta^2c^2}\displaystyle\sum_j W_{11}(z_k - z_j)\, x_j, \\ -\dfrac{e^2}{m\gamma\beta^2c^2}\displaystyle\sum_j W_{02}(z_k - z_j)\, x_k, \end{cases} \qquad j, k \in [0, \ldots, \text{macroparticle number} - 1]. \tag{54}$$

If we denote by N the macroparticle number, this results in N^2 operations and in particular in N^2 wake function evaluations, which can be computationally very expensive.

The computation can be made much faster by a simple numerical trick, which is similar to the concept of macroparticle models itself. As long as the wake function is sufficiently smooth and does not vary strongly within a given interval $[z_i, z_k]$, we can further discretize a macroparticle system into a set of longitudinal slices. We assume the wake function to be constant within a given slice. In this case, it is sufficient to evaluate the wake function only once for all macroparticles within this slice. Instead of having to do a macroparticle-to-macroparticle evaluation we now only need to perform a slice-to-slice evaluation of the wake function. Equations (54) thus become

$$\Delta x'_k[i] = \begin{cases} -\dfrac{e^2}{m\gamma\beta^2c^2}\displaystyle\sum_j N[j]\,W_{01}(z[i]-z[j]), \\ -\dfrac{e^2}{m\gamma\beta^2c^2}\displaystyle\sum_j N[j]\,W_{11}(z[i]-z[j])\,\langle x\rangle[j], \\ -\dfrac{e^2}{m\gamma\beta^2c^2}\displaystyle\sum_j N[j]\,W_{02}(z[i]-z[j])\,x_k[i], \end{cases} \tag{55}$$
$$i,j \in [0,\ldots,\text{slice number} - 1],$$
$$k \in [0,\ldots,\text{macroparticle number} - 1].$$

Here, $\Delta x'_k[i]$ is the change in momentum for a macroparticle in slice i, $N[j]$ is the number of macroparticles in slice j, $z[i]$ is the centre longitudinal position of slice i, $\langle x\rangle[j]$ is the mean horizontal offset of slice j and $x_k[i]$ is the horizontal position of a macroparticle in slice i.

If we denote by n the slice number, the number of evaluations now reduces to n^2. If we consider that typically we have of the order of several thousand macroparticles per slice, this means that the number of evaluations is reduced by six orders of magnitude compared to the macroparticle-to-macroparticle evaluation. Furthermore, if the discretization of the macroparticle system is done longitudinally uniformly such that all slices have the same width, the number of slice-to-slice distances, and therefore the necessary wake field evaluations, reduces to $2 \times n - 1$. The wake function can be pre-calculated for all those distances and the values can be stored in an array. In this case the numerical algorithm to be performed for applying wake field kicks to a macroparticle system would look as illustrated in Fig. 12.

Figure 12 assumes a macroparticle bunch encountering a constant horizontal wake function W_{Cx} (i.e., a W_{01}-type wake function in Eq. (55)). Axis A) shows the macroparticle bunch and axis B) the wake function as it will be seen by the individual slices. Axis C) plots the number of macroparticles in each slice and axis D) finally depicts the wake field kicks that will be applied to each slice. The longitudinal slices are rendered on all axes as the coloured bars, where each coloured bar corresponds to a slice. The colours in axes C) and D) match in the sense that each coloured slice on axis C) will receive the wake kick from the corresponding coloured slice on axis D).

The formula at the bottom of Fig. 12 demonstrates how a function that models wake field kicks would have to be implemented. The number of macroparticles per slice N for the n slices and the wake function W_{Cx} for each of the $2 \times n - 1$ slice-to-slice distances from $[-L, +L]$ are pre-computed and stored in two arrays. The wake field kicks are then computed according to this formula (or, similarly, to Eq. (55)) for all macroparticles in a slice by updating all the angles of those macroparticles.

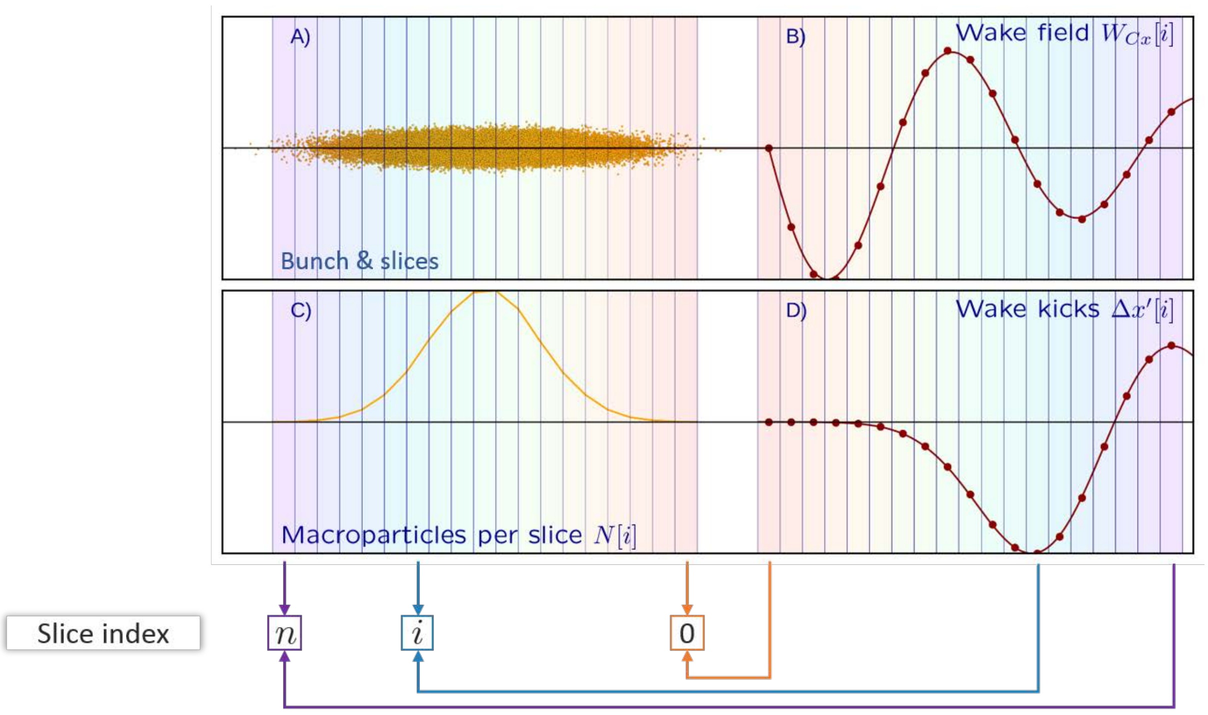
We will now go through some examples of numerically modelled wake field effects.

3.1 Constant wake fields

Constant wake fields are generated by structures with left–right or top–bottom symmetry. They create a closed orbit distortion similar to the potential well distortion that exists in the longitudinal plane. The Hamiltonian which includes constant wake fields W_{Cx} is given as

$$H = \frac{1}{2}\,x_i'^2 + \frac{1}{2}K(s)\,x_i^2 - \frac{e^2}{m\gamma\beta^2c^2C}\sum_{j=0}^{n-1} N(z_j)\,W_{Cx}(z_i - z_j)\,x_i + H_\mathrm{s}(z,\delta). \tag{56}$$

Here, x_i is a macroparticle in slice i, $N(z_j)$ is the number of macroparticles in slice j, z_j is the centre longitudinal position of slice j and n is the number of slices. The wake field term in the Hamiltonian is linear in x_i. Thus, it generates an orbit shift (like dipoles would).



$$\Delta x'[i] = -\frac{e^2}{m\gamma\beta^2c^2}\sum_{j=0}^{i} N[j]\cdot W_{Cx}[i-j]\,, \qquad x'[i] \to x'[i] + \Delta x'[i]\,, \qquad i \in [0,\ldots,n-1]$$

Fig. 12: Sketch of the algorithm to compute and apply wake field kicks. The coloured bars correspond to, and identify via their colour, the individual longitudinal slices. A) shows the macroparticle bunch. Axis B) shows the wake function (marker value) as it will be seen by the colour-corresponding slices. Axis C) plots the number of macroparticles in each slice and axis D) finally depicts the wake field kicks (marker value) that will be applied to each colour-corresponding slice. The arrows highlight which wake field kick will be applied to which slice. The wake field kick itself is evaluated for each slice according to the formula rendered below the plot.

From the Hamiltonian in Eq. (56), we can evaluate, apart from the betatron motion, which we already solved in Section 3, the corresponding additional kick from the wake fields as

$$\Delta x'_i = \underbrace{-\frac{e^2}{m\gamma\beta^2c^2}\overbrace{\sum_{j=0}^{n-1} N(z_j)\,W_{Cx}(z_i - z_j)}^{\substack{\text{Slice-dependent orbit shift}\\\text{(if line density does not change)}}}}_{\text{Dipolar term} \to \text{orbit kick}}\,. \tag{57}$$

We can see from Eqs. (56) or (57) that a constant wake field generates a slice-dependent orbit shift. This results in a closed orbit distortion.

Equipped with the possibility of initializing and tracking macroparticle systems and having now also included wake field kicks, we can run a simulation that includes constant wake field effects. Figure 13 shows an example of such a simulation. A macroparticle bunch is initialized on the design orbit. The constant wake field shifts the orbit along the bunch. As a result, the bunch oscillates around the distorted orbit. Depending on the line density of the bunch and the shape of the constant wake field, this distortion may look different. In the example above, the distortion is pronounced towards the tail of the bunch. The bunch takes on a banana-like shape and the tail of the bunch flaps around the distorted orbit. For more information on the effects of constant wakes, the reader is referred to Ref. [16].

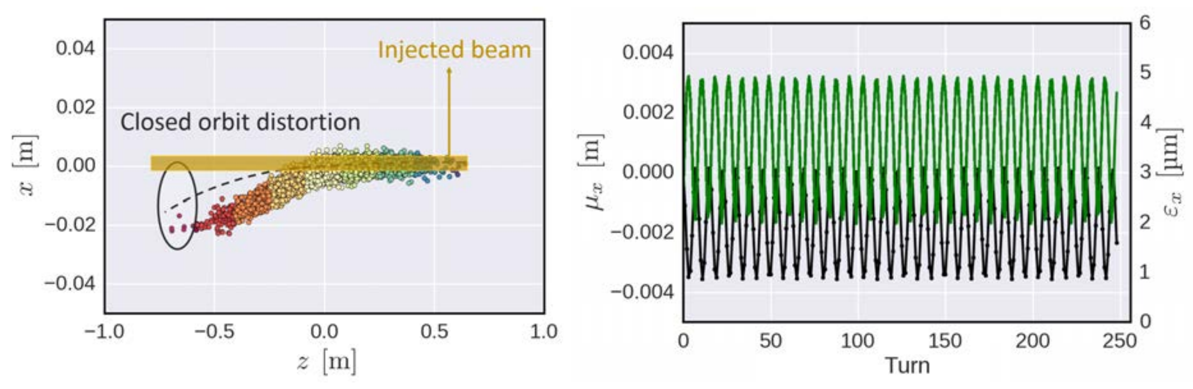


Fig. 13: Macroparticle simulation of the effects of a constant wake field. The initialization of the macroparticle bunch is outlined by the yellow rectangle. The distorted orbit is sketched by the dashed line. As a consequence of the constant wake field the tail of the macroparticle bunch will oscillate around the distorted orbit. The plot on the right shows the resulting oscillation in the mean bunch position and emittance. The motion remains bounded and the beam is stable.

3.2 Dipole wake fields

Dipole wake fields are the type of wake fields that generate instabilities. The kind of instability depends on the machine configuration. We will discuss three different forms of instabilities with simulations below. The Hamiltonian which includes dipole wake fields W_{Dx} is given as

$$H = \frac{1}{2}\,x_i'^2 + \frac{1}{2}K(s)\,x_i^2 - \frac{e^2}{m\gamma\beta^2c^2C}\sum_{j=0}^{n-1} N(z_j)\,\langle x_j\rangle\, W_{Dx}(z_i - z_j)\,x_i + H_{\rm s}(z,\delta). \tag{58}$$

Here, x_i is a macroparticle in slice i, $N(z_j)$ is the number of macroparticles in slice j, z_j is the centre longitudinal position of slice j and n is the number of slices. The wake field term in the Hamiltonian is linear in x_i. Thus, it generates an orbit shift (like dipoles would). However, in contrast to the constant wake field case we see that the Hamiltonian now contains an additional dependency on $\langle x_j\rangle$. Thus, different slices within the bunch get coupled. One can already imagine that this coupling of slices can lead to peculiar effects.

From the Hamiltonian in Eq. (58), we can evaluate, again apart from the betatron motion, the corresponding additional kick from the wake fields as

$$\Delta x_i' = \underbrace{-\frac{e^2}{m\gamma\beta^2c^2}\overbrace{\sum_{j=0}^{n-1} N(z_j)\,\langle x_j\rangle\, W_{Dx}(z_i - z_j)}^{\text{Slice-dependent orbit shift}}}_{\text{Dipolar term}\,\to\,\text{orbit kick}}. \tag{59}$$

We can see from Eqs. (58) or (59) that a dipole wake field generates a slice-dependent orbit shift. This shift in orbit is now not only dependent on the charge of a given slice but it is dependent on the weighted mean horizontal offset of the slice. Thus, the dynamics of the individual slices enters back into the equation. One can imagine that there are stationary solutions to the system described by the Hamiltonian in Eq. (58) and characterized by the kick in Eq. (59). Indeed, such solutions can be found analytically for simple cases by employing the Vlasov equation in a perturbation approach, as done in Ref. [12].

3.2.1 Beam break-up instability

Having learned how to implement dipolar wake fields, we can study the beam dynamics numerically. As the first case, we will investigate the behaviour of a bunch subject to a dipolar wake field in the absence of synchrotron motion. In this configuration, the head of the bunch will generate dipolar wake fields, which will shift the orbit of the following slices in the bunch. This will trigger a mean dipole motion of these slices, which consequently will generate stronger dipolar wake fields for the slices in the bunch still to follow. This process builds up turn by turn causing the tail of the bunch to oscillate increasingly violently and rendering the bunch unstable. This type of instability is the beam break-up instability and can occur in linear accelerators or in circular accelerators at transition where the synchrotron motion is frozen. The beam is inherently unstable and will grow exponentially if no additional means of stabilization is put in place. Figure 14 shows a simulation of a macroparticle bunch subject to a dipolar wake field. The synchrotron motion was turned off. The build-up of the oscillation towards the tail of the bunch is clearly visible. This is accompanied by a fast exponential growth of the motion and the emittance of the bunch.

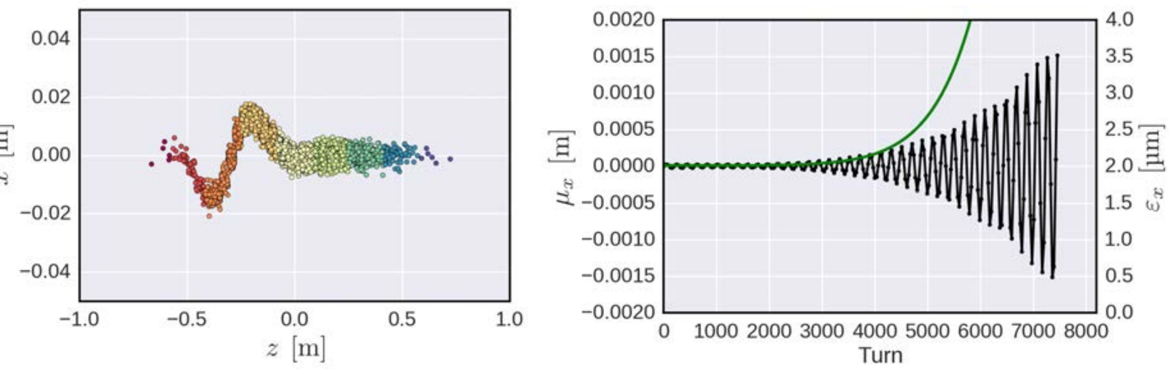


Fig. 14: Macroparticle simulation of the effects of a dipolar wake field. The synchrotron motion was turned off. The build-up of the oscillation towards the tail of the bunch is clearly visible in the left-hand plot. The exponential growth of the mean position and emittance of the bunch is shown in the right-hand plot.

3.2.2 Transverse mode coupling instability

We repeat the simulation from above investigating the behaviour of a bunch subject to a dipolar wake field; however, now we include synchrotron motion. We keep the chromaticity at zero. In this configuration, the head of the bunch will still generate dipolar wake fields, which will shift the orbit of the following slices in the bunch, triggering a mean dipole motion of these slices. Due to the synchrotron motion, eventually, the head and tail slices will exchange their positions. This will prevent the motion of the tail slices from building up coherently. In fact, in absence of chromaticity, gradually, this motion that builds up within a certain fraction of a synchrotron period gets washed out again at a later stage. Synchrotron motion acts as a stabilizing mechanism and the larger the synchrotron tune the more effective this washing-out of the coherent dipole motion is. This is true as long as the beam intensity remains below the TMCI threshold. At this point, the beam intensity is so high that the dipole motion builds up coherently before it can be cleared away via the synchrotron motion. A particularly violent instability emerges from this, which in many aspects is similar to the beam break-up instability discussed above. Figure 15 shows an example of a simulation of a TMCI. The macroparticle bunch is heavily distorted and the motion and the emittance of the bunch grow exponentially.

We can run several simulations at different intensities to gain more insight into this mechanism. For each simulation, at a given intensity, we do a spectral analysis of the coherent motion. We obtain spectra similar to the ones shown in Fig. 16. Below the TMCI threshold we can distinguish several spikes in the spectrum which correspond to different azimuthal and radial modes which arise from the interaction of the macroparticle bunch with the dipolar wake fields in combination with the synchrotron motion. These are in fact the stationary solutions, which were already mentioned earlier and discussed in more detail in Ref. [12]. What can be observed is that, as the intensity increases, certain modes tend to approach each other. This is indicated in the left-hand plot in Fig. 16. At a certain intensity these

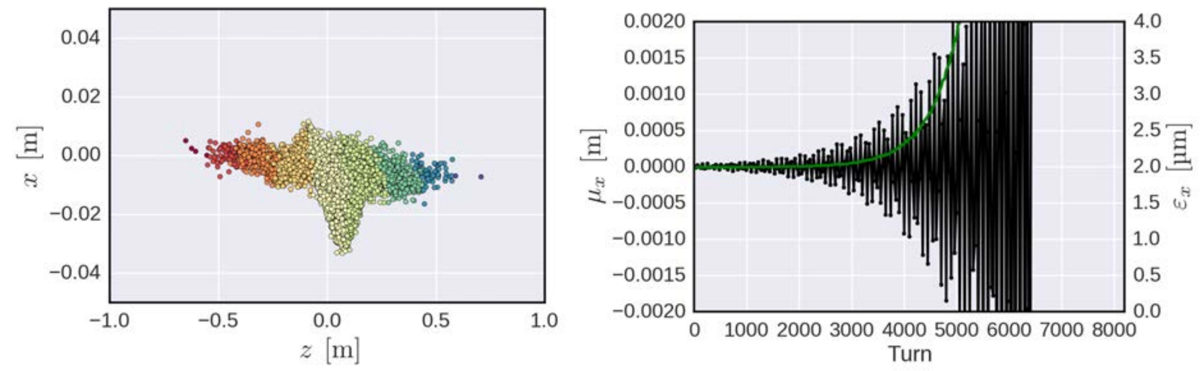


Fig. 15: Macroparticle simulation of the effects of a dipolar wake field. The synchrotron motion in turned on. The bunch intensity is above the TMCI threshold. The macroparticle bunch is heavily distorted. The exponential growth of the mean position and emittance of the bunch is shown in the right-hand plot.

modes merge or couple, as shown in the right-hand plot in Fig. 16. This is exactly the TMCI threshold at which the beam becomes violently unstable. The TMCI is a threshold effect. The beam is entirely stable below the intensity threshold and becomes inherently unstable above this threshold. Therefore, the TMCI threshold is often a hard limit for most machines.

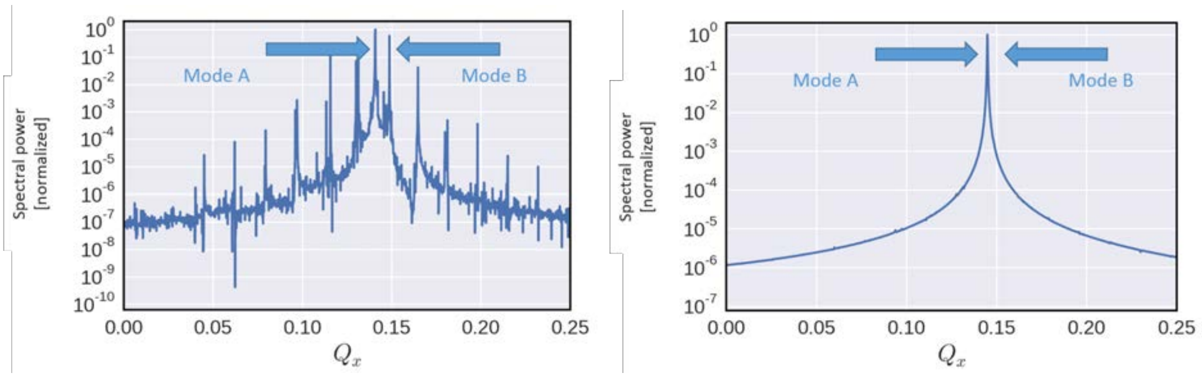


Fig. 16: Bunch spectra obtained from the mean position for a simulation below (left) and above (right) the intensity threshold for TMCI. Several spikes are distinguishable below the threshold. Two of the spikes (modes A and B) approach each other as the intensity increases until they finally merge as shown in the right-hand plot. At this point the bunch becomes violently unstable.

One of the strategies employed to raise the TMCI threshold is to increase the synchrotron tune. This was implemented recently in the CERN Super Proton Synchrotron (SPS) [17]. In changing the machine optics, the transition energy could be lowered such that the injection energy of the beam actually ended up farther away from the transition energy. This leads to an increase in the synchrotron tune. Figure 17 shows two plots in the way they are typically presented to visualize TMCI. The plots feature the bunch spectra (mode number on the y axis vs. marker colour and size) laid out against the bunch intensity. The mode number is given as the betatron tune shift in units of the synchrotron tune $m = (Q_x - Q_{x0})/Q_\mathrm{s}$. The left-hand plot in Fig. 17 displays the coherent tune shifts with intensity and highlights a mode coupling of modes –2 and –3 taking place at an intensity of 1.5×10^{11} ppb. Increasing the synchrotron tune leads to a larger separation of the individual mode lines causing them to couple only at a later stage at 4×10^{11} ppb, as illustrated in the right-hand plot in Fig. 17.

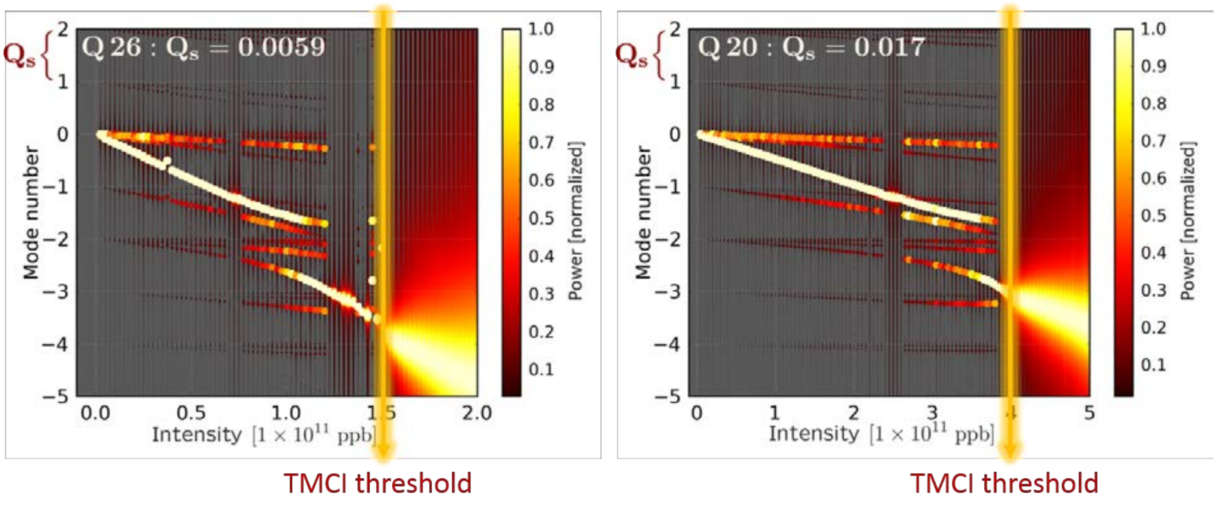


Fig. 17: TMCI spectra in the SPS for the two different optics (Q26 & Q20). The bunch spectra are plotted against the intensity. The spectral power is indicated by the markers and scales with the marker size and colour according to the legend. Mode coupling occurs between modes –2 and –3 at an intensity of $1.5 \times 1{-}^{11}$ ppb in the left-hand plot for which $Q_\mathrm{s} = 0.0059$ and at an intensity of 4×10^{11} ppb in the right-hand plot for which $Q_\mathrm{s} = 0.017$.

3.2.3 Head–tail instability

Finally, we study the case of a bunch subject to a dipolar wake field including synchrotron motion and now also including chromaticity. In this configuration, the head of the bunch will generate dipolar wake fields shifting the orbit of the following slices in the bunch while the synchrotron motion will periodically exchange head and tail slices within the bunch. As long as we are below the TMCI threshold, the bunch would be stable due to the synchrotron motion clearing away the dipole motion before it can build up coherently. This is true at zero chromaticity. Chromaticity, however, correlates the transverse motion of slices with their longitudinal position and thus incorporates a form of temporal interrelationship between the transverse and the longitudinal motions. This introduces a synchronicity mechanism which can lead to the synchrotron motion no longer clearing away the coherent dipole motion, but, instead, may cause it to actually seed a new form of coherent dipole motion. Finite chromaticity in combination with synchrotron motion (of course, otherwise chromaticity has no effect) and dipolar wake fields, in this case, generate head–tail instabilities. These are usually much less violent, i.e., they feature slower rise times, than the TMCI discussed in the previous section. On the other hand, head–tail instabilities are not a threshold effect. At finite chromaticity a bunch is inherently unstable, which would lead to an exponential growth of the head–tail mode if no additional means of stabilization is put in place. Head–tail modes are, again, stationary solutions of the accelerator–beam system. The imaginary part of their complex tune shift is given essentially by the overlap of the mode spectrum with the impedance. At zero chromaticity this overlap vanishes. At finite chromaticity certain modes are damped while others grow. Head–tail modes feature distinct patterns when observed in a pickup.

Figure 18 shows simulations of a head–tail instability for two sets of chromaticities compared to measurements in the LHC. The modes clearly exhibit their time-stationary nature. The different head–tail modes can be characterized by the number of nodes seen along the waveform. This number in fact corresponds to the radial mode number when performing the analytical analysis.

In summary, dipole wake fields generate instabilities that can be effectively investigated by numerical methods using macroparticle models. We looked at three different types of instabilities, which can be classified as shown in Table 2.

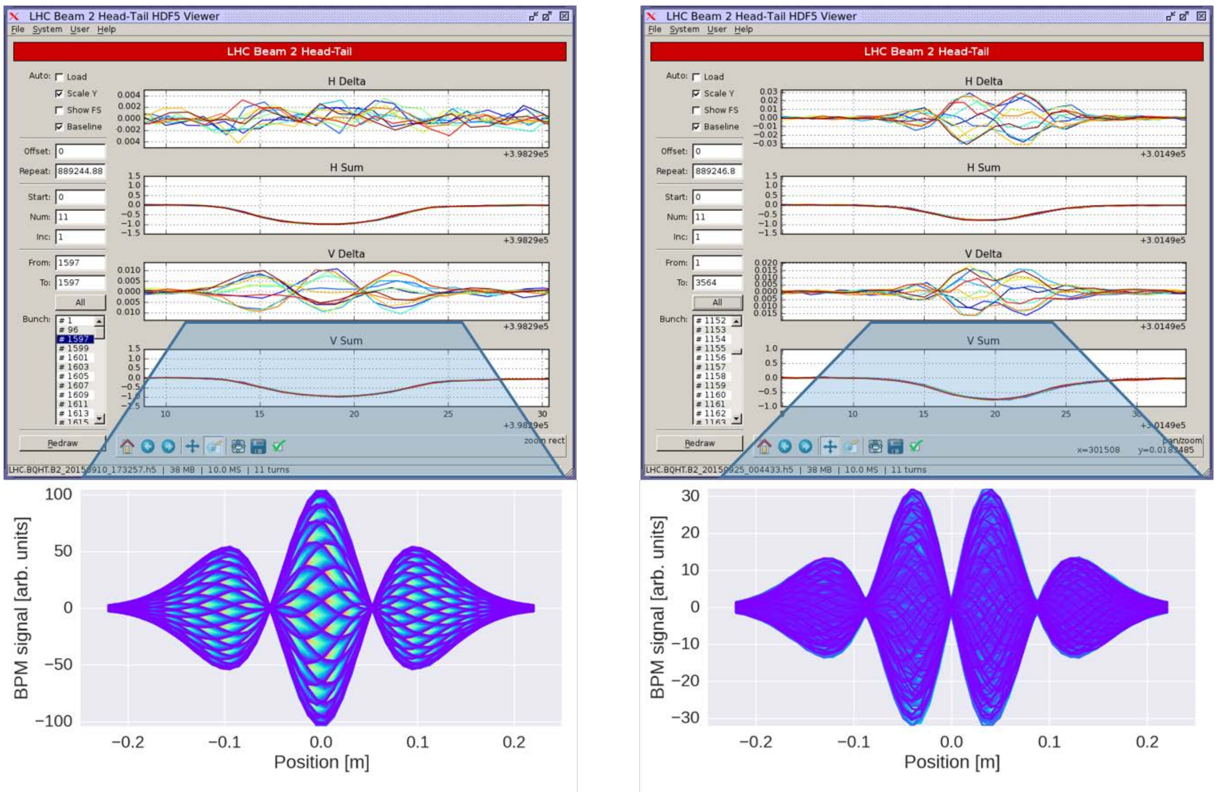


Fig. 18: Macroparticle simulation of a head–tail instability compared with measurements in the LHC. The left-hand plot shows a (radial) mode 2 instability occurring at a chromaticity of $Q' = 10$. The right-hand plot shows a (radial) mode 3 instability occurring at a chromaticity of $Q' = 15$.

Table 2: Classification of different types of instabilities generated by dipolar wake fields along with their conditions.

	Synchrotron motion	Chromaticity	Threshold effect	Rise time
Beam break-up	No	Irrelevant	No	Fast
TMCI	Yes	No	Yes	Fast
Head–tail instability	Yes	Yes	No	Slow

3.3 Quadrupole wake fields

Quadrupole wake fields are the highest order wake fields in the target coordinates that are commonly still considered. They do not usually generate instabilities. The Hamiltonian which includes quadrupole wake fields W_{Qx} is given as

$$H = \frac{1}{2}\,x_i'^2 + \frac{1}{2}K(s)\,x_i^2 - \frac{e^2}{m\gamma\beta^2c^2C}\sum_{j=0}^{n-1} N(z_j)\,W_{Qx}(z_i - z_j)\,x_i^2 + H_\mathrm{s}(z,\delta). \tag{60}$$

Here, x_i is a macroparticle in slice i, $N(z_j)$ is the number of macroparticles in slice j, z_j is the centre longitudinal position of slice j and n is the number of slices. The wake field term in the Hamiltonian is quadratic in x. Thus, it generates a tune shift (like quadrupoles would).

From the Hamiltonian in Eq. (60), we can evaluate, again apart from the betatron motion, the corresponding additional kick from the wake fields as

$$\Delta x_i' = \underbrace{-\frac{e^2}{m\gamma\beta^2 c^2} \overbrace{\sum_{j=0}^{n-1} N(z_j)\, W_{Qx}(z_i - z_j)}^{\substack{\text{Slice-dependent tune shift} \\ \text{(if line density does not change)}}}}_{\text{Quadrupolar term} \,\rightarrow\, \text{tune kick}} \, x_i. \tag{61}$$

We can see from Eqs. (60) or (61) that a quadrupole wake field generates a slice-dependent tune shift. This results in an additional coherent tune shift, which in combination with dipole wake fields may expedite or hinder mode coupling. In addition it introduces a tune spread within the bunch combined with all the consequences such as potential resonance crossing or in principle even Landau damping.

Having learned how to implement quadrupolar wake fields, we can study the beam dynamics numerically. We find that a bunch subject to a quadrupolar wake field remains stable. We can perform a spectral analysis of each individual macroparticle to obtain the tune footprint of the macroparticle system. The resulting footprint is shown in Fig. 19. It is clearly visible how the footprint is grouped according to the slices of the bunch. Each coloured marker is actually a group of macroparticles within a given slice. The slice-dependent tune shift reflects the convolution with the quadrupolar wake field according to Eq. (61).

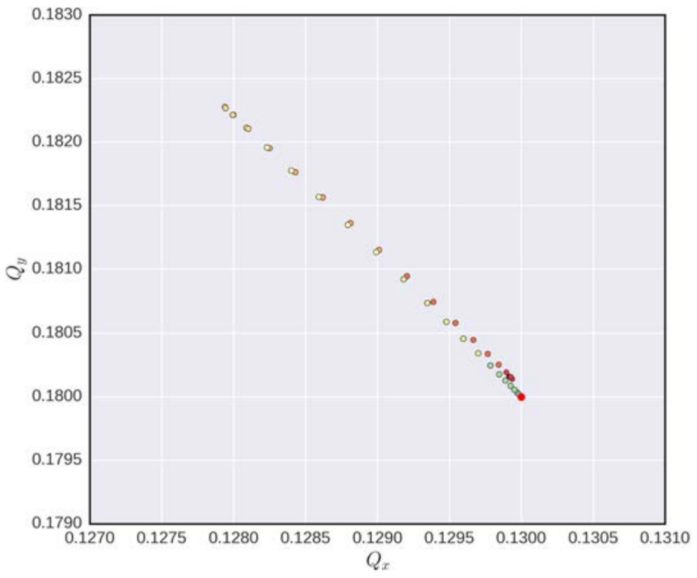
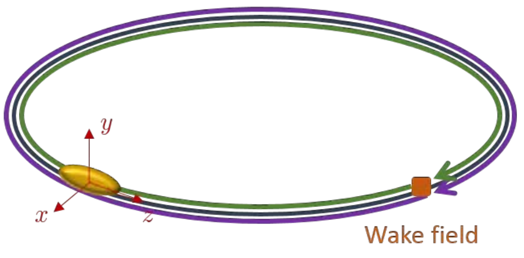


Fig. 19: Tune footprint of a macroparticle bunch subject to a quadrupolar wake field

This concludes our investigation of the numerical modelling of wake fields. This was the first truly collective effect we implemented which had to be treated taking the full macroparticle system self-consistently into account. We introduced wake fields as the electromagnetic response function of an object and learned that the net wake field kick is obtained via the convolution of the wake function with the macroparticle distribution. We studied constant, dipole and quadrupole wake fields with simulations and learned about the different types of instabilities. In the final part of the lectures we will examine another type of collective effect. This will consist of a second macroparticle system to interact with our macroparticle bunch. Being a dynamical system itself, this macroparticle system cannot be modelled via an electromagnetic response function as done for a rigid object via the wake fields. This will make the problem quite a bit more involved.

4 Modelling beam dynamics together with electron clouds

In the previous sections we learned how to initialize a macroparticle system representing a physical particle beam and how to track this macroparticle system through a circular machine transversely and longitudinally given the machine optics and RF configuration. Finally, we learned how to include and treat collective effects via wake fields to interact self-consistently with this macroparticle system. A feature of wake fields—being the electromagnetic response function of an object—is that they are a stationary property of the object that generates them. Unless the object changes its geometry or electromagnetic properties the wake function will not change. It can be computed once for an object and can then be used turn after turn. Figure 20 summarizes the numerical treatment of beam dynamics under the influence of collective effects via macroparticle models developed so far.



$$\begin{pmatrix} x_i \\ x'_i \end{pmatrix}\Bigg|_{k+1} = \mathcal{M}_i \begin{pmatrix} x_i \\ x'_i \end{pmatrix}\Bigg|_{k}$$

$$\begin{pmatrix} z_i \\ \delta_i \end{pmatrix}\Bigg|_{k+1} = \mathcal{I}\left[\begin{pmatrix} z_i \\ \delta_i \end{pmatrix}\Bigg|_{k}\right]$$

$$(x'_i)\big|_{k+1} = (x'_i)\big|_{k} + \mathcal{WK}$$

1. Initialize a macroparticle distribution with a given emittance
2. Update transverse coordinates and momenta according to the linear periodic transfer map – adjust the individual phase advance according to chromaticity and detuning with amplitude
3. Update the longitudinal coordinates and momenta according to the leap-frog integration scheme
4. Update momenta only (apply kicks) according to wake field generated kicks
5. Repeat turn-by-turn…

Fig. 20: Figurative summary of beam dynamics components treated so far. These include macroparticle system initialization, single-macroparticle tracking in the transverse and the longitudinal planes and collective interaction among macroparticles via wake fields.

In this section we will investigate another form of collective effects introduced not by a static object but by either the charged particle beam itself or also by another charged particle system. This can be a second beam which brings about the beam–beam interaction. This is a phenomenon that needs to be studied around the collision point of particle colliders. Instead of another beam it can also be a different form of charged particle distribution even with different particle species and energies such as for example electron clouds. We will use the example of the electron cloud throughout the rest of this section to illustrate the numerical concepts and methods used to treat these types of collective effects.

Electron clouds are a phenomenon that develops in accelerators of high-intensity positively charged beams. Primary electrons get released through residual gas ionization or synchrotron radiation. These electrons are accelerated in the electromagnetic fields generated by the positively charged circulating beam. At some point they will hit the vacuum chamber walls and depending on their energy and impact angle as well as on the surface properties—characterized by the secondary electron yield (SEY)—they will release a certain amount of secondary electrons. These will again be accelerated by the circulating beam and will also hit the vacuum chamber walls releasing yet more secondary electrons. Depending on the configuration of the circulating beam (intensity, bunch spacing etc) this can lead to an avalanche effect in generating an increasing amount of electrons, which we call multipacting. This generation of electrons will saturate when the electron density has reached a value such that the resulting space-charge force repels secondary electrons sufficiently to suppress any further multipacting. At the end of this electron cloud build-up process one is left with stationary distributions of electrons at different locations around the ring. We call these stationary distributions electron clouds. Of course, one can imagine that these electron clouds are able to generate fields that can substantially perturb the motion of the circulating beam.

Electron clouds contain their own dynamics and macroscopic properties. In contrast to the static objects treated via wake fields in the previous section, electron clouds dynamically change their macroscopic properties. As such, if one were to calculate the electromagnetic response function of electron clouds, this would continuously change. In addition, the electromagnetic fields generated by electron clouds are highly non-linear, so the linear treatment using dipole and quadrupole wake fields would no longer be appropriate and higher order wake fields would have to be included. Most of the advantages gained from the wake field concept would no longer apply for electron clouds. Other strategies are required for an efficient numerical treatment of the interaction of charged particle beams with electron clouds.

4.1 Electron clouds as additional macroparticle systems

To numerically treat the electron cloud effects, the electron cloud has to be modelled as a macroparticle system in itself. This makes beam dynamics simulations with electron clouds much more involved, as one now needs to treat not only the macroparticle system resembling the physical particle beam but also one or several macroparticle systems resembling the electron clouds.

The same reasoning that was applied earlier for a physical particle beam also applies for the electron clouds. We can handle the electron clouds numerically by clustering the physical electrons into a discrete set of macroparticles while at the same time taking care about noise issues. We need to know the generalized coordinates and the canonically conjugate momenta of each individual macroparticle along with its charge and mass. For reasons that will be discussed further below, the electron cloud macroparticle system can be treated purely transversely. As such, it will be fully characterized by the generalized coordinates, given as the horizontal and vertical positions (x, y) in metres with respect to the centre of the vacuum chamber (usually) and the canonically conjugate momenta, given as the corresponding horizontal and vertical velocities $(\dot{x}, \dot{y})$. Initialization will usually take place as import of a distribution from an electron cloud build-up simulation. The electron cloud build-up simulations are treated in detail in Refs. [18, 19] and will not be further discussed here. If a distribution from an electron cloud build-up simulation is not available, one often chooses to start from a distribution that uniformly fills the vacuum chamber (or a part of it) with a given electron density.

As done earlier for the charged particle beam, the electron cloud macroparticle system can be represented by some allocated memory block in the main memory which is sufficiently large to accommodate all the relevant quantities of the macroparticle system, i.e., generalized coordinates and canonically conjugate momenta, charges and masses. Table 3 shows an example memory layout of an electron cloud macroparticle system in the main memory where each of the four phase-space variables is allocated as an array of length equal to the macroparticle number which contains the corresponding values for every macroparticle.

Table 3: Example memory layout of an electron cloud macroparticle system in the main memory. Each of the four phase-space variables is allocated as an array of length equal to the macroparticle number N.

Count	0	1	2	...	N
x	...	...	...	...	...
$\dot{x}$	...	...	...	...	...
y	...	...	...	...	...
$\dot{y}$	...	...	...	...	...

The action of the electron cloud on a charged particle beam will apply via the electromagnetic fields generated by the electron cloud and impacting the charged particle beam. On the other hand, the charged particle beam acts back onto the electron cloud as it also generates electromagnetic fields which in turn will cause electrons to be accelerated towards the positively charged particle beam. It follows that we will now have to handle the dynamics of the two macroparticle systems self-consistently.

During the beam passage, the electrons are accelerated, which causes the electron cloud to reform and, therefore, changes its macroscopic distribution. The electromagnetic fields generated by the electron cloud are thus strongly modified and consequently highly dynamic already during a single bunch passage. The resulting electromagnetic fields in the vicinity of the passing beam will perturb the beam motion and eventually lead to a distortion of the particle distribution within the beam. Then, again, the electromagnetic fields generated by the passing beam are changed, which in turn affects the electron dynamics and electron cloud reformation (and thus also the electromagnetic fields generated by the electron cloud) during the beam passage.

In frozen models, the distortion of the particle distribution within the beam is not taken into account and the electromagnetic fields are computed only for one beam passage. The electromagnetic fields generated by the electron cloud during this beam passage are stored as a field map and are then re-applied to the beam at every passage. Fully self-consistent models, on the other hand, do a full re-computation of the electromagnetic fields taking into account the current distributions of both the electron cloud and the charged particle beam at every iteration. It is clear that this becomes a strongly coupled system of many subsystems where each of the subsystems already contains highly complex dynamics of their own.

The motions of the electrons and the charged particles in the beam take place at very different time-scales due to their very different energy scales (the electrons are nearly at rest while the beam is moving at relativistic speeds). The electron motion takes place within fractions of the bunch length whereas the charged particle motion takes place within fractions of a revolution period. We will discuss the modelling and implementation of the electron cloud and charged particle beam interaction in the next sections.

4.2 Numerical treatment and computation of fields

The modelling of the accelerator–beam system now needs to include electron clouds as part of the accelerator, with these, at the same time, being independent macroparticle systems with their own dynamics. Figure 21 shows a sketch of a model machine layout. At each interaction the electromagnetic fields need to be computed for both the electron cloud and the charged particle beam. These fields then need to be applied to the two macroparticle systems and the macroparticles need to be propagated. The computation of the electromagnetic fields for the, in principle, arbitrary macroparticle distributions is, in general, a non-trivial task. Fortunately, we can employ a few basic assumptions which will simplify this exercise.

The electron cloud is assumed to be distributed over a considerable length L along the vacuum chamber much larger than its transverse dimensions b such that is has a low aspect ratio $A = b/L \ll 1$. In this case we can assume the electric fields of the electron cloud to be purely transverse. Since the electrons are also nearly at rest just before the beam passage we have for the electron velocity $v \ll c$ and the magnetic fields of the electron cloud can be practically neglected. The charged particle beam is assumed to be close to relativistic. For these types of beams we know that the electric field in the laboratory frame is strongly enhanced in the direction perpendicular to the direction of flight, i.e. it is also almost purely transverse. As mentioned already we can now see that the electromagnetic interaction between the electron cloud and the charged particle beam can be treated purely transversely and as an electrostatic problem.

Hence, what we need to do is to numerically solve the 2D Poisson equation at every encounter. Using the electric fields obtained from the solution, we then need to kick and propagate both the electrons and the charged particles in the beam. Since the electron motion happens very fast—on the time-scale of

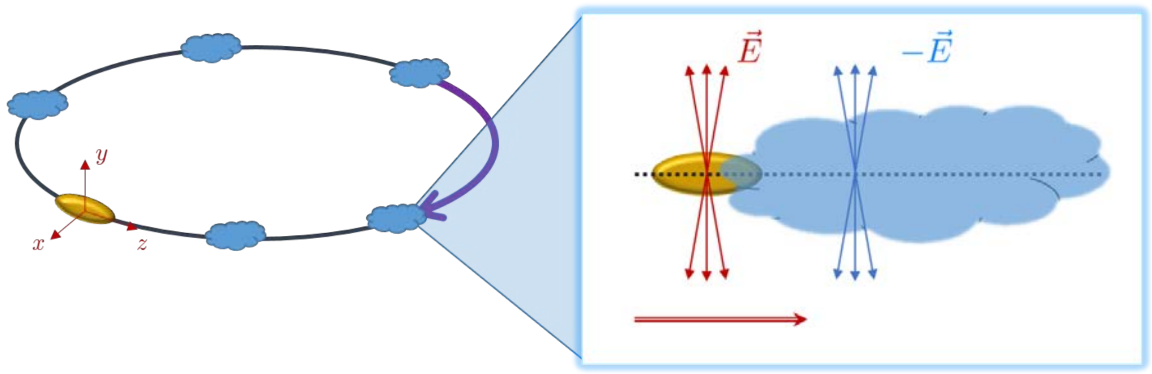


Fig. 21: Machine layout for a machine containing electron clouds. The electron clouds are placed at several locations along the ring. At each location, an electron cloud with charged particle beam interaction takes place.

fractions of a bunch length—the electrons receive a kick and then have to be propagated within this time frame, while the charged particles in the beam receive just the kick and are propagated through the ring via the betatron motion. For this reason, a macroparticle bunch is discretized into a set of longitudinal slices, as was done for the wake field interactions. The slice width will determine the maximum time frame in which the electrons will be linearly propagated. This situation is depicted in Fig. 22.

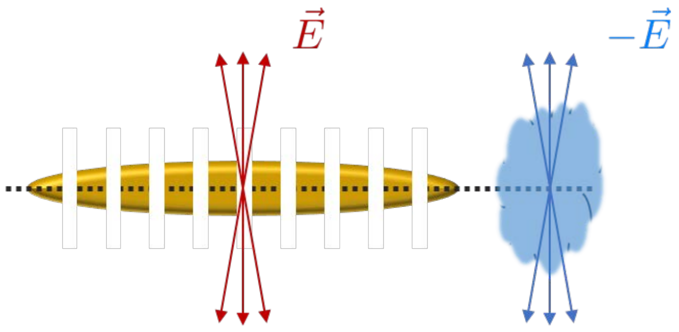


Fig. 22: Discretization of the incoming charged particle beam into a set of longitudinal slices. The interaction takes place slice-by-slice and the electrons are propagated on the time-scale of a single slice.

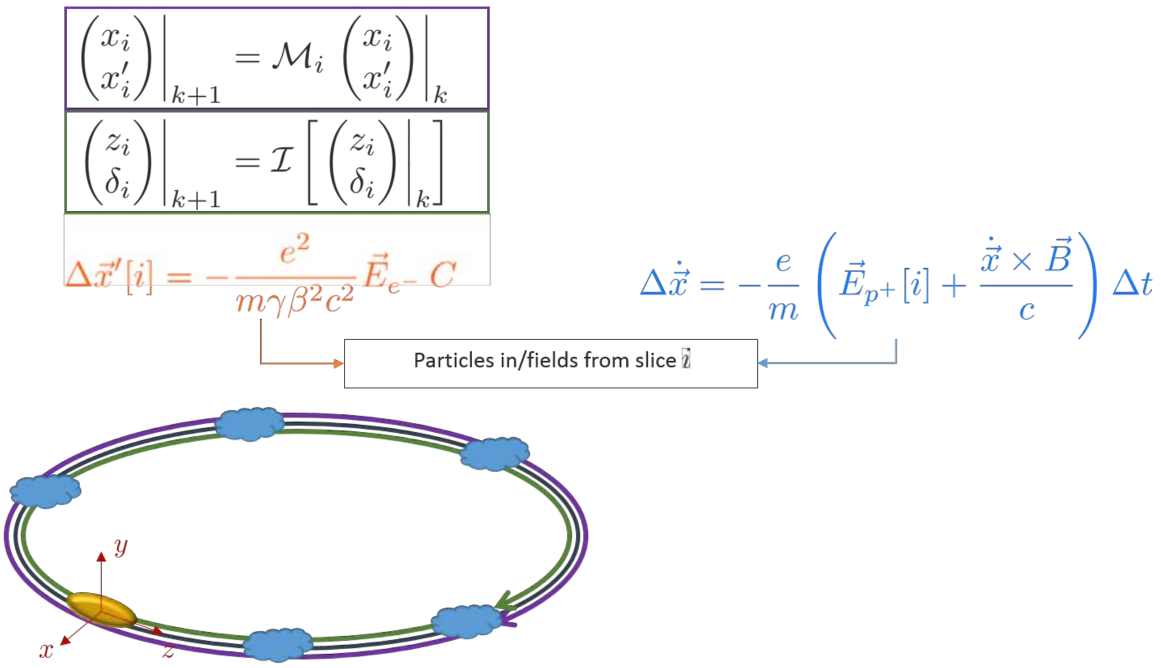
The motion of the charged particle beam on the other hand takes place on a much larger time-scale. Hence, the impact of the electron cloud is treated as an integrated effective force and the total kick is applied after the passage of a slice through the electron cloud. As such, since the kicks from each segment of the electron cloud will just linearly add up, the electron cloud can be longitudinally projected and treated for each slice in a 2D plane.

The overview of the numerical treatment of beam dynamics under the influence of electron clouds is now slightly adapted in comparison with Fig. 20 and is pictured in Fig. 23.

For each slice at each iteration the following steps are performed. First, one needs to compute the electric fields of the respective slice and of the projected electron cloud by numerically solving the Poisson equation

$$\begin{aligned} \Delta\phi_{\mathrm{e}^-}(x,y) &= -\frac{\rho_{\mathrm{e}^-}(x,y)}{\varepsilon_0}, \\ \Delta\phi_{\mathrm{p}^+}(x,y) &= -\frac{\rho_{\mathrm{p}^+}(x,y)}{\varepsilon_0}, \end{aligned} \tag{62}$$

where ϕ_{e^-} and ϕ_{p^+} and ρ_{e^-} and ρ_{p^+} are the potentials and densities of the electrons and the charged particles of the beam, respectively. There are several means of numerically solving the 2D Poisson



$\begin{pmatrix} x_i \\ x_i' \end{pmatrix}\Big|_{k+1} = \mathcal{M}_i \begin{pmatrix} x_i \\ x_i' \end{pmatrix}\Big|_k$
$\begin{pmatrix} z_i \\ \delta_i \end{pmatrix}\Big|_{k+1} = \mathcal{I}\left[\begin{pmatrix} z_i \\ \delta_i \end{pmatrix}\Big|_k\right]$
$\Delta\vec{x}'[i] = -\frac{e^2}{m\gamma\beta^2c^2}\vec{E}_{e^-}C$
$\Delta\dot{\vec{x}} = -\frac{e}{m}\left(\vec{E}_{p^+}[i] + \frac{\dot{\vec{x}}\times\vec{B}}{c}\right)\Delta t$
Particles in/fields from slice i
y
x
z

Fig. 23: Figurative summary of beam dynamics with electron clouds. The single- macroparticle tracking in the transverse and the longitudinal planes is included as earlier. In addition, several electron clouds are placed along the ring. The momenta of the electrons and the charged particles in the beam are updated according to the self-generated electric fields and any potential external magnetic field.

equation. One of the rather widely used methods for solving electron cloud problems is the particle-in-cell (PIC) algorithm—this will be discussed in more detail in the Appendices.

Having obtained the electric field of the electron cloud, we apply the corresponding kick to all macroparticles in the slices as already indicated in Fig. 23,

$$\begin{aligned} \Delta x_k'[i] &= \frac{e^2}{m\gamma\beta^2c^2}\, E_x(x_k, y_k)[i]\, L, \\ \Delta y_k'[i] &= \frac{e^2}{m\gamma\beta^2c^2}\, E_y(x_k, y_k)[i]\, L, \\ i &\in [0, \dots, \text{slice number} - 1], \\ k &\in [0, \dots, \text{macroparticle number} - 1]. \end{aligned} \tag{63}$$

Here, L is the length of the electron cloud and E_x and E_y the generated horizontal and vertical electric fields during the passage of slice i.

In the final step we need to update the velocities and positions of the electrons in the electron cloud. This is a multiscale problem and requires careful treatment. As already seen in Fig. 23, we will need to solve the system of equations

$$\begin{aligned} \dot{\vec{x}} &= \vec{v}, \\ \dot{\vec{v}} &= \frac{e}{m}\left(\vec{E}_{\mathrm{p}^+} + \frac{\vec{v}\times\vec{B}_{\mathrm{ext.}}}{c}\right), \end{aligned} \tag{64}$$

where now, to be general, we set $\vec{x}$, $\vec{v}$, $\vec{E}$, $\vec{B}$ to be vectors. Looking at Eq. (64), we realize that the equations of motion contain a magnetic field term.

The majority of segments of a circular accelerator actually consist of regions encompassed by magnetic fields. If these regions are occupied by electron clouds, which is usually the case, then there will be a significant impact of these external magnetic fields on the electron dynamics, which needs to be taken into account. To numerically solve the set of equations in (64), these are discretized in time to

$$\begin{aligned}\frac{\mathbf{x}_{k+1}-\mathbf{x}_k}{\Delta t} &= \mathbf{v}_{k+1},\\ \frac{\mathbf{v}_{k+1}-\mathbf{v}_k}{\Delta t} &= \frac{e}{m}\left(\mathbf{E}_k+\frac{(\mathbf{v}_{k+1}+\mathbf{v}_k)\times\mathbf{B}_k}{2c}\right).\end{aligned} \tag{65}$$

From Eq. (65), the multiscale nature of the problem now becomes evident (slow guiding drift from electric fields with fast cyclotron motion from magnetic fields). For a correct treatment of the electron motion, the cyclotron motion needs to be well resolved. The cyclotron period can be as short as fractions of a slice length such that a single time step has to be split into even smaller subintervals. The special, apparently implicit form of the velocity equation in Eq. (65), using the average velocity at a time $k+1/2$, ensures that the algorithm conserves the phase-space volume, i.e. it is volume-preserving [3]. This is important as electrons can perform several cyclotron oscillations during one slice passage and the motion is required to remain stable. Because it is implicit, the numerical solution of Eq. (65) is rather expensive. Fortunately, there is another slightly adapted algorithm that is very well suited to treat these kinds of problems. This is the Boris algorithm which is borrowed from plasma physics where it is the de facto standard for particle pushing (see Refs. [20, 21]). While Eq. (65) appears to be implicit, it can actually be made manifest explicit by writing

$$\mathbf{v}^- = \mathbf{v}_k + \frac{e}{m}\mathbf{E}_k\frac{\Delta t}{2}, \tag{66a}$$

$$\frac{\mathbf{v}^+-\mathbf{v}^-}{\Delta t} = \frac{e}{2mc}\left((\mathbf{v}^++\mathbf{v}^-)\times\mathbf{B}_k\right), \tag{66b}$$

$$\mathbf{v}_{k+1} = \mathbf{v}^+ + \frac{e}{m}\mathbf{E}_k\frac{\Delta t}{2}. \tag{66c}$$

Equations (66a) and (66c) eliminate the electric field. Then Eq. (66b) can in fact be solved by basic geometric means. In assuming this form, the algorithm—although it is not strictly symplectic—becomes explicit (efficiency), time centred (local accuracy) and the energy error is globally bound (preservation of phase-space volume).

With this final step in updating the velocities and positions of the electrons, we have completed the cycle for one slice and can now take on the following slice. By propagating the electrons in this way, in a slice-by-slice manner, during the beam passage, electrons will be attracted towards the passing beam. We call this process the pinch of the electron cloud, as electrons are pinched towards the beam. A schematic of this slice-by-slice interaction during a beam passage is shown in Fig. 24.

Depending on the configuration of the external magnetic fields the electrons will form different structures in electron density and exhibit different locations where the electrons will be concentrated into hot spots. Fig. 25 shows electron cloud pinches for a field-free region compared to a dipole and a quadrupole magnetic field region. The simulation uses an LHC-type vacuum chamber. Regions of high electron density are light coloured. The different patterns in electron cloud density are clearly visible. In the field-free region the electrons are free to propagate towards the passing beam. In regions with magnetic fields the electrons are trapped to circulate around and follow the magnetic field lines. In dipoles, this leads to the formation of the characteristic stripes of high electron density. In quadrupoles, the magnetic field can even form a magnetic bottle causing electrons to be reflected and preventing further impact into the vacuum chamber but forcing them around the beam. Figure 25 also highlights the multipacting that takes place on the inner surface of the vacuum chamber.

[3] In fact, Eq. (65) resembles a leap-frog scheme, which is one of the most simple symplectic integrators. It is not strictly a leap-frog scheme, though, because in a leap-frog scheme the accelerating force cannot depend on velocity.

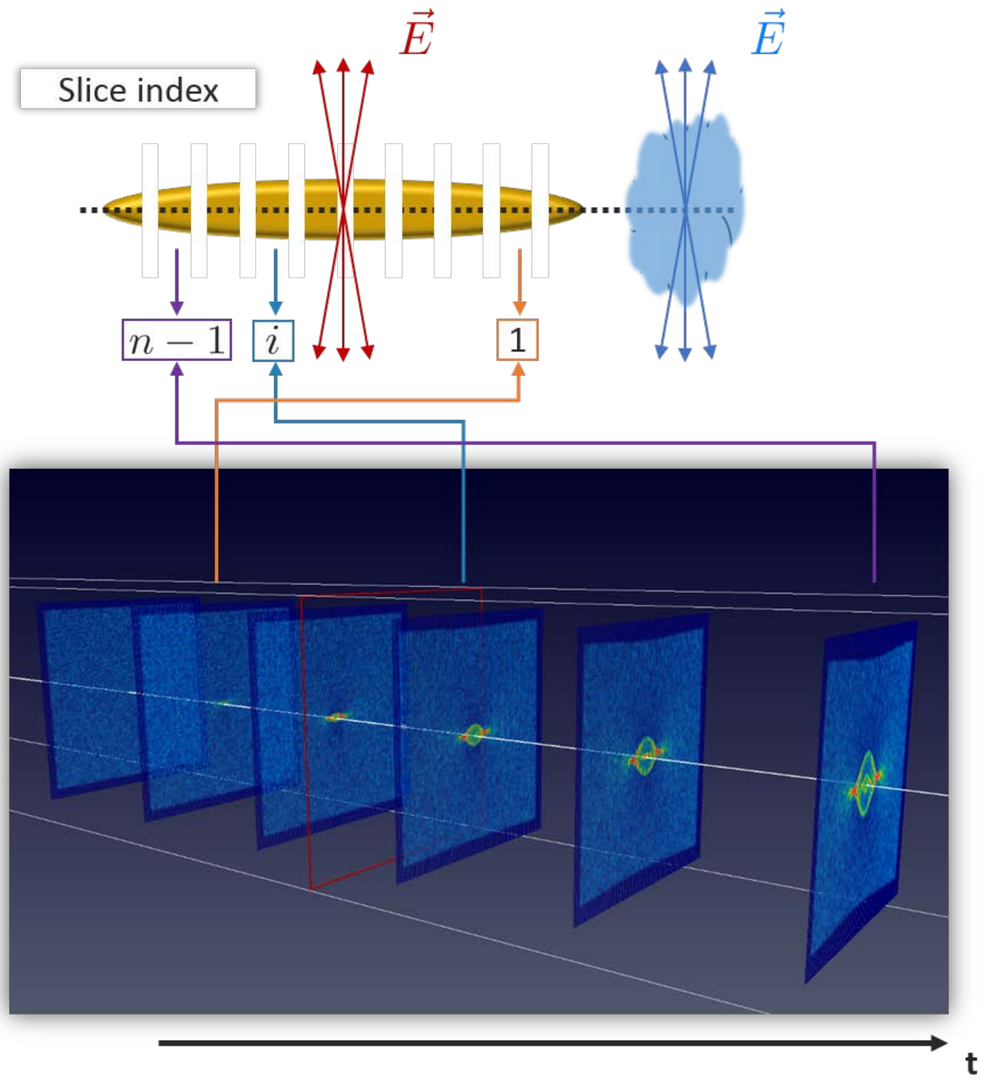
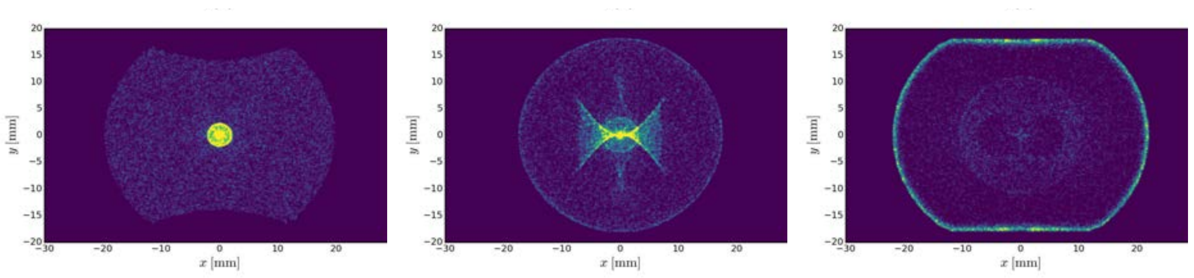


Fig. 24: The slice-by-slice interaction of an electron cloud with a passing bunch. Each plane pictures a snapshot of the current projected electron density distribution on the grid which is used for the field computation.

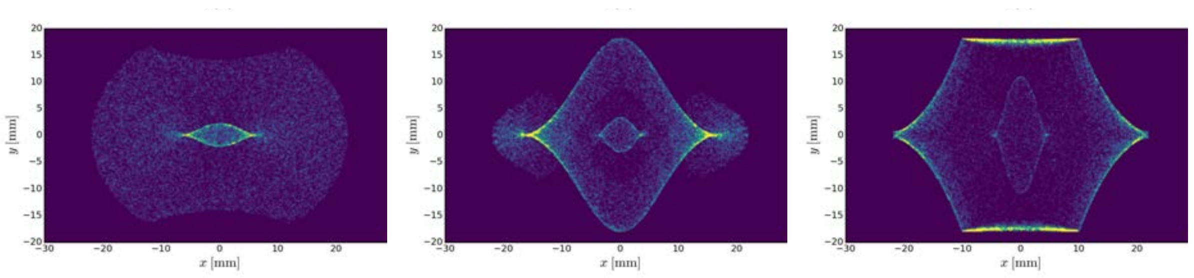
4.3 Combining build-up and instability simulations

Fully self-consistent electron cloud simulations require both build-up and instability simulation. Build-up simulations accurately model the electron dynamics and usually use a coarser model for the passing beam, making a weak–strong approximation, i.e., the beam is rigid and remains unaffected by the electron clouds. As we have just learned, in reality, this is not true and after several revolutions the beam may have significantly changed its original profile, which in turn changes the originally simulated build-up process and hence the stationary electron cloud distributions that evolve from this. The instability simulation, on the other hand, in addition to the electron dynamics also takes the detailed beam dynamics into account. Therefore, the correct approach would be to combine the two simulations in a way as outlined in Fig. 26.

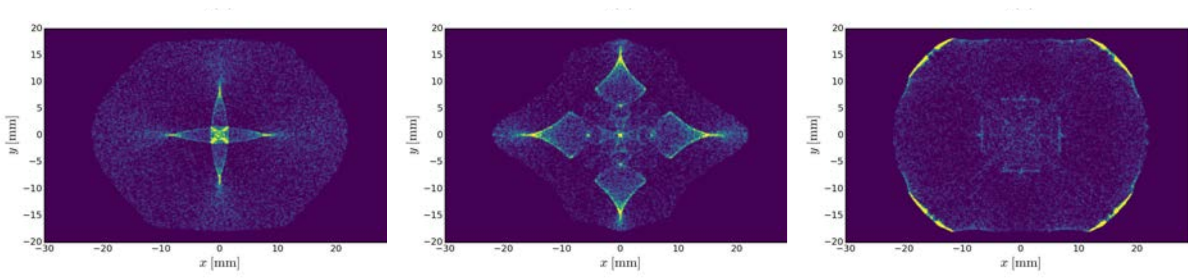
The problem is that the two simulations cover entirely different time-scales. The electron cloud build-up is usually completed within a single turn but requires multiple bunches for the multipacting to take place. The electron cloud instability, although it is relatively fast, can require several thousands of turns to grow. During this time, for a self-consistent description, multiple electron clouds together with multiple bunches would need to be stored and propagated at every time step. This cannot be easily



(a) Field-free region



(b) Dipole magnet



(c) Quadrupole magnet

Fig. 25: Electron cloud pinch during the passage of a charged particle beam within different magnetic field configurations. Sketched is the electron density evolution within an LHC-type beam screen. High-density regions are coloured light. The multipacting which takes place upon impact on the boundaries is well recognizable.

done in an efficient way and would require highly optimized code with parallel capabilities. Efforts are ongoing to overcome this limitation. To date, most of the instability simulations are carried out by assuming a given stationary electron cloud distribution at every bunch passage.

4.4 Application of electron cloud instability simulations

Electron clouds pose serious intensity limitations especially for circular machines with high brightness and closely spaced bunches. They deposit a considerable amount of heat onto the beam screens, which becomes a big issue especially for superconducting machines like the LHC. They reduce the dynamic aperture, which leads to beam quality degradation and poor lifetime. Finally, electron clouds also generate coherent instabilities. Whereas the heat loads can be evaluated by means of build-up simulations, the incoherent losses and coherent instabilities require instability simulations. These are mostly used to determine threshold values on the electron cloud density before beams go unstable due to electron clouds. Figure 27 shows an example of a typical electron cloud instability simulation. Electron clouds are initialized at different densities. For each density a separate simulation is run to evaluate the interaction of the electron cloud with the circulating beam. The emittance evolution of the circulating beam is observed for each case and the density threshold is set to the lowest value for which an exponential

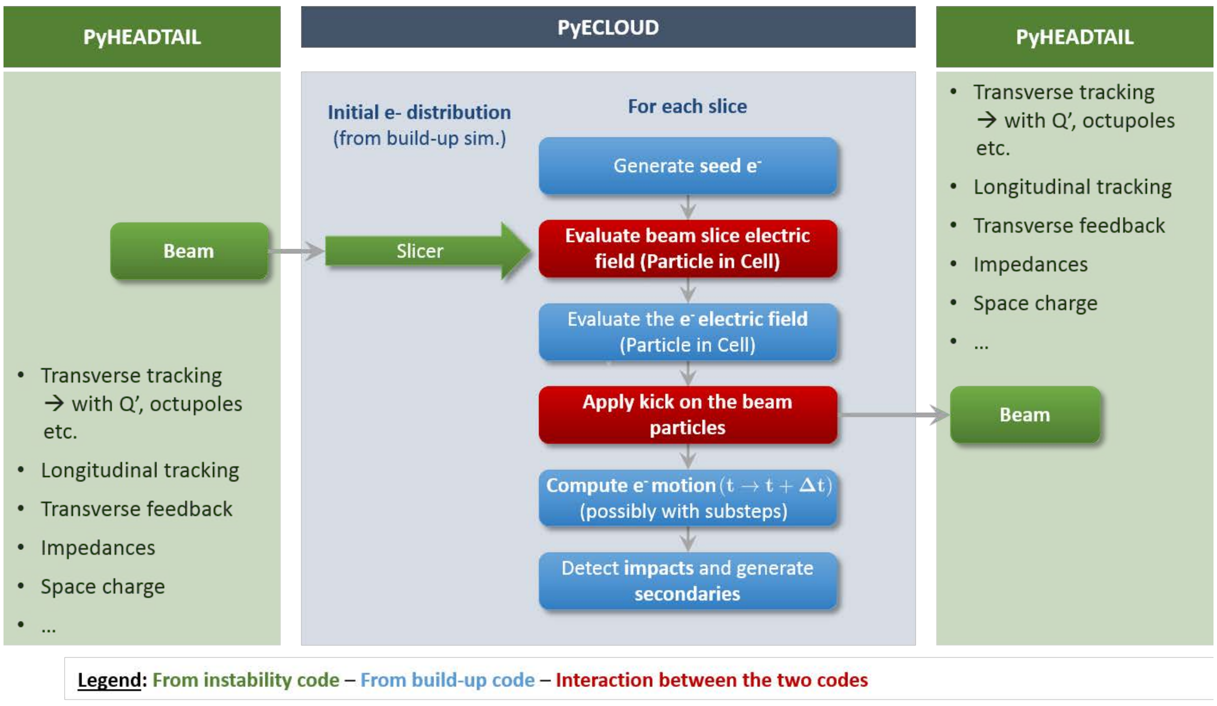

PyHEADTAIL
PyECLOUD
PyHEADTAIL
Initial e- distribution
(from build-up sim.)
For each slice
Generate seed e⁻
Beam
Slicer
Evaluate beam slice electric field (Particle in Cell)
Evaluate the e⁻ electric field (Particle in Cell)
Apply kick on the beam particles
Compute e⁻ motion (t → t + Δt) (possibly with substeps)
Detect impacts and generate secondaries
Transverse tracking → with Q', octupoles etc.
Longitudinal tracking
Transverse feedback
Impedances
Space charge
...
Beam
Legend: From instability code – From build-up code – Interaction between the two codes

Fig. 26: Scheme for a simulation strategy combining build-up (e.g. PyECLOUD) and instability (e.g. PyHEADTAIL) simulations. Of course this requires a good and modular code design.

growth of the emittance is observed for the first time. As the density is increased further the emittance grows more rapidly. For high densities the emittance tends to saturate. This is understood to be a combination of several effects linked to the strong growth in the beam size. In combination with the strong non-linearities of the electron cloud fields, the large beam accumulates a large tune spread causing it to strongly decohere and also at some point to be Landau damped, which is understood to be one of the reasons why the beam stabilizes. On the other hand, the large beam reduces the strength of the electron cloud pinch, which also renders the beam more stable. This can be seen for the orange curve in Fig. 27. Figure 28 shows the corresponding BPM signal that would be observed. There is a strong intrabunch motion that looks similar to what would be expected from a TMCI. After more turns this initially coherent signal decoheres, as can be seen more clearly in the 3D representation of the BPM signal in Fig. 28.

The instability threshold density can be compared to the multipacting threshold density expected for a given SEY. From this the beam stability can be inferred as well as its dependence on different SEYs. This technique was used for example to identify the most critical vacuum chambers in the SPS and prioritize them accordingly for potential amorphous carbon coating which can significantly lower the SEY of the inner surface of the vacuum chamber.

Another application which combines the build-up with the instability simulations is the investigation of single-bunch instabilities for multibunch batches in the LHC. This was studied in detail in Ref. [22]. From build-up simulations strong electron clouds are expected for the injection of batches with multiple bunches at a bunch spacing of 25 ns. Observations with the pickups from the transverse damper feature strong transverse instabilities towards the end of the batches. This is usually a clear indication for electron cloud activity, which is increased towards the end of a batch. To reproduce and confirm this effect with simulations, a build-up simulation is run for the injection of a single batch. The electron distribution is recorded just before each bunch's passage. This distribution is then imported into an instability simulation which is then performed for each of the corresponding bunches, respectively. Figure 29 shows the computed electron cloud line density as obtained by the build-up simulation. The red dots indicate the line density of the snapshots of the distribution loaded into the instability simulation.

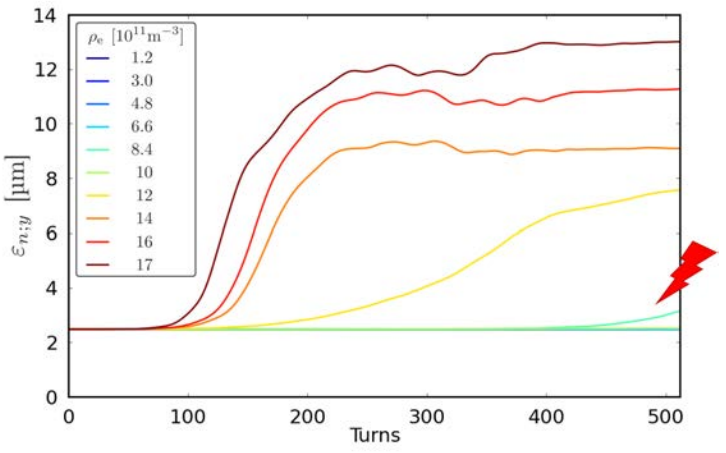


Fig. 27: Electron cloud instability simulation to determine the density threshold. The curves show the transverse emittance evolution for different electron cloud densities. The threshold is the lowest density for which an exponential growth of the emittance is observed over the given number of turns (here at $\rho_e \approx 8.5 \times 10^{11}$ m^{-3}).

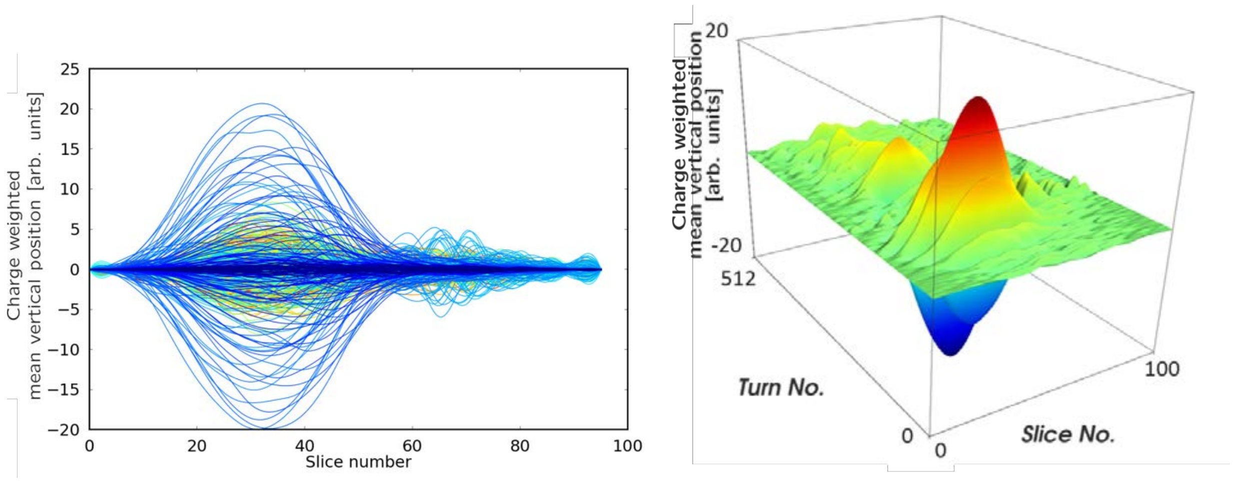


Fig. 28: The charge-weighted transverse offset as would be measured in a BPM pickup is shown for the case of $\rho_e \approx 14 \times 10^{11}\mathrm{m}^{-3}$. A strong coherent motion can be seen towards the tail of the bunch similar to what is observed for a TMCI. The motion decoheres, combined with an emittance blow-up after approximately 128 turns.

Figure 30 shows the comparison of the measurements and the simulations of the observed electron cloud instability. The instability occurs for exactly the same bunches in the measurement as in the simulations and the overall behaviour of the beam is well captured. This helped a lot in the understanding of both the the electron cloud build-up and instabilities in the LHC.

Appendix A: Brief introduction to the particle-in-cell algorithm

This section introduces briefly some of the basic concepts of the PIC algorithm. Essentially, it follows the same philosophy that we already encountered when introducing macroparticle models themselves. The physical particle systems were significantly reduced in the number of degrees of freedom by moving to macroparticle systems. This was done by clustering physical particles within a representative region in space into single macroparticles to obtain a numerical representation of the original physical particle system as a macroparticle system. The PIC algorithm now, instead of physical systems of discrete particles, targets physical systems of continuous fields. In the framework of the algorithm, the physical space

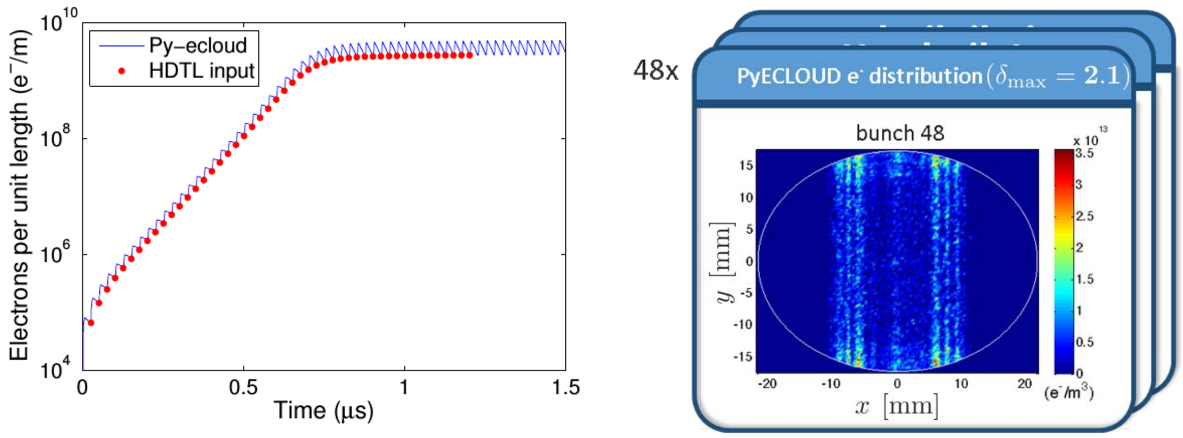


Fig. 29: Snapshots of the electron distributions are taken from a build-up simulation (blue) just before each bunch passage. The snapshots are then imported into an instability simulation (red) for each of the 48 corresponding bunches, respectively.

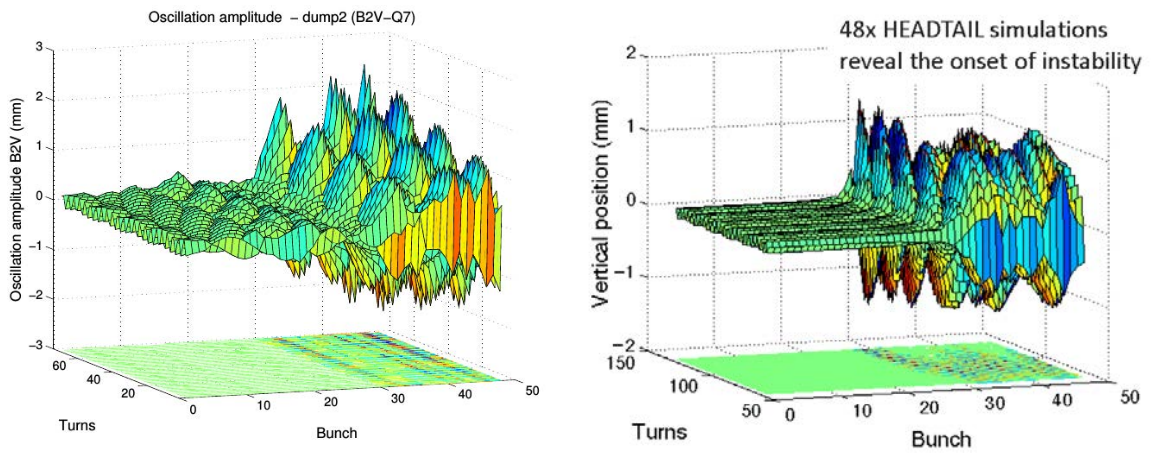


Fig. 30: Transverse oscillations measured with the pickups of the transverse feedback for 73 turns just before the beam dump (left). Bunch-by-bunch oscillations in the vertical plane as obtained from an instability simulation (right). In both cases the second half of the batch is rendered unstable by the electron cloud.

on which the fields are defined is now discretized into a set of regularly or irregularly distributed space points which is imbued with a topology by assigning to each space point the connectivity information to other space points. The set of space points together with the connectivity information forms a mesh. The space points are then called mesh nodes and the areas surrounded by connected sets of mesh nodes form mesh cells. Figure A.1 shows some examples of different discretizations of space with different types of meshes. The choice of the mesh type depends strongly on the problem to be analysed.

Computationally often even macroparticle systems are too big to efficiently handle the particle-to-particle interactions induced, for example, by electrostatic fields. One way to deal with this, then, is to perform a further step of discretization according to the PIC algorithm. In this particular case, of course, a computational gain is only made if the resulting number of space points is less than the number of macroparticles. It is then interesting to think about the gain that one can expect from computing the electrostatic forces particle-to-particle in comparison with computing them via a mesh. The electrostatic fields at the macroparticle positions $\mathbf{x}_k$ can be computed in a rather straightforward manner by summing

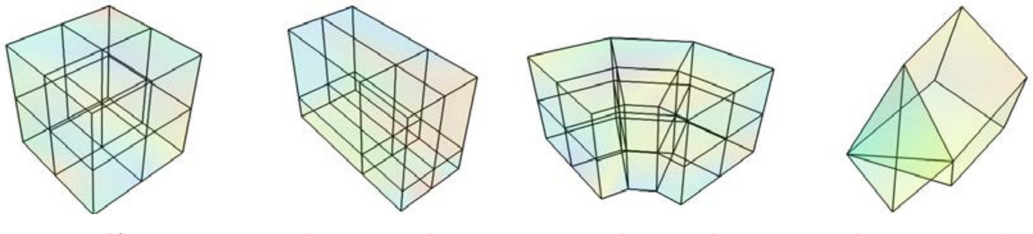


Fig. A.1: Different basic types of mesh commonly used to computationally represent domains in space (taken from Ref. [24]).

all forces from all other macroparticle positions $\mathbf{x}_l$,

$$\Delta \mathbf{x}_k = \sum_{l \neq k} \frac{e^2}{2\pi\varepsilon_0} \frac{1}{|\mathbf{x}_k - \mathbf{x}_l|}, \qquad k \in [0, \ldots, \text{macroparticle number}]. \tag{A1}$$

This sum needs to be evaluated for every macroparticle, which makes the algorithm scale as

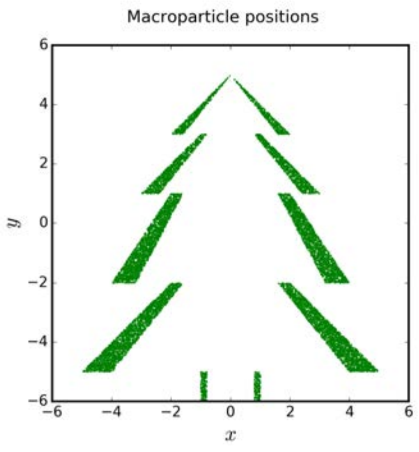
$$\frac{1}{2}\left(N^2_{\text{macroparticle}} - N_{\text{macroparticle}}\right).$$

Here, $N_{\text{macroparticle}}$ is the macroparticle number, which can be of the order of a few millions. Introducing mesh nodes and computing the forces on the mesh nodes instead reduces the number of operations according to the number of mesh nodes $N_{\text{mesh nodes}}$ proportionally. If we assume a macroparticle system of 1×10^6 macroparticles and compare this to a mesh of dimensions 128×128 then the number of operations reduces from $\sim 500 \times 10^9$ to $\sim 0.1 \times 10^9$ when moving from a particle-to-particle (p2p)- to a mesh node-to-mesh node (m2m)-based computation. Of course, for the latter there will be some overhead from the scattering and gathering of the macroparticles to the mesh nodes and vice versa. However, even this large reduction factor is often not enough to achieve satisfactory results, especially as the dimensions of the mesh increase. In addition, sometimes regions in space need to be computed that do not contain any macroparticles but still provide an important contribution to the electromagnetic fields. Here, the goal is not to reduce the number of mesh nodes compared to the number of macroparticles but rather to obtain a well-resolved discrete sampling of the regions of interest. And, finally, introducing boundary conditions makes the situation a lot more complicated. To satisfy the boundary conditions for arbitrarily shaped boundaries, in principle, an infinite number of image charges would need to be placed iteratively. For these reasons, other more efficient methods are used instead for solving the field equations.

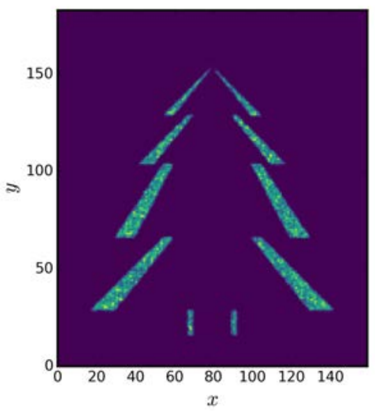
The PIC algorithm basically consists of three main steps.

- Macroparticles are scattered to the grid. This is done by depositing charges via an interpolation algorithm from the macroparticle positions to the locations of the mesh nodes.
- A potential is computed by solving the field equations on the mesh. There are many different ways for solving partial differential equations (PDEs) on a mesh and we will mention just a few below. Once the potential is known, the differential on the mesh yields the force fields.
- The force fields are then gathered from the mesh node locations back to the macroparticle positions. This must be done via the same interpolation that was used to scatter macroparticles to the mesh, otherwise the conservation of momentum will be violated.

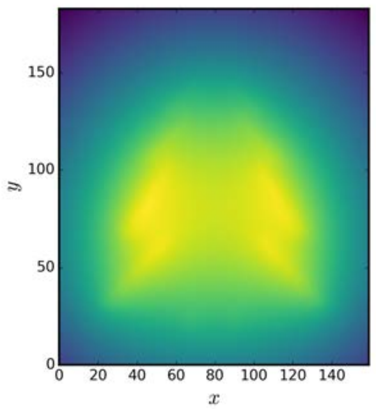
We will discuss these steps briefly below using the example of solving the 2D Poisson equation on a rectangular mesh with dimensions $N_x \times N_y$ within a bounded domain Ω and some given (Dirichlet)



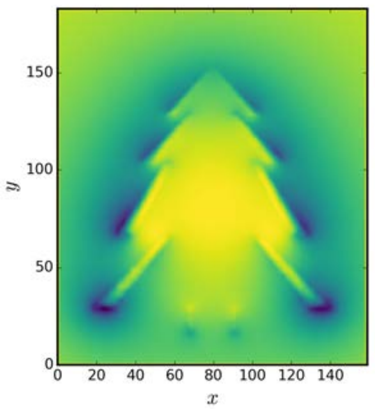
(a) Macroparticles



(b) Charge density



(c) Electrostatic potential



(d) Magnitude of the electric field

Fig. A.2: The different stages of the PIC algorithm, here for computation of the magnitude of the electrostatic fields from a given charge distribution in open space.

boundary conditions on $\partial\Omega$

$$\begin{aligned} \Delta\varphi &= -\frac{\rho}{\varepsilon_0} \qquad &&\text{in } \Omega, \\ \varphi &= b &&\text{on } \partial\Omega. \end{aligned} \tag{A2}$$

An excellent reference that explains all parts of the algorithm in detail is Ref. [23]. As an example, we may consider Fig. A.2, which shows the results of the different stages of the space-charge calculation for a rather arbitrary given charge distribution.

A.5 Macroparticle and field interpolation

To solve the Poisson equation on the mesh one first needs to know the charge density on the mesh. For this, the macroparticle charges are distributed (scattered) to the mesh nodes. There are several ways of doing this. One way is to assign the full macroparticle charge to the nearest mesh node. This is the nearest grid point (NGP) scheme, which is computationally very efficient but lacks resolution and suffers from high noise. A more refined variant is to assign weighted fractions of the macroparticle charge to all the surrounding mesh nodes within a given cell. This results in a linear interpolation of the macroparticle charge density to the mesh and is the cloud-in-cell (CIC) scheme. It offers a good compromise between computational effort and resolution and is often employed in many standard PIC codes. Higher order interpolation schemes can be systematically constructed depending on the constraints (i.e. long-range constraint, smoothness constraint or momentum conservation constraint) imposed on them. Generally, these interpolation schemes are characterized by a shape function S and the charge weighting is then generally given as (see Ref. [23], Section 5-3-4)

$$W_i(x) = \int_{x_i-h/2}^{x_i+h/2} S(x-x')\,\mathrm{d}x' = \int \Pi\left(\frac{x'}{h}\right) S(x-x')\,\mathrm{d}x', \tag{A3}$$

with h the mesh spacing and Π the 'top-hat' function

$$\Pi(x) = \begin{cases} 0, & |x| > \frac{1}{2}, \\ \frac{1}{2}, & |x| = \frac{1}{2}, \\ 1, & |x| < \frac{1}{2}. \end{cases}$$

We will look at an example for a 2D CIC scheme charge assignment on a regular rectangular mesh with constant mesh spacing h_x and h_y. The shape function for this scheme is given as

$$S(x) = \frac{1}{H}\,\Pi\left(\frac{x}{H}\right).$$

The charge of each macroparticle is distributed to the four neighbouring nodes of the mesh according to the CIC scheme to obtain a discrete representation of the charge distribution. In particular, for a macroparticle at the location (x, y) carrying charge q, the indices of the neighbouring grid nodes can be easily computed by means of a floor division (see Fig. A.3)

$$i = \text{floor}\left[\frac{x-x_0}{h_x}\right], \quad j = \text{floor}\left[\frac{y-y_0}{h_y}\right], \tag{A4}$$

where x_0, y_0 is the bottom left corner of the grid. Then the fractional normalized distances are given as (see Fig. A.3)

$$f_x = \frac{x-x_0}{h_x} - i, \quad f_y = \frac{y-y_0}{h_y} - j, \tag{A5}$$

and the charge density matrix is updated as

$$\begin{aligned} \tilde{\rho}_{i,j} &= \rho_{i,j} + \frac{q}{h_x\,h_y}\,(1-f_x)(1-f_y), \\ \tilde{\rho}_{i+1,j} &= \rho_{i+1,j} + \frac{q}{h_x\,h_y}\,f_x(1-f_y), \\ \tilde{\rho}_{i,j+1} &= \rho_{i,j+1} + \frac{q}{h_x\,h_y}\,(1-f_x)f_y, \\ \tilde{\rho}_{i+1,j+1} &= \rho_{i+1,j+1} + \frac{q}{h_x\,h_y}\,f_x\,f_y. \end{aligned} \tag{A6}$$

Once the fields have been obtained on the mesh, the inverse operation is performed in the same manner as the fields are gathered back to the macroparticle positions:

$$\mathbf{E}(x,y) = \mathbf{E}_{i,j}\,(1-f_x)(1-f_y) + \mathbf{E}_{i+1,j}\,f_x(1-f_y) + \mathbf{E}_{i,j+1}\,(1-f_x)f_y + \mathbf{E}_{i+1,j+1}\,f_xf_y. \tag{A7}$$

A.6 Field computation

Poisson solvers on discretized grids exist in many variants.

Finite-difference methods are used on structured grids where the continuous domain in space is replaced by a discrete set of points on a grid on which the electric and magnetic fields are computed.

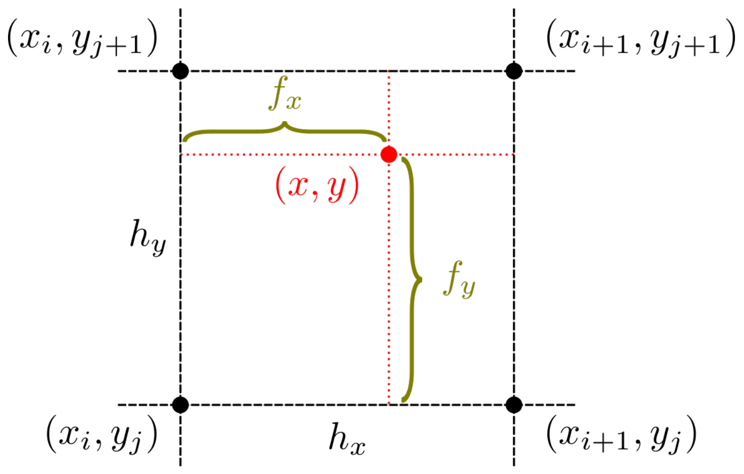


Fig. A.3: Nodes and distances involved in the macroparticle scatter and field gather stages of the space charge field calculation.

Derivatives are then approximated with differences between neighbouring grid point values. Thus, the differential Poisson equation is turned into an algebraic equation. We assume again a 2D rectangular grid, not necessarily uniform, that discretizes a domain Ω in the x and in the y directions into N_x and N_y subintervals, respectively, as

$$[h_{x,0},\, h_{x,1},\, \ldots,\, h_{x,N_x}]\,, \quad [h_{y,0},\, h_{y,1},\, \ldots,\, h_{y,N_y}]\,. \tag{A8}$$

Furthermore, we introduce what is known in the finite integration technique (FIT), which has been introduced by Weiland (see Ref. [25]), as the mesh spacing (edges) on the dual grid

$$\hat{h}_{x,i} = \begin{cases} \dfrac{h_{x,i-1} + h_{x,i}}{2}, & 0 < i < N_x, \\ \dfrac{h_{x,i}}{2}, & i = 0, N_x \end{cases} \tag{A9}$$

and equally for the y direction. The discretization of the second-order derivative with second-order finite differences in the general case gives

$$\frac{\partial^2 \varphi(x_i, y_j)}{\partial x^2} \approx \frac{\varphi(x_{i-1}, y_i)}{\hat{h}_{x,i} h_{x,i-1}} + \frac{2\varphi(x_i, y_i)}{h_{x,i-1} h_{x,i}} + \frac{\varphi(x_{i+1}, y_i)}{\hat{h}_{x,i} h_{x,i}}. \tag{A10}$$

Consequently, it follows that the discretization of the Poisson equation with second-order finite differences on the above-described non-equidistant grid leads to the following system of equations:

$$\begin{aligned} &\hat{h}_j \left(\frac{1}{h_{i-1}} \varphi_{i-1,j} - \left(\frac{1}{h_{i-1}} + \frac{1}{h_i} \right) \varphi_{i,j} + \frac{1}{h_i} \varphi_{i+1,j} \right) \\ &+ \hat{h}_i \left(\frac{1}{h_{j-1}} \varphi_{i,j-1} - \left(\frac{1}{h_{j-1}} + \frac{1}{h_j} \right) \varphi_{i,j} + \frac{1}{h_j} \varphi_{i,j+1} \right) \\ = &- \hat{h}_i \hat{h}_j \frac{\rho_{i,j}}{\varepsilon_0}, \end{aligned} \tag{A11}$$

where we now set $\varphi(x_i, y_j) = \varphi_{i,j}$ and $h_{x,i} = h_i$. The same system of equations is obtained with the application of the FIT. More details along with a generalization to higher dimensions and the treatment of boundaries—also refined, using the Shortley–Weller star—can be found in Ref. [26].

The grid spacings define the coefficients of the potential. These can be assembled in a matrix such that Eq. (A11) can be written in matrix form as

$$\boldsymbol{A}\vec{\varphi} = -\frac{1}{\varepsilon_0}\vec{\rho}. \tag{A12}$$

Here, $\boldsymbol{A}$ is the discrete Laplace operator, which is an $N_x\,N_y \times N_x\,N_y$ (usually sparse) matrix, and $\vec{\varphi}$ and $\vec{\rho}$ are vectors with entries ordered along the grid nodes according to the lexicographical rule

$$\{v_{1,1}, v_{2,1}, \ldots, v_{N_x,1}, v_{1,2}, v_{2,2}, \ldots, v_{N_x,2}, \ldots, v_{1,N_y}, v_{2,N_y}, \ldots, v_{N_x,N_y}\}, \quad v = \varphi, \rho.$$

By solving this system of linear equations it is possible to obtain the electrostatic potential at the nodes of the grid. The electric fields at the internal nodes of the grid can then be computed easily using the central difference formula

$$\begin{aligned}(E_x)_{i,j} &= -\left(\frac{1}{2h_i}\varphi_{i+1,j} + \left(\frac{1}{2h_{i-1}} - \frac{1}{2h_i}\right)\varphi_i - \frac{1}{2h_{i-1}}\varphi_{i-1,j}\right),\\ (E_y)_{i,j} &= -\left(\frac{1}{2h_j}\varphi_{i,j+1} + \left(\frac{1}{2h_{j-1}} - \frac{1}{2h_j}\right)\varphi_j - \frac{1}{2h_{j-1}}\varphi_{i,j-1}\right).\end{aligned} \tag{A13}$$

For boundary nodes, a forward difference formula is adopted.

There are many different types of solvers available to solve Eq. (A11) and the optimal solver to be employed depends mostly on the structure of the Laplace matrix $\boldsymbol{A}$. If the matrix is very sparse and if the geometry and the grid do not change in time, a LU decomposition (where 'LU' stands for 'lower upper', and also called LU factorization, see e.g., Ref. [27]), which is a direct solver, can be very efficient. This decomposition needs to be done only once for the Laplace matrix $\boldsymbol{A}$ and the potential can be found from there on for any charge density by forward and backward substitution.

Iterative solvers can become interesting for very large scale problems. They use less memory and are often well suited for parallelization. Iterative solvers start with an initial guess solution and continue iterations until a stopping criterion is satisfied (typically that the error or residual is less than a given tolerance). They then return the final guess solution. For the fast iterative solution of the system there are many alternative algorithms depending on the properties of the Laplace matrix $\boldsymbol{A}$ (primarily symmetry and definiteness). The most straightforward iterative methods are the relaxation-type methods. Typical examples are the Jacobi, Gauss–Seidel and SOR (successive over relaxation) algorithms. These classical iteration methods, also known as stationary methods, however, lack speed in comparison with the more modern non-stationary methods such as the Krylov-subspace methods like CG (conjugate gradient), BiCG (bi-conjugate gradient) and its derivatives, the minimal residual algorithms like the GMRES (generalized minimal residual) or hybrid methods like the BiCGSTAB (bi-conjugate gradient stabilized) [28].

Another often employed and very efficient way of solving the Poisson equation for open boundaries, in particular, employs Green's functions in combination with the fast Fourier transform (FFT) algorithm. This method becomes interesting for situations where boundaries are far away from any source charge distributions up to the limit where open boundaries can be assumed. The 2D electrostatic potential of a point charge in open space is simply given by the Green's function of the 2D Laplacian

$$G(x - x_0, y - y_0) = \frac{1}{2\pi}\left((x - x_0)^2 + (y - y_0)^2\right)^{1/2}. \tag{A14}$$

Then the electrostatic potential can be computed at any point (i, j) on the grid by means of the sum

$$\begin{gathered}\varphi_{ij} = \frac{e}{\varepsilon_0}\sum_{m,n\neq i,j}\rho_{i,j}\,G(x_i - x_m, y_j - y_n),\\ i, j, m, n \in [0, \ldots, \text{grid point number} - 1].\end{gathered} \tag{A15}$$

Equation (A15), which runs over all grid points, is computationally often already significantly cheaper compared to Eq. (A1), which, instead, runs over all macroparticles (both are of order $\mathcal{O}(N^2)$, where N is the grid point number or the macroparticle number, respectively). Nevertheless, also the number of grid points can be high and the computation of the potential can be made yet more efficient by using another algorithm. This makes use of the convolution theorem, which states that a convolution in the time domain reduces to a product in the frequency domain, where Eq. (A15), instead, reads

$$\begin{aligned} \hat{\varphi}_{ij} &= \frac{e^2}{2\pi\varepsilon_0}\,\hat{\rho}_{i,j}\cdot\hat{G}_{i,j}, \\ i,j &\in [0,\ldots,\text{grid point number}-1]. \end{aligned} \tag{A16}$$

Equation (A16) scales linearly with the number of grid points (i.e. $\mathcal{O}(N)$). A Fourier transform still needs to be performed on the charge density ρ and the Green's function G as well as an inverse Fourier transform to obtain the potential φ in the time domain. This produces some overhead, which, finally, when employing the FFT, leads to a scaling of the algorithm as $\mathcal{O}(N\log N)$.

The algorithm produces the correct potential for open boundaries by exploiting certain properties of the FFT algorithm. To implement the algorithm correctly, therefore, the computational domain needs to be extended to double its original size along each dimension, i.e. in 2D the area is fourfold. The charge density is left as it is in its original quadrant and set to zero in all other quadrants. The Green's function is mirrored from one quadrant to another—this is required due to the particular way the FFT algorithm works. The multiplication is then done in the frequency domain and the inverse FFT is made to obtain the potential on the extended grid. The potential in the original quadrant is the potential generated by the given charge distribution in an open space whereas the potential in all other quadrants is unphysical, but also not of interest and can, hence, be discarded. This algorithm was first proposed by Hockney [23] and has since then been used for many problems in accelerator physics, e.g. [29].

As a final remark, we mention that using the Green's function for a point charge in open space can be problematic for grids featuring large aspect ratios. The Green's function for a point charge in this case does not well combine with the charge interpolation to the grid points using the commonly employed CIC method. While the Green's function takes the perspective of point-centred charges, the CIC method assumes uniformly distributed charges over one grid cell. Combining the two perspectives will lead to a poor representation of the forces within the grid cells. Especially, along the dimension sampled at lower frequency, the short-range nature of the point-charge Green's function will cause the resulting forces to decay rapidly, when the averaged force within one grid cell would actually not do so. For this reason, a more robust way of computing the forces is to use the integrated Green's function, which essentially is the point-charge Green's function integrated over one grid cell [29]. In 2D the integrated Green's function reads

$$G(x,y) = -\frac{1}{4\pi}\left(-3\,xy + xy\,\log(x^2+y^2) + x^2\arctan\left(\frac{y}{x}\right) + y^2\arctan\left(\frac{x}{y}\right)\right). \tag{A17}$$

Other means of solving the Poisson equation (or more complex PDEs) include finite-element methods which are used preferably on unstructured grids for complicated geometries where the continuous domain is divided into a discrete mesh of elements. The Poisson equation is treated as an eigenvalue problem and initially a trial solution is calculated using basis functions that are localized in each element. The final solution is then obtained by optimization until the required accuracy is reached. We will not discuss this method further here. There is a lot of literature available for the interested reader to embark further into these types of numerical methods.

Acknowledgments

I would like to thank L.R. Carver, G. Iadarola, G. Rumolo and M. Schenk for their help and their valuable input as well as for checking the material. Also, I would like to thank L. Mether, G. Rumolo and M. Schenk for proofreading this document.

References

[1] Y.H. Chin, User's guide for new MOSES version 2.0, CERN/LEP-TH/88-05 (1988).

[2] N. Mounet, DELPHI: an analytic Vlasov solver for impedance-driven modes, HSC Section Meeting, May 2014.

[3] X. Buffat, Transverse beams stability studies at the Large Hadron Collider, CERN-THESIS-2014-246 (2015).

[4] M. Ferrario, HOMDYN (higher order modes dynamics), Workshop on High Average Power & High Brightness Beams, Working Group A, Injector Design, November 2004.

[5] G. Rumolo and F. Zimmermann, Practical user guide for ECLOUD, SL-Note-2002-016-AP (2002).

[6] G. Rumolo and F. Zimmermann, Practical user guide for HEADTAIL, SL-Note-2002-036-AP (2002).

[7] R.D. Ruth, Single particle dynamics in circular accelerators, SLAC-PUB-4104 (1986).

[8] E.D. Courant and H.S. Snyder, *Ann. Phys.* **3** (1958) 1. http://dx.doi.org/10.1016/0003-4916(58)90012-5

[9] S.Y. Lee, *Accelerator Physics*, 3rd ed. (London: World Scientific, 2011). http://dx.doi.org/10.1142/8335.

[10] W. Herr, *Tools for Non-linear Dynamics* (CAS, Poland, 2015).

[11] E. Forest, *Beam Dynamics, A New Attitude and Framework* (Sydney: Harwood, 1998).

[12] A.W. Chao, *Physics of Collective Beam Instabilities in High Energy Accelerators*, 1st ed. (Wiley-VCH, 1993).

[13] S. Berg and F. Ruggiero, Landau damping with two-dimensional betatron tune spread, CERN-SL-AP-96-71-AP (1996).

[14] S. Berg and F. Ruggiero, Stability diagrams for Landau damping, LHC Project Report 121 (1997).

[15] W. Herr, in Proceedings of the CAS-CERN Accelerator School: Advanced Accelerator Physics Course, Trondheim, Norway, 18-29 August 2013, edited by W. Herr, CERN-2014-009 (CERN, Geneva, 2014), pp. 377-404, http://dx.doi.org/10.5170/CERN-2014-009.377

[16] C. Zannini *et al.*, Effects of an asymmetric chamber on the beam coupling impedance, Proc. IPAC2012, 2012.

[17] H. Bartosik, Beam dynamics and optics studies for the LHC injectors upgrade, CERN-THESIS-2013-257 (2013).

[18] G. Iadarola, Electron cloud studies for CERN particle accelerators and simulation code development, CERN-THESIS-2014-047 (2014).

[19] G. Rumolo, Electron cloud, these proceedings, 2016.

[20] C. Birdsall and A. Langdon, *Plasma Physics Via Computer Simulation* (McGraw-Hill, New York, 1985).

[21] H. Qin *et al.*, *Phys. Plasmas* **20** (2013) 084503.

[22] H. Bartosik *et al.*, Benchmarking HEADTAIL with electron cloud instabilities observed in the LHC, Proc. Fifth Electron-cloud Workshop, 2012.

[23] R.W. Hockney and J.W. Eastwood, *Computer Simulation using Particles* (CRC Press, 1988). http://dx.doi.org/10.1887/0852743920

[24] Kitware Inc., The Visualization Toolkit (VTK) data model, 2016, http://www.vtk.org/data-model/.

[25] T. Weiland, *Archiv für Elektronik und Übertragungstechnik* **31** (1977) 116.

[26] A. Markoviḱ, Numerical computation of space-charge fields of electron bunches in a beam pipe of elliptical shape, TESLA Report 2005-21 (2005).

[27] X.S. Li, *ACM Trans. Math. Software* **31**(3) (2005) 302. http://dx.doi.org/10.1145/1089014.1089017

[28] U. van Rienen, *Numerical Methods in Computational Electrodynamics. Linear Systems in Practical Application*, Vol. 12 of *Lecture Notes in Computational Science and Engineering* (Springer, Berlin Heidelberg, 2001). http://dx.doi.org/10.1007/978-3-642-56802-2

[29] J. Qiang *et al.*, *J. Comput. Phys.* **198** (2004) 278. http://dx.doi.org/10.1016/j.jcp.2004.01.008