Beam Tracking I & II

· *Longitudinal Tracking in Synchrotrons* ·

Helga Timko
CERN, SY-RF Group

RF CAS, Berlin, 23rd June 2023

With many thanks to the entire BLonD user and developer team

Content

Beam tracking basics

- Tracking of beams
- Discretisation

Longitudinal tracking

- Reference frame
- Equations of motion
- Periodicity

Effects on particle energy

- Collective effects and impedance
- Multi-turn wake
- Synchrotron radiation

RF modelling

- Tune and beam loading
- Global and local control loops

Particle distribution

- Observables and parametrisation
- Matching distributions

6D effects

- Some examples

Code design, optimisation and benchmarks

- Good practises

Content

Beam tracking basics

- Tracking of beams
- Discretisation

Longitudinal tracking

- Reference frame
- Equations of motion
- Periodicity

Effects on particle energy

- Collective effects and impedance
- Multi-turn wake
- Synchrotron radiation

RF modelling

- Tune and beam loading
- Global and local control loops

Particle distribution

- Observables and parametrisation
- Matching distributions

6D effects

- Some examples

Code design, optimisation and benchmarks

- Good practises

Introduction

Particle-in-cell method

- Developed in the 1950's with the first computers for plasma simulations by Buneman [1] and Dawson [2]
- Calculates EM fields on a grid in a Eulerian (stationary) frame, particles in continuous phase space in a Lagrangian (moving) frame
  - Many integrators (implicit and explicit) have been developed to solve the particle equations of motions
  - Many field solvers (like finite difference and finite element methods) have been developed too
- Collective effects are often calculated particle by particle with Monte-Carlo (random) algorithms
  - Collisions/interactions have N_p^2 complexity

Beam simulations

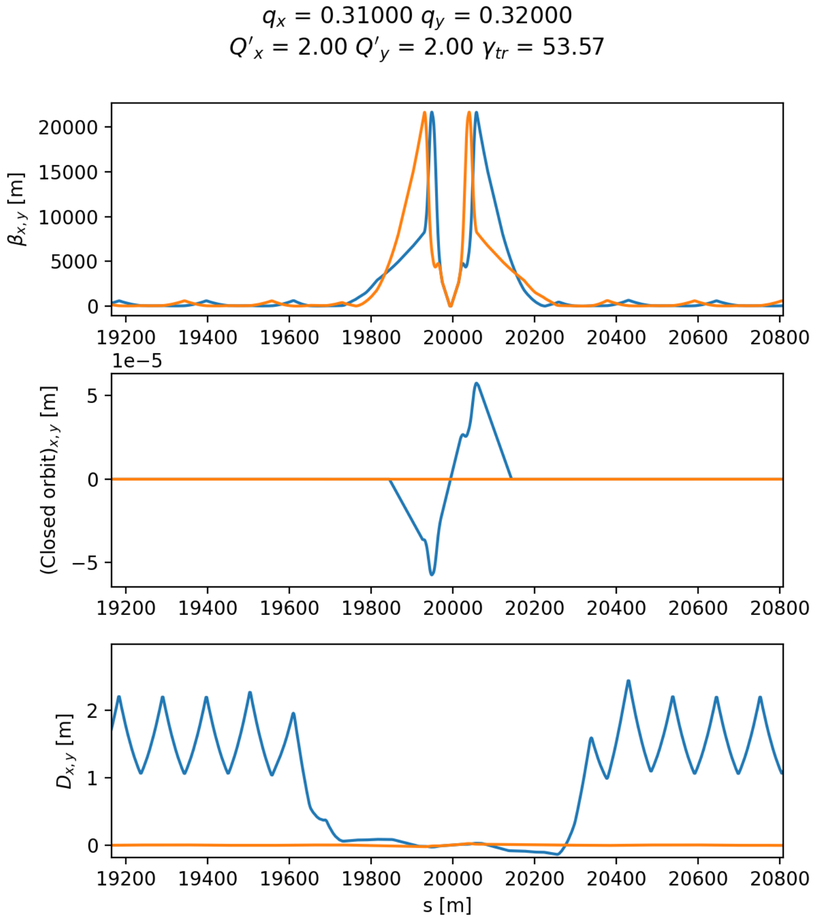
- Beam is essentially a single-component plasma in a strong external RF field
- It also calculates the fields on a grid and the particles in continuous phase space
- Simplifications can be done for collective effects

[1] O. Buneman: 'Dissipation of currents in ionised media', Phys Rev. **115**, 1959.
[2] J. Dawson: 'One-dimensional plasma model', Phys. Fluids **5**, 445, 1962.

Accelerator model (1)

Picture the accelerator and its components as it is in the real machine

- The transverse plane can be parametrised with a lattice description of dipole, quadrupole, and higher-order magnets (e.g. MAD-X [3])
- The lattice determines:
  - Tunes, chromaticities, slip factor, twiss parameters…
  - For longitudinal applications, determines the phase slippage or **drift equation**
- Transverse simulations also model collective effects in the transverse plane
  - Transverse impedance and its effect on the beam (coupled-bunch phenomena, head-tail motion, etc.)



Example for LHC parameters from XSuite [4]

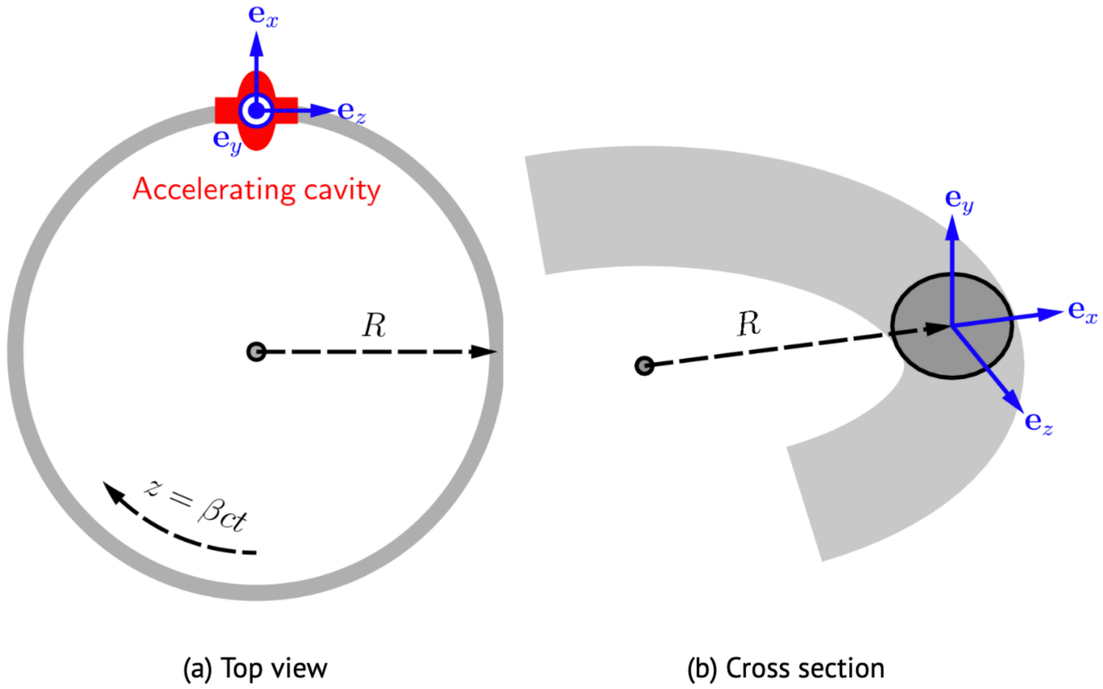
[3] MAD-X simulation suite, CERN, http://madx.web.cern.ch/madx

[4] XSuite simulation suite, CERN, https://github.com/xsuite, https://xsuite.readthedocs.io

Accelerator model (2)

Picture the accelerator and its components as it is in the real machine

- The longitudinal tracking determines the evolution of:
  - RF frequency and voltage amplitude and phase,
  - The change in synchronous energy following a magnetic ramp,
  - Collective effects due to longitudinal impedance,
  - Other effects like synchrotron radiation,
  - I.e. the evolution of the particle energy or **kick equation**
- Some phenomena cannot be divided into purely longitudinal or transverse problems and require 6D modelling
  - Intra-beam scattering (IBS), electron-cloud effects, etc.

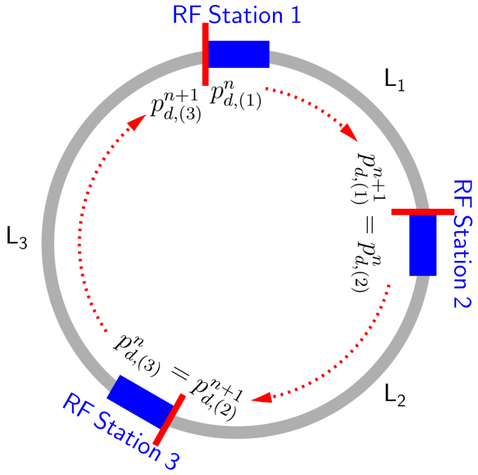


Longitudinal and transverse coordinates in a synchrotron

Discretisation

Basic time step

- Turn-by-turn mapping
  - For the energy kick given by the RF cavity
    - The cavity is assumed to be pointlike, i.e. the kick is integrated over the passage through the cavity
    - The energy kick is otherwise truly discrete
  - For the time/phase drift over the ring
    - An average slippage is assumed, as given by lattice simulations
- Sub-cycling might be needed for several RF stations or large changes per turn
  - Example: machines with large synchrotron radiation (e.g. FCC)
    - Several tracking sequences (kick-drift) have to be made per turn
    - In this case, impedance and slippage might be different in different sections of the ring

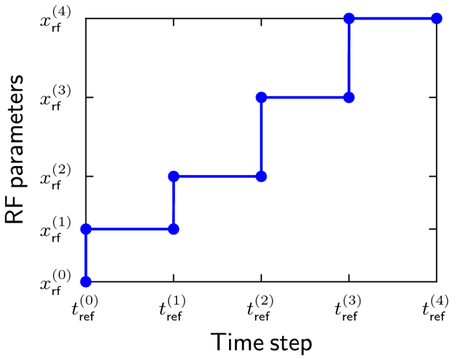


Example for a longitudinal ring model from BLonD [5]

Discretisation of variables

Step-wise constant or interpolation schemes

- Depends on the application and the rate of change



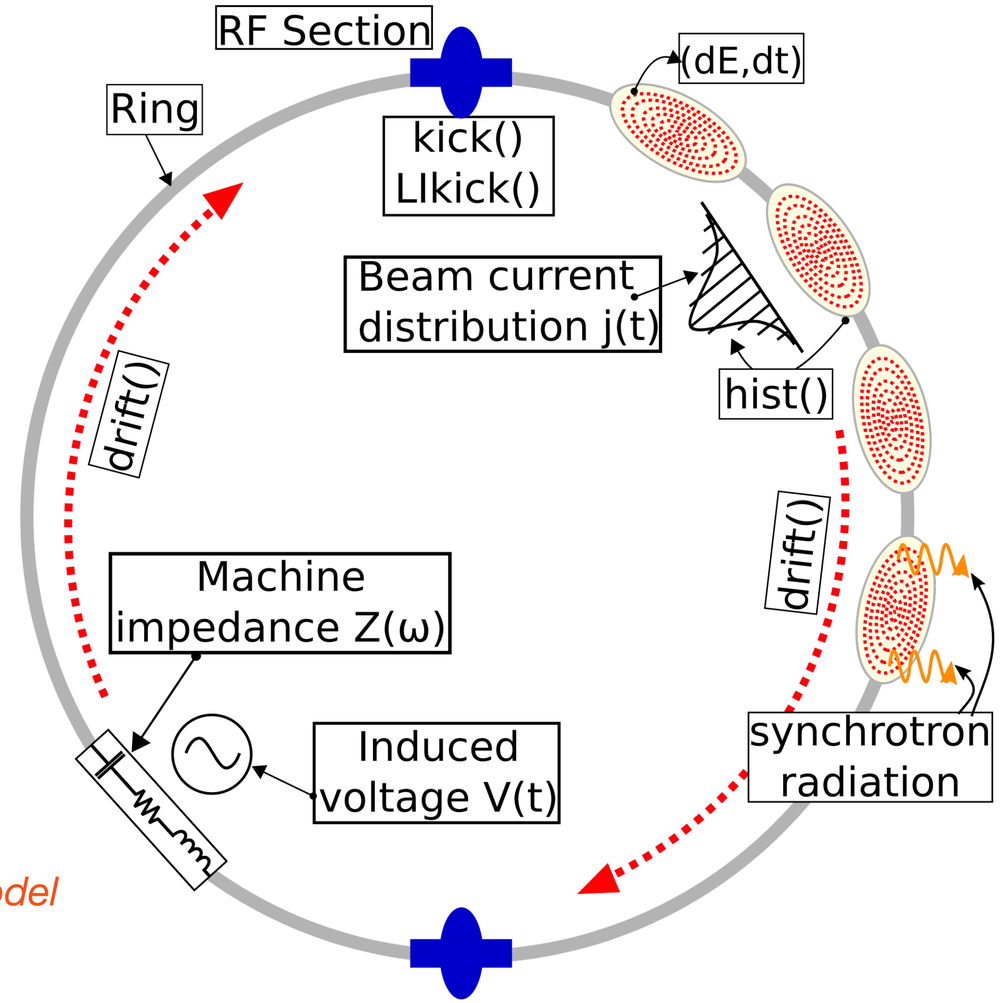
Step-wise discretisation of RF parameters

[5] BLonD simulation suite, CERN, https://github.com/blond-admin/BLonD, http://blond-admin.github.io

Building a full longitudinal model

Some components

- Generating bunches
- Basic equations of motion
  - Kick and drift
  - One or several RF stations providing the RF voltage kick
- Interaction with machine impedance
  - Beam line density (profile) calculation
  - Induced voltage calculation
- Other effects on particle energy
  - E.g. synchrotron radiation
- Etc.



Example of a synchrotron model

Courtesy of K. Iliakis

Content

Beam tracking basics

- Tracking of beams
- Discretisation

Longitudinal tracking

- Reference frame
- Equations of motion
- Periodicity

Effects on particle energy

- Collective effects and impedance
- Multi-turn wake
- Synchrotron radiation

RF modelling

- Tune and beam loading
- Global and local control loops

Particle distribution

- Observables and parametrisation
- Matching distributions

6D effects

- Some examples

Code design, optimisation and benchmarks

- Good practises

Reference clock

The basic time step of reference: revolution period

- Snapshot of the system is taken at this instance
  - Design orbit/energy/magnetic field
  - Radio-frequency parameters
  - Beam coordinates
  - …
- The reference or design clock is therefore defined by

$$t_{d,(0)} \equiv 0 \quad \text{and} \quad t_{d,(n)} \equiv \sum_{k=1}^{n} T_{\mathrm{rev},(k)} \ \text{ for } \ n \geq 1$$

- where the revolution period is defined by

$$T_{\mathrm{rev},(n)} = \frac{2\pi R_d}{\beta_{d,(n)} c}$$

- and the design momentum is tied to the design magnetic field

$$p_{d,(n)} = |q| \rho B_{d,(n)}$$

β_d	relativistic beta from design momentum
ρ	bending radius of magnets
B_d	design dipole magnetic field
c	speed of light
R_d	design orbit
q	particle charge
T_{rev}	revolution period

Longitudinal coordinates

Choose convenient coordinate pairs for your application

- Some examples of frequently used conjugate variables are [6] $(\varphi, \Delta E/\omega_{\mathrm{rev}})$ and (φ, δ)
  - Resulting in the kick and drift equations of

$$\dot{\varphi} = \frac{h\omega_{\mathrm{rev}}^2\eta}{\beta_d^2 E_d}\left(\frac{\Delta E}{\omega_{\mathrm{rev}}}\right) = h\omega_{\mathrm{rev}}\eta\delta$$

$$\frac{d}{dt}\left(\frac{\Delta E}{\omega_{\mathrm{rev}}}\right) = \frac{\beta_d^2 E_d}{\omega_{\mathrm{rev}}}\dot{\delta} = \frac{eV}{2\pi}(\sin\varphi - \sin\varphi_d)$$

  - Described by the Hamiltonians of

$$H\left(\varphi, \frac{\Delta E}{\omega_{\mathrm{rev}}}\right) = \frac{1}{2}\frac{h\eta\omega_{\mathrm{rev}}}{\beta_d^2 E_d}\left(\frac{\Delta E}{\omega_{\mathrm{rev}}}\right)^2 + \frac{eV}{2\pi}\{\cos\varphi - \cos\varphi_d + (\varphi - \varphi_d)\sin\varphi_d\}$$

$$H\left(\varphi, \delta\right) = \frac{1}{2}h\eta\omega_{\mathrm{rev}}\delta^2 + \frac{\omega_{\mathrm{rev}}eV}{2\pi\beta_d^2 E_d}\{\cos\varphi - \cos\varphi_d + (\varphi - \varphi_d)\sin\varphi_d\}$$

$\Delta E = E - E_d$	relative particle energy
$\delta = \frac{\Delta p}{p_d} \simeq \frac{\Delta E}{\beta_d^2 E_d}$	relative momentum offset
$\eta = \eta(\delta) = \eta_0 + \eta_1\delta + \eta_2\delta^2 + \ldots$	slippage factor
φ	RF voltage phase at particle arrival
φ_d	phase of design particle
e	elementary charge
E	particle energy
E_d	energy of design particle
E	particle energy
h	harmonic number
V	RF voltage amplitude

Tip: use dimensionless or scaled coordinates in optimised code whenever you can

- Restrict to the optimal ranges of the numeric functions used (e.g. $(0,2\pi)$ or $(-\pi, \pi)$ in trigonometric functions)
- For 6D simulations, $z = \beta_d ct$ is often a preferred choice

[6] S. Y. Lee: 'Accelerator Physics', World Scientific, 3rd Ed., 2012.

A side note on frequency slippage

The frequency slippage of an off-momentum particle is defined w.r.t. reference (synchronous) particle [6]

- The momentum compaction and the slippage factors are related as

$$\frac{\Delta\omega}{\omega_{\mathrm{rev}}} = \frac{\omega - \omega_{\mathrm{rev}}}{\omega_{\mathrm{rev}}} \equiv -\eta(\delta)\delta = -(\eta_0 + \eta_1\delta + \eta_2\delta^2 + \dots)\delta$$

- The momentum compaction and the slippage factors are related as

$$\eta_0 = \alpha_0 - \frac{1}{\gamma_d^2} = \frac{1}{\gamma_T^2} - \frac{1}{\gamma_d^2}$$

$$\eta_1 = \frac{3\beta_d^2}{2\gamma_d^2} + \alpha_1 - \alpha_0\eta_0$$

$$\eta_2 = -\frac{\beta_d^2(5\beta_d^2 - 1)}{2\gamma_d^2} + \alpha_2 - 2\alpha_0\alpha_1 + \frac{\alpha_1}{\gamma_d^2} + \alpha_0^2\eta_0 - \frac{3\beta_d^2\alpha_0}{2\gamma_d^2}$$

- Interpretation e.g. below transition: for $\gamma < \gamma_T \rightarrow \eta_0 < 0$
  - Particles with higher energy $\delta > 0$ do one turn faster $\Delta\omega > 0$

$\Delta E = E - E_d$ relative particle energy

$\delta = \frac{\Delta p}{p_d} \simeq \frac{\Delta E}{\beta_d^2 E_d}$ relative momentum offset

[6] S. Y. Lee: ‘Accelerator Physics’, World Scientific, 3rd Ed., 2012.

A side note on linear accelerators

Essentially RF machines

- RF potential and induced voltage can be described in a similar manner
- On the other hand, one particle passes one RF structure only onces
  - No turn-by-turn tracking
  - Rather a mapping from one location to another in the machine
- Governed by different EOMs
  - Arrival time of the particle w.r.t. the RF wave still a key concept

Synchrotron vs linac: main differences

- In a synchrotron, cavities are treated as point-like objects, while in linacs the time traversing a cavity needs to be taken into account
- In a synchrotron, the energy and beta change of a particle is small per turn, while in linacs the rate of energy gain is crucial to describe as a function of the longitudinal coordinate

Equations of motion

Convention used in these lectures: as in the Beam Longitudinal Dynamics simulation suite BLonD

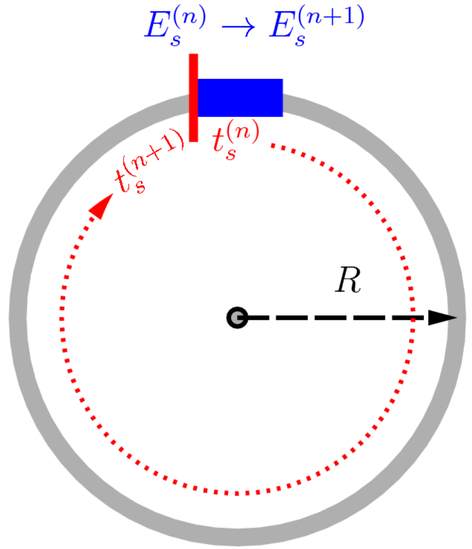
- Deviations from the reference particle in time and energy
  - Mapping from $(\Delta t_{(n)}, \Delta E_{(n)}) \rightarrow (\Delta t_{(n+1)}, \Delta E_{(n+1)})$,
    - Where $\Delta t_{(n)} \equiv t_{(n)} - t_{d,(n)}$ and $\Delta E_{(n)} \equiv E_{(n)} - E_{d,(n)}$
  - Make a choice which coordinate to update first (arbitrary)
  - Kick equation

$$\Delta E_{(n+1)} = \Delta E_{(n)} + \sum_{k=1}^{n_{\mathrm{rf}}} qV_{k,(n)} \sin(\omega_{\mathrm{rf},k,(n)} \Delta t_{(n)} + \varphi_{\mathrm{rf},k,(n)}) - (E_{d,(n+1)} - E_{d,(n)}) + E_{\mathrm{other},(n)}$$

Multiple RF systems in one location

Change of design energy (magnetic field)

All other effects on energy (e.g. induced voltage)



  - Drift equation

$$\Delta t_{(n+1)} = \Delta t_{(n)} + T_{\mathrm{rev},(n+1)} \left[\left(1 + \alpha_{0,(n+1)} \delta_{(n+1)} + \alpha_{1,(n+1)} \delta_{(n+1)}^2 + \alpha_{2,(n+1)} \delta_{(n+1)}^3 \right) \frac{1 + \frac{\Delta E_{(n+1)}}{E_{d,(n+1)}}}{1 + \delta_{(n+1)}} - 1 \right]$$

Synchronism

Why “reference” time/energy/particle and not “synchronous” time/energy/particle?

- Condition to be a synchronous particle when tracking from time step $n \rightarrow n+1$
  - Enter and exit the cavity with exactly the design energy

    $\Delta E_{(n)} = \Delta E_{(n+1)} = 0$, or equivalently, $E_{(n)} = E_{d,(n)}$ and $E_{(n+1)} = E_{d,(n+1)}$
  - This reduces the drift equation to

    $\Delta t_{(n+1)} = \Delta t_{(n)} + 0$
- Consider the simple example of particle motion without intensity effects and constant RF parameters
  - The synchronous particle in two consecutive turns has to fulfil the following kick equations:

    $$\sum_{k=1}^{n_{\mathrm{rf}}} qV_k \sin(\omega_{\mathrm{rf},k}\Delta t_{(n)} + \varphi_{\mathrm{rf},k}) = E_{d,(n+1)} - E_{d,(n)}$$

    $$\sum_{k=1}^{n_{\mathrm{rf}}} qV_k \sin(\omega_{\mathrm{rf},k}\Delta t_{(n+1)} + \varphi_{\mathrm{rf},k}) = E_{d,(n+2)} - E_{d,(n+1)}$$
  - If the acceleration rate is not constant, i.e. $E_{d,(n+1)} - E_{d,(n)} \neq E_{d,(n+2)} - E_{d,(n+1)}$, then $\Delta t_{(n)} \neq \Delta t_{(n+1)}$
- This means that the particle synchronous from $n \rightarrow n+1$ is not anymore synchronous from $n+1 \rightarrow n+2$
  - A synchronous particle is **not a trackable particle**, it is an **abstract particle**

Synchrotron frequency

Each trajectory in phase space has a different synchrotron frequency $\omega_s(\Delta t, \Delta E)$

- The central synchrotron frequency ω_{s0} can be determined from the EOMs
  - In the approximation of small-amplitude oscillations and without intensity effects
  - For simplicity, consider a single RF system and a zeroth order slippage and define $\sin\varphi_s \equiv \dfrac{\dot{E}_d}{qV}$

$$\Delta\dot{E} = \frac{qV}{T_{\mathrm{rev}}}\left(\sin(\omega_{\mathrm{rf}}\Delta t + \varphi_{\mathrm{rf}}) - \sin\varphi_s\right) \text{ and } \Delta\dot{t} = \frac{\eta_0}{\beta_d^2 E_d}\Delta E$$

- Expanding the phase and the sine function around φ_s results in

$$\omega_{\mathrm{rf}}\Delta t + \varphi_{\mathrm{rf}} = \varphi_s + (\omega_{\mathrm{rf}}\Delta t + \varphi_{\mathrm{rf}} - \varphi_s) \equiv \varphi_s + \varepsilon,\ \ \varepsilon \ll 1$$

$$\sin(\omega_{\mathrm{rf}}\Delta t + \varphi_{\mathrm{rf}}) = \sin\varphi_s\cos\varepsilon + \cos\varphi_s\sin\varepsilon \approx \sin\varphi_s(1 + \mathcal{O}(\varepsilon^2)) + \cos\varphi_s(\varepsilon + \mathcal{O}(\varepsilon^3))$$

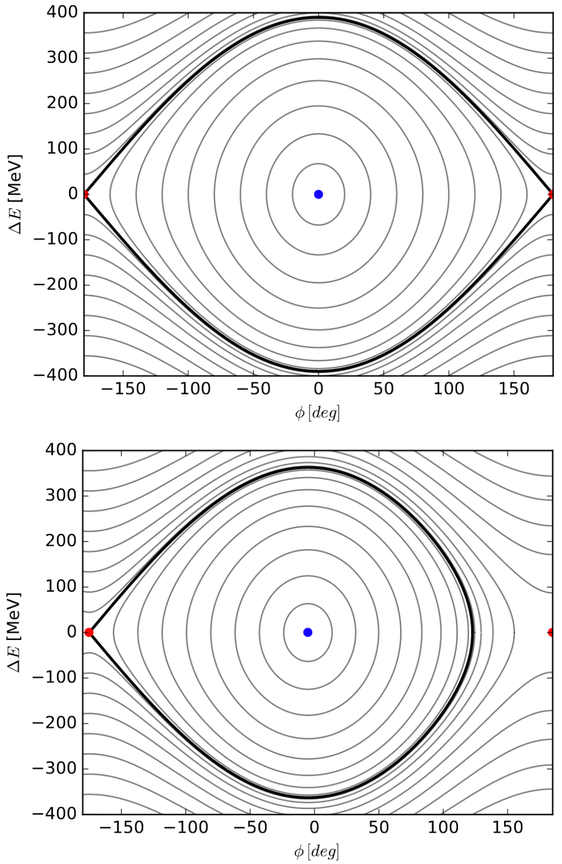
- Taking a second derivative of the drift (kick) equation, and combining it with the kick (drift) equation:

$$\Delta\ddot{E} = \frac{qV\cos\varphi_s\,\omega_{\mathrm{rf}}}{T_{\mathrm{rev}}}\Delta\dot{t} = \frac{qV\eta_0\cos\varphi_s\,\omega_{\mathrm{rf}}\omega_{\mathrm{rev}}}{2\pi\beta_d^2 E_d}\Delta E \quad \text{OR} \quad \Delta\ddot{t} - \Delta\ddot{t}_s = \frac{qV\eta_0\cos\varphi_s\,\omega_{\mathrm{rf}}\omega_{\mathrm{rev}}}{2\pi\beta_d^2 E_d}(\Delta t - \Delta t_s)$$

- Oscillating motion if $q\eta_0\cos\varphi_s < 0$. For $\omega_{\mathrm{rf}} = h\omega_0$ the synchrotron frequency reads:

$$\omega_{s,0} = \sqrt{\frac{-hqV\eta_0\cos\varphi_s}{2\pi\beta_d^2 E_d}}\omega_{\mathrm{rev}}$$

N.B. phase jump at transition crossing!



Trajectories in phase space from [7]

Top: stationary bucket

Bottom: accelerating bucket

[7] J. Esteban Müller: 'Longitudinal intensity effects in the CERN Large Hadron Collider ', CERN-THESIS-2016-066, 2016.

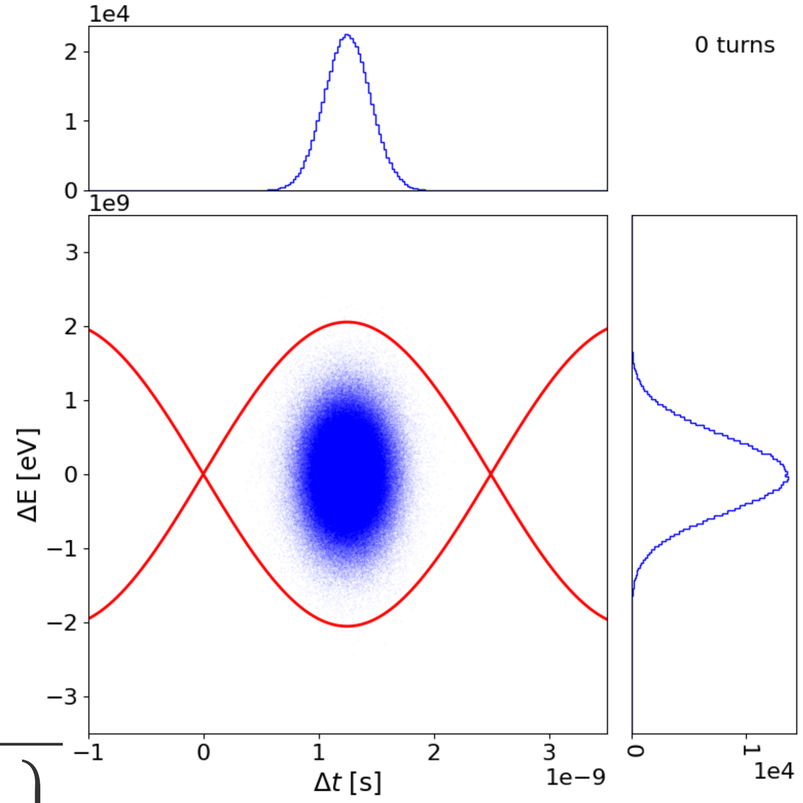
Separatrix and phase space

Separatrix

- The last bound trajectory
- In general, calculated from the full potential well
  - RF voltages and induced voltage
- To construct the separatrix, the unstable fixed point (UFP) needs to be determined $(\Delta t_{\mathrm{ufp}}, \Delta E = 0)$
- Calculated numerically by looking for the smallest (largest) zero crossing of the total voltage waveform above (below) transition
- The separatrix is the equipotential curve that goes through the UFP

  $H(\Delta t_{\mathrm{ufp}}, \Delta E = 0) = H(\Delta t_{\mathrm{sep}}, \Delta E_{\mathrm{sep}})$
- Without intensity effects:

$$\Delta E_{\mathrm{sep}} = \pm \sqrt{\frac{2\beta_d^2 E_d}{\eta_0} \left\{ \sum_{k=0}^{n_{\mathrm{rf}}-1} \frac{qV_k}{T_0 \omega_{\mathrm{rf}}^k} \left[\cos(\omega_{\mathrm{rf}}^k \Delta t_{\mathrm{ufp}} + \varphi_{\mathrm{rf}}^k) - \cos(\omega_{\mathrm{rf}}^k \Delta t_{\mathrm{sep}} + \varphi_{\mathrm{rf}}^k) \right] + \dot{E}_d (\Delta t_{\mathrm{ufp}} - \Delta t_{\mathrm{sep}}) \right\}}$$



RF bucket within the separatrix with phase space of single bunch and its profiles in time and energy

Symplectic condition (1)

Consider a generic coordinate mapping

- From $\mathbf{x} \equiv (\mathbf{q}, \mathbf{p}) = (q_1, \ldots, q_n, p_1, \ldots, p_n)$ to $\mathbf{y} \equiv (\mathbf{Q}, \mathbf{P}) = (Q_1, \ldots, Q_n, P_1, \ldots, P_n)$
  - Defining the 2n by 2n matrix

    $$\mathcal{S} \equiv \begin{pmatrix} \mathbf{0} & \mathbf{1} \\ -\mathbf{1} & \mathbf{0} \end{pmatrix}$$

  - We can rewrite Hamilton's equations as:

    $$\dot{\mathbf{q}} = \frac{\partial H}{\partial \mathbf{p}} \text{ and } \dot{\mathbf{p}} = -\frac{\partial H}{\partial \mathbf{q}} \rightarrow \dot{\mathbf{x}} = S\frac{\partial H}{\partial \mathbf{x}}$$

  - The updated coordinates also fulfil Hamilton's equation (condition for a canonical transformation):

    $$\dot{\mathbf{y}} = S\frac{\partial H}{\partial \mathbf{y}}$$

  - Applying the chain rule on the updated coordinates that can be expressed as $\mathbf{y} = \mathbf{y}(\mathbf{x})$

    $$\dot{y}_i = \frac{dy_i(x)}{dt} = \frac{\partial y_i}{\partial x_k}\frac{dx_k}{dt} = \frac{\partial y_i}{\partial x_k}\mathcal{S}_{kl}\frac{\partial H}{\partial x_l} = \underbrace{\frac{\partial y_i}{\partial x_k}}_{\equiv \mathcal{M}} \mathcal{S}_{kl} \underbrace{\frac{\partial y_l}{\partial x_m}}_{= \mathcal{M}^T} \frac{\partial H}{\partial y_m}$$

  - Mapping or Jacobi matrix:

Symplectic condition (2)

- Written in the matrix form and combining the condition for y to fulfil Hamilton's equations:

$$\dot{\mathbf{y}} = \frac{d\mathbf{y}(\mathbf{x})}{dt} = \mathcal{M}\frac{d\mathbf{x}}{dt} = \mathcal{M}\mathcal{S}\frac{\partial H}{\partial \mathbf{x}} = \mathcal{M}\mathcal{S}\mathcal{M}^T\frac{\partial H}{\partial \mathbf{y}} \quad \overset{!}{=} \quad S\frac{\partial H}{\partial \mathbf{y}}$$

- From which the **symplectic condition** can be extracted:

$$\mathcal{M}\mathcal{S}\mathcal{M}^T = \mathcal{S}$$

- It is equivalent to conserving the symplectic 2-form

$$d\mathbf{p} \wedge d\mathbf{q} \equiv \sum_i dp_i \wedge dq_i = d\mathbf{P} \wedge d\mathbf{Q}$$

- For 1D systems, **symplecticity** is equivalent to being **area preserving**
  - Some integrators, like Euler or Runge-Kutta are not area preserving
  - ➡ This can lead to fictitious (numerical) damping or excitation [8]

[8] A. W. Chao *et al.*: 'Handbook of Accelerator Physics and Engineering', 2nd Ed., World Scientific, 2013.

Symplecticity of EOMs

Consider now the longitudinal mapping governed by the kick and drift equations, in the absence of collective effects ($E_{\text{other}} = 0$)

- The coordinate mapping is: $\mathbf{x} \equiv (\Delta t_{(n)}, \Delta E_{(n)}) \rightarrow \mathbf{y} \equiv (\Delta t_{(n+1)}, \Delta E_{(n+1)})$
- We can rewrite the EOMs as

$$\Delta t_{(n+1)} = \Delta t_{(n)} + f\left(\Delta E_{(n+1)}(\Delta t_{(n)}, \Delta E_{(n)})\right)$$

$$\Delta E_{(n+1)} = \Delta E_{(n)} + g\left(\Delta t_{(n)}\right)$$

- Using the chain rule, the mapping or Jacobian matrix becomes

$$\mathscr{M} = \begin{pmatrix} 1 + f'g' & f' \\ g' & 1 \end{pmatrix}$$

- This matrix fulfils the symplecticity condition, and is therefore area preserving

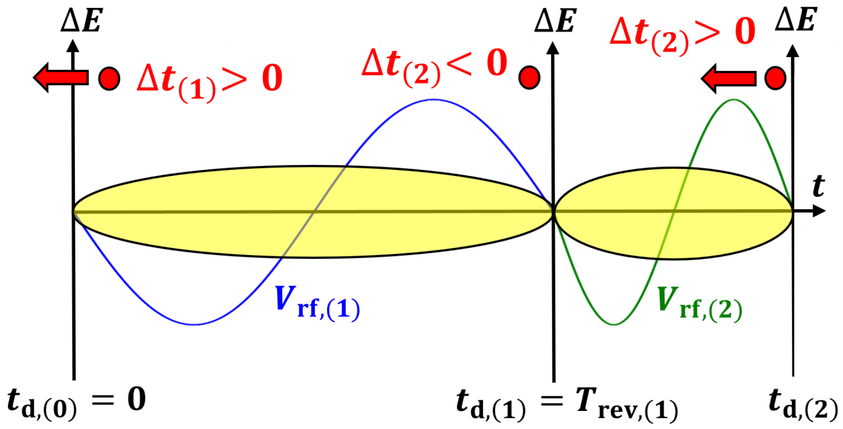
$$\mathscr{M}^{\mathrm{T}}\mathcal{S}\mathscr{M} = \begin{pmatrix} 1 + f'g' & g' \\ f' & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} 1 + f'g' & f' \\ g' & 1 \end{pmatrix} = \begin{pmatrix} -g' & 1 + f'g' \\ -1 & f' \end{pmatrix}\begin{pmatrix} 1 + f'g' & f' \\ g' & 1 \end{pmatrix} =$$

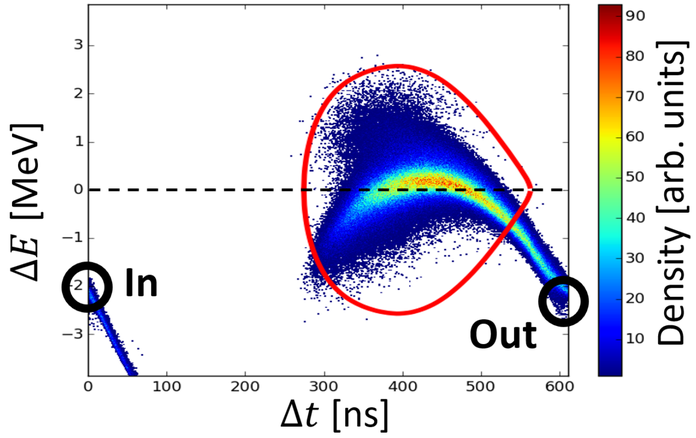
$$= \begin{pmatrix} (-g' + g')(1 + f'g') & -f'g' + 1 + f'g' \\ -(1 + f'g') + f'g' & -f' + f' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = \mathcal{S}$$

Periodicity

According to the turn-by-turn discretisation, the time coordinate of any particle should be $(0, T_{\mathrm{rev},(n)})$

- In some cases, however, the particles might cross these time boundaries
  - E.g. small h machines, full machine, debunching beam, etc.
- Introduce periodic boundary conditions to the coordinate frame:
  - Particles that have $\Delta t_{(n)} < 0$ have to be tracked twice
  - Particles that have $\Delta t_{(n)} > T_{\mathrm{rev},(n)}$ have to be put on hold for a turn



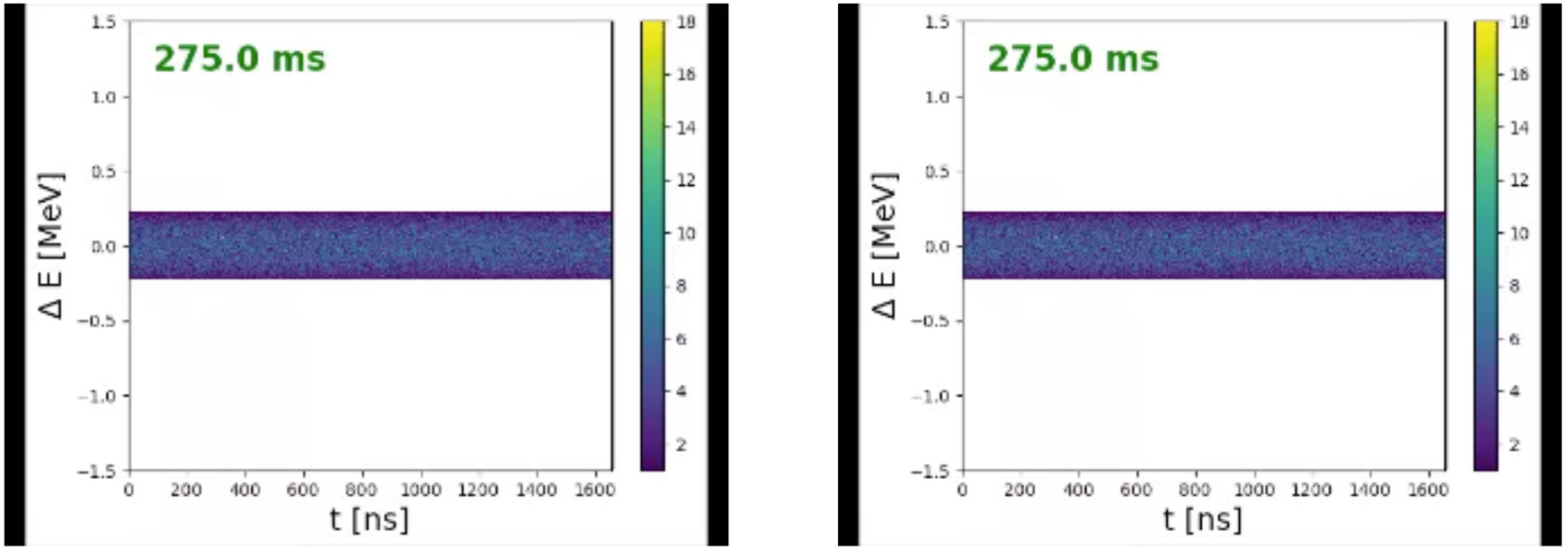



```python
if self.periodicity:

    # Distinguish the particles inside the frame from the particles on
    # the right-hand side of the frame.
    self.indices_right_outside = \
        bm.where(self.beam.dt > self.rf_params.t_rev[turn + 1])[0]
    self.indices_inside_frame = \
        bm.where(self.beam.dt < self.rf_params.t_rev[turn + 1])[0]

    if len(self.indices_right_outside) > 0:
        # Change reference of all the particles on the right of the
        # current frame; these particles skip one kick and drift
        self.beam.dt[self.indices_right_outside] -= \
            self.rf_params.t_rev[turn + 1]
        # Synchronize the bunch with the particles that are on the
        # RHS of the current frame applying kick and drift to the
        # bunch
        # After that all the particles are in the new updated frame
        self.insiders_dt = bm.ascontiguousarray(
            self.beam.dt[self.indices_inside_frame])
        self.insiders_dE = bm.ascontiguousarray(
            self.beam.dE[self.indices_inside_frame])
        self.kick(self.insiders_dt, self.insiders_dE, turn)
        self.drift(self.insiders_dt, self.insiders_dE, turn + 1)
        self.beam.dt[self.indices_inside_frame] = self.insiders_dt
        self.beam.dE[self.indices_inside_frame] = self.insiders_dE
        # Check all the particles on the left of the just updated
        # frame and apply a second kick and drift to them with the
        # previous wave after having changed reference.
        self.indices_left_outside = bm.where(self.beam.dt < 0)[0]

    else:
        self.kick(self.beam.dt, self.beam.dE, turn)
        self.drift(self.beam.dt, self.beam.dE, turn + 1)
        # Check all the particles on the left of the just updated
        # frame and apply a second kick and drift to them with the
        # previous wave after having changed reference.
        self.indices_left_outside = bm.where(self.beam.dt < 0)[0]

    if len(self.indices_left_outside) > 0:
        left_outsiders_dt = bm.ascontiguousarray(
            self.beam.dt[self.indices_left_outside])
        left_outsiders_dE = bm.ascontiguousarray(
            self.beam.dE[self.indices_left_outside])
        left_outsiders_dt += self.rf_params.t_rev[turn + 1]
        self.kick(left_outsiders_dt, left_outsiders_dE, turn)
        self.drift(left_outsiders_dt, left_outsiders_dE, turn + 1)
        self.beam.dt[self.indices_left_outside] = left_outsiders_dt
        self.beam.dE[self.indices_left_outside] = left_outsiders_dE
```

An example of periodicity

And this is how it looks like in a simulation for the CERN Proton Synchrotron Booster (PSB)

- Beam injection process simulated



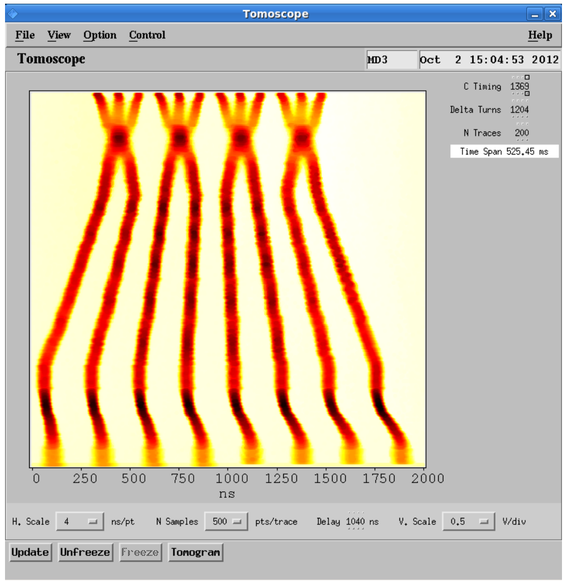
PSB injection on acceleration ramp with RF locked to the B-field (left) and fixed RF frequency (right)

Courtesy of D. Quartullo

RF gymnastics

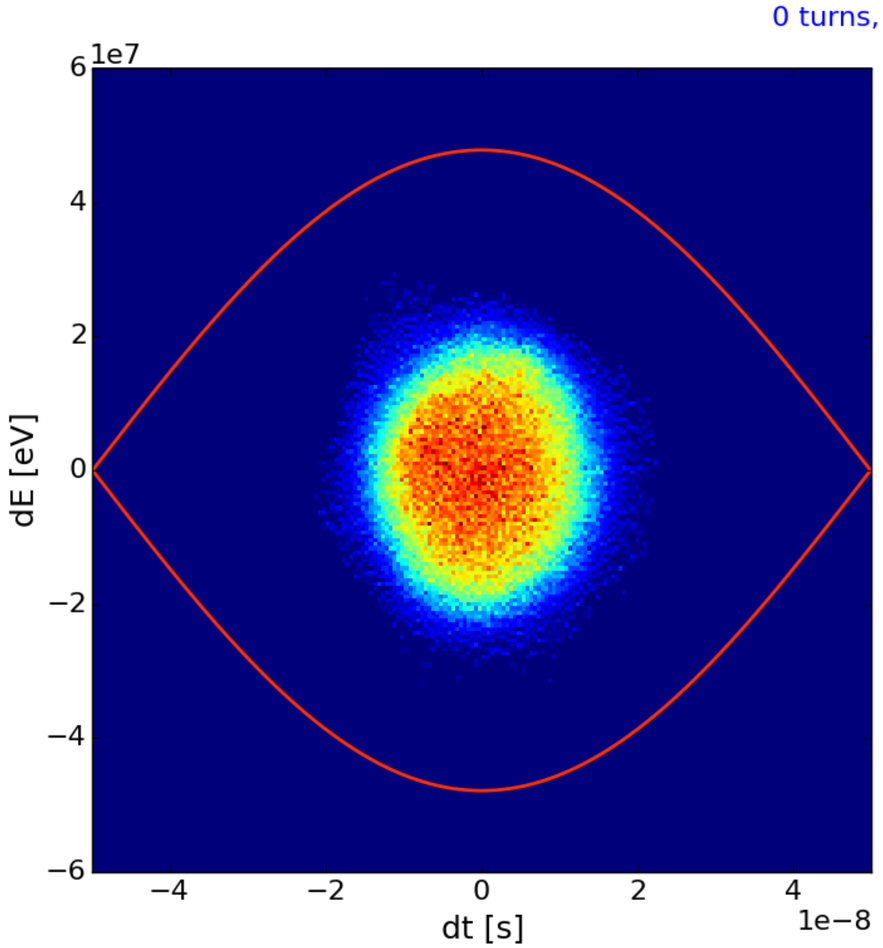
Multi-harmonic systems

- RF gymnastics defined by the RF voltage programme of several harmonics
  - Splitting or merging: adiabatically add a higher-harmonic RF voltage
  - Bunch rotation: non-adiabatically increase the RF voltage and extract or recapture bunch after $\frac{1}{4}T_s$
  - Bunch compression: change of harmonic of a given RF system



Bunch Compression, Merging and Splitting (BCMS) scheme in the CERN Proton Synchrotron (PS)

Courtesy of H. Damerau

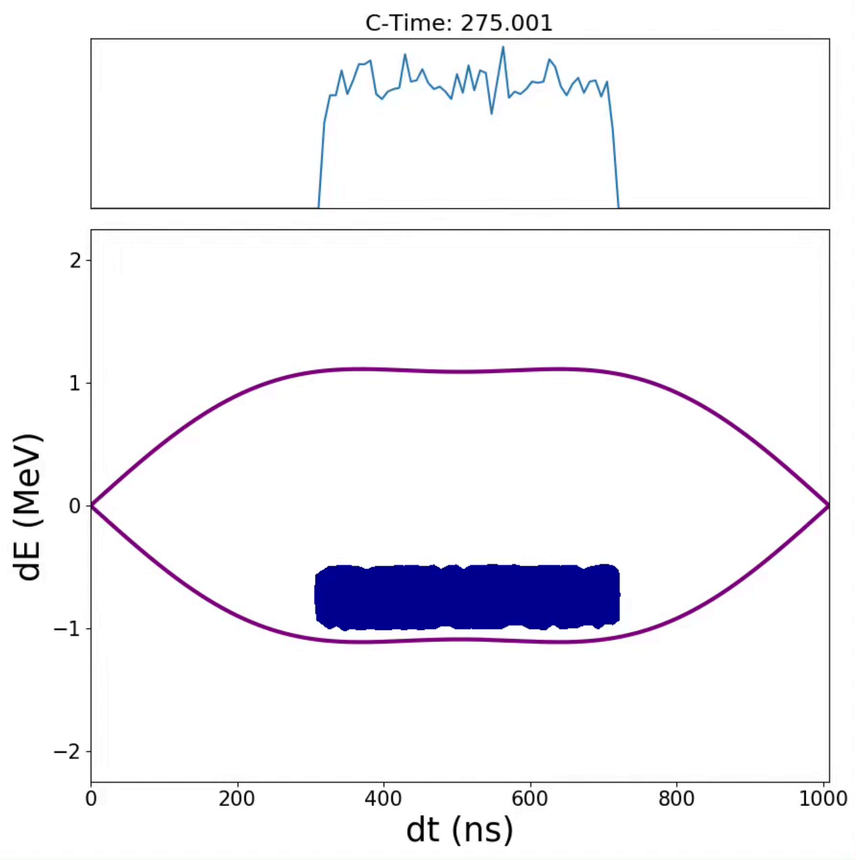


Double splitting and bunch rotation in the PS, then capture in the Super Proton Synchrotron (SPS)

Longitudinal painting

Longitudinal painting

- Is used to “fill” a part of the longitudinal phase space
  - Emittance is filamenting for longitudinal stability in the machine where injected to
- Multi-turn injection
  - Example PSB: injecting small bunches at different energy offsets over several turns
  - Injected bunch has a small emittance
  - Final bunch fills the entire bucket
- From computational point of view
  - Requires expanding the object that holds the beam coordinates



Multi-turn injection and longitudinal painting

Courtesy of S. Albright

Content

Beam tracking basics

- Tracking of beams
- Discretisation

Longitudinal tracking

- Reference frame
- Equations of motion
- Periodicity

Effects on particle energy

- Collective effects and impedance
- Multi-turn wake
- Synchrotron radiation

RF modelling

- Tune and beam loading
- Global and local control loops

Particle distribution

- Observables and parametrisation
- Matching distributions

6D effects

- Some examples

Code design, optimisation and benchmarks

- Good practises

Induced voltage

Voltage induced by the beam particles in an electromagnetic object

- In time domain, convolution of beam profile and wake function:

$$V_{\mathrm{ind}}(\Delta t) = -\,q\,N_p \int_{-\infty}^{+\infty} \lambda(\tau)\,W(\Delta t - \tau)\,d\tau$$

  - The minus sign ensures an energy loss for a positive wake potential

- The energy kick due to the induced voltage enters the E_{other} term

$$E_{\mathrm{ind}}(\Delta t) = q\,V_{\mathrm{ind}}(\Delta t)$$

  - Has to performed before the drift if the the energy is updated first

- In frequency domain, the impedance of the electromagnetic object is

$$Z(\omega) = \int_{-\infty}^{+\infty} W(t)\,e^{-j\,\omega\,t}\,dt$$

- And V_{ind} becomes the product of the impedance and the beam spectrum

$$V_{\mathrm{ind}}(\Delta t) = \frac{1}{2\,\pi}\int_{-\infty}^{+\infty} V_{\mathrm{ind}}(\omega)\,e^{j\,\omega\,\Delta t}\,d\Delta t = -\frac{q\,N_p}{2\,\pi}\int_{-\infty}^{+\infty} Z(\omega)\,\Lambda(\omega)\,e^{j\,\omega\,\Delta t}\,d\Delta t$$

$\int_{-\infty}^{+\infty} \lambda(t)\,dt = 1$ beam profile, normalised to 1

$\Lambda(\omega) = \int_{-\infty}^{+\infty} \lambda(t)\,e^{-j\,\omega\,t}\,dt$ beam spectrum

N_p number of real particles in the beam

$W(t)$ wake function

Discretising the induced voltage

The beam profile is discretised on an equidistant grid $\lambda[m] = \lambda(t_m)$

- In frequency domain, knowing the machine impedance:

  $$V_{\mathrm{ind}}[k] = -\,q\,N_p\,\mathrm{IFFT}\left(Z[i]\;\Lambda[i]\right) = -\,q\,N_p\,\mathrm{IFFT}\left(Z[i]\;\mathrm{FFT}(\lambda[m])\right)$$

  - Multiplication is faster than convolution
  - However, need to calculate beam spectrum via FT and inverse FT for induced voltage

- In time domain, could use a discrete convolution

  $$V_{\mathrm{ind}}[k] = -\,q\,N_p\;\sum_m \lambda[m]\,W[k-m]$$

  - Unless we use small arrays, this is **time consuming** as it has a computational complexity of $\mathcal{O}(K \times M)$

- In time domain, a faster way is to use again FFTs, using the circular convolution theorem

  $$V_{\mathrm{ind}}[k] = -\,q\,N_p\,\mathrm{IFFT}\left\{\mathrm{FFT}\left(W[n]\right)\;\mathrm{FFT}\left(\lambda[n]\right)\right\}$$

  - Fast Fourier transforms reduce the computational complexity to $\mathcal{O}(N\log(N))$
  - Must **carefully zero-pad** to result in a linear convolution: $N \geq K + M - 1$

Tip: use one of the heavily optimised FFT libraries (e.g. [9]); it is hard to do better than that!

[9] FFTW library, the Fastest Fourier Transform in the West, https://www.fftw.org/

Impedance sources (1)

Impedance table

- Import tables based on electromagnetic modelling of accelerator elements (e.g. CST [10])

Resonator

$$Z(\omega) = \frac{R_s}{1 + jQ\left(\frac{\omega}{\omega_r} - \frac{\omega_r}{\omega}\right)}$$

$$W(t > 0) = 2\alpha R_s e^{-\alpha t}\left(\cos\tilde{\omega}t - \frac{\alpha}{\tilde{\omega}}\sin\tilde{\omega}t\right)$$

$$W(0) = \alpha R_s, \text{ with } \alpha = \frac{\omega_r}{2Q} \text{ and } \tilde{\omega} = \sqrt{\omega_r^2 - \alpha^2}$$

ω_r	resonant frequency
Q	cavity Q
R_s	shunt impedance

Travelling wave cavity [11]

$Z = Z_+ + Z_-$ and $\tilde{\omega}_\pm \equiv \omega \pm \omega_r$ where

$$Z_\pm(f) \equiv R_s\left[\left(\frac{\sin\frac{\tilde{\omega}_\pm\tau}{2}}{\frac{\tilde{\omega}_\pm\tau}{2}}\right)^2 \mp 2i\frac{\tilde{\omega}_\pm\tau - \sin\tilde{\omega}_\pm\tau}{\left(\tilde{\omega}_\pm\tau\right)^2}\right]$$

$$W(0 < t < \tau) = \frac{4R_s}{\tau}\left(1 - \frac{t}{\tau}\right)\cos\omega_r t$$

$$W(0) = \frac{2R_s}{\tau}$$

τ	filling time
ω_r	resonant frequency
R_s	shunt impedance

[10] CST Studio Suite, Dassault Systemes, https://www.3ds.com/products-services/simulia/products/cst-studio-suite/

[11] G. Dôme: 'The SPS acceleration system travelling wave drift-tube structure for the CERN SPS', CERN Report , CERN-SPS-ARF-77-11, 1977.

Impedance sources (2)

Resistive wall impedance

$$Z(f) = \frac{Z_0 c L}{\pi} \frac{1}{\left[1 - i\,\mathrm{sgn} f\right] 2bc\sqrt{\frac{\sigma_c Z_0 c}{4\pi|f|} + i2\pi b^2 f}}$$

b	beam pipe radius
L	beam pipe length
Z_0	vacuum impedance

Constant $\Im(Z/n)$ e.g. for loss of Landau damping or space charge modelling

$$\frac{Z}{n} = \frac{Z}{f/f_{\mathrm{rev}}} = \text{const.}$$

- Simplifies the computation of the induced voltage, replacing FFTs with the derivative of the line density

$$V_{\mathrm{ind}}[k] = -\frac{q\,T_{\mathrm{rev}}}{2\,\pi\,T_s}\frac{Z}{n}\frac{d\lambda[k]}{dn}$$

  - Caveat: line density derivative can add numerical noise!

Beam (in)stability

Induced voltage is usually added to look for stable/unstable solutions

- What to resolve, what are the observables, and to how many bunches can we restrict?
- Hard to give a generic recipe, but for sure you must resolve what you want to observe!
  - E.g. microwave instabilities: must have a fine grid over one bunch
  - E.g. coupled-bunch instabilities: must resolve long-range wakes

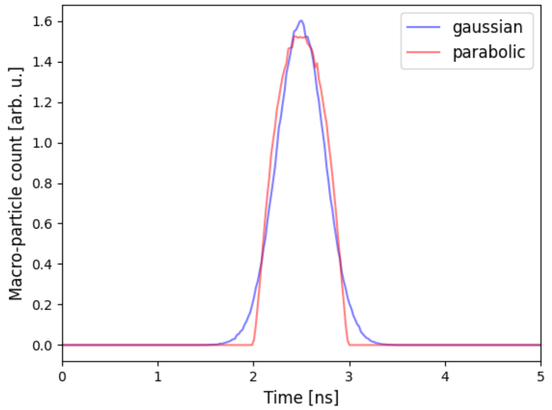
Designing your simulation: think of impedance vs beam spectrum

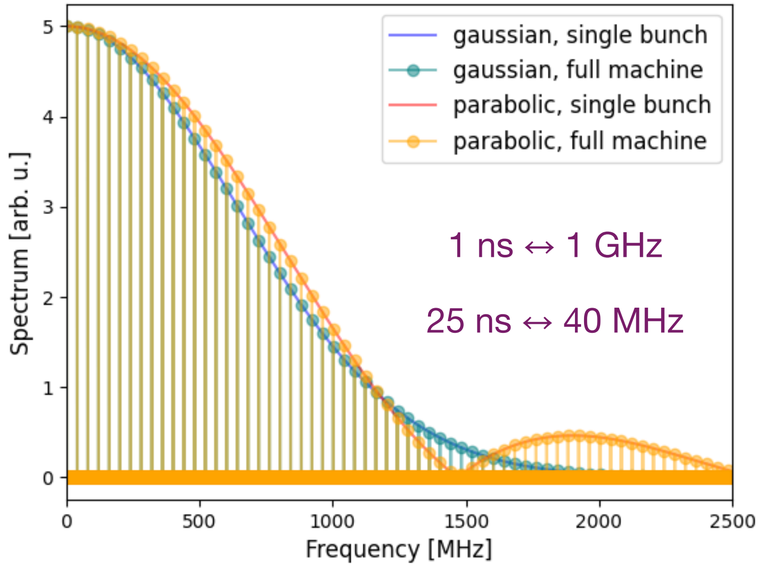
- What is the highest frequency impedance that you must resolve in view of your beam spectrum (bunch length)?
- What is the resolution of your narrowest impedance that you want to resolve? Adjust your beam profile length and impedance resolution

$$\Delta t_{\mathrm{bin}} = 1/f_{\mathrm{max}} \text{ and } t_{\mathrm{max}} = N\Delta t_{\mathrm{bin}} = 1/\Delta f$$

Observables

- Dipole (centre-of mass) and quadrupole (bunch length) oscillations





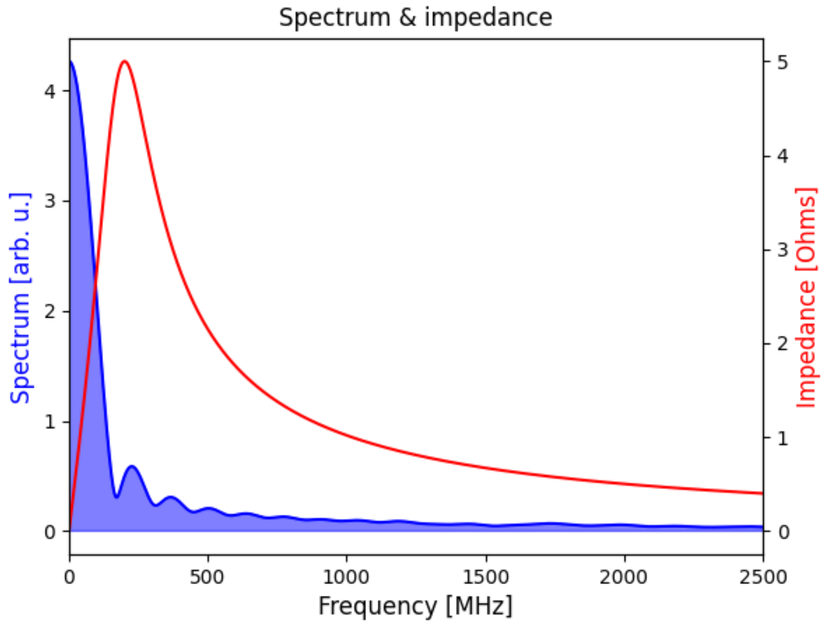
Top: bunch profiles of 1 ns long gaussian and parabolic bunches

Bottom: corresponding spectra for single bunches and full machine of 25 ns spacing

Narrow-band and broad-band regime

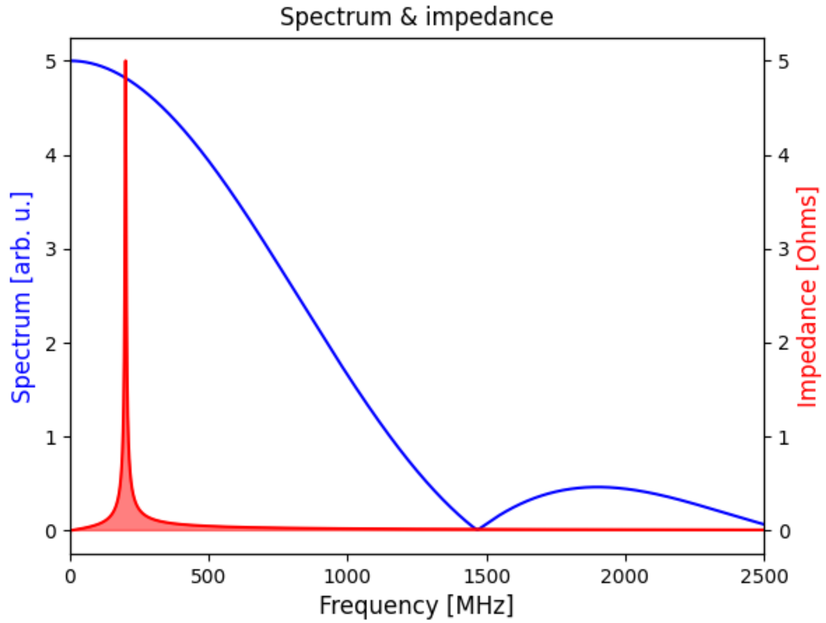
Machine impedance can also be described as a sum of resonators

- Two edge cases: narrow-band and broad-band regime
- Impedances that most effectively drive instabilities have $\omega_r/(2Q) \sim 1/\tau$



Broad-band or short-bunch regime:

the bunch spectrum samples the integral impedance, that is R_{sh}/Q



Narrow-band or long-bunch regime:

the bunch spectrum samples the impedance at a given frequency, that is R_{sh}

Resolving beam and impedance

Using the right binning

- Long wake or narrow impedance
  - In time domain, t_{max} has to be large, in frequency domain Δf_{bin} must be small
  - Could happen, that multiple turns have to be resolved
- Short wake or broad impedance
  - In time domain, Δt_{bin} has to be small, in frequency domain f_{max} has to be large

Using the right cutoff frequency

- Some broad-band impedance models may produce unphysical impedance at high frequencies
  - This can result in unphysical beam instabilities
  - Make sure to use the physical cutoff frequency for your impedance models

Using the right amount of simulation particles and binning

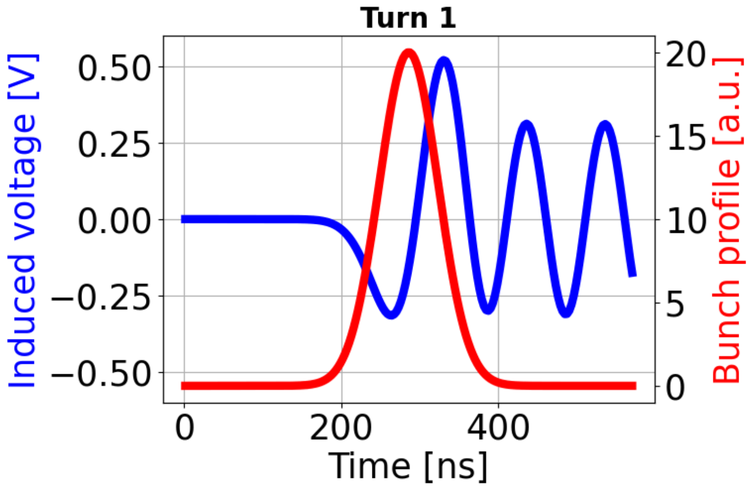
- E.g. a fine binning in time domain requires more macro particles to be simulated

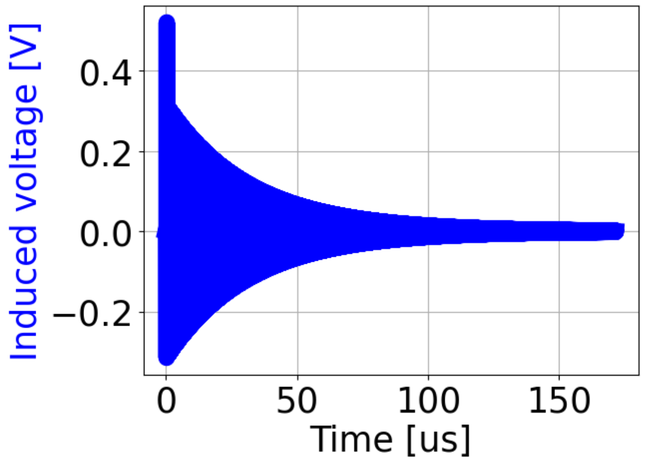
Tip: always perform convergence studies on the number of particles and number of beam profile bins

Multi-turn wake

Impedances might couple particles over several machine turns

- Need to keep bunch profiles over several past turns in the memory

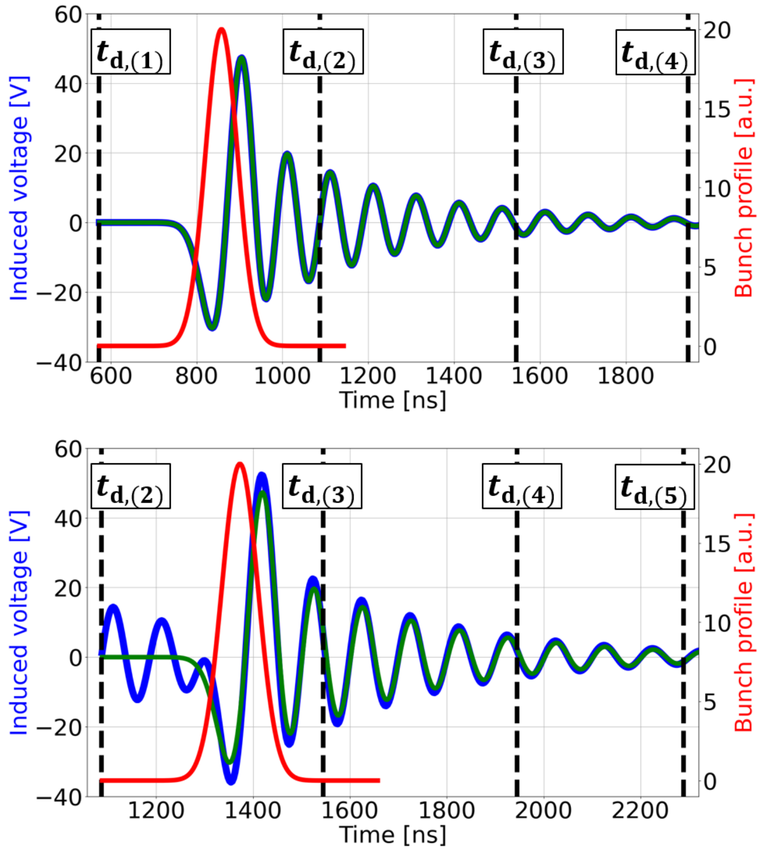




Example: multi-turn wake calculations in the PSB; 300 turns of wake is tracked

PSB flattop 1.4 GeV, h=1, single bunch with resonator of Rs = 5000, fr = 10 MHz, Q = 1000

Courtesy of D. Quartullo



Top: first turn

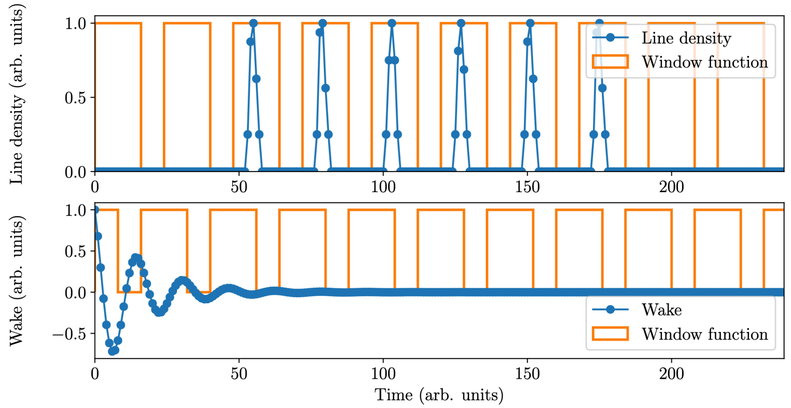
Bottom: second turn

Courtesy of A. Lasheen

Advanced schemes

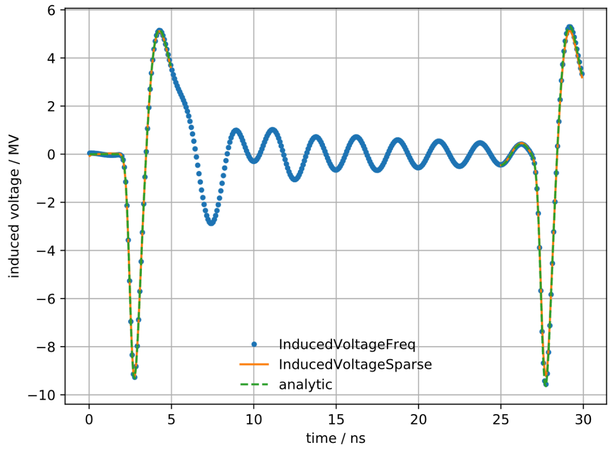
Compressed convolution [12,13]

- When there is proportionally large bunch spacing compared to the bunch length
- Only keep the relevant data points and use a window function for the zeros



Sparse slicing or non-uniform grid

- Via non-uniform Fourier transform
  - See equations in [14]
- Or more exotic ones like chirp and z-transform [15]



[12] J. Komppula, K. Li, and N. Mounet at the PyHEADTAIL meeting https://indico.cern.ch/event/735184/

[13] I. Karpov at the BLonD meeting https://indico.cern.ch/event/996624/#5-follow-up-on-compressed-conv

[14] M. Schwarz at the BLonD meeting https://indico.cern.ch/event/996624/#4-non-homogeneously-sampled-bu

[15] M. Vadai at the BLonD meeting https://indico.cern.ch/event/996624/#2-non-uniform-fft

Top: Window function for compressed convolution

Bottom: induced voltage with sparse slicing

Potential-well distortion

Perturbation of potential well due to induced voltage

- In single RF and short-bunch approximation

$$\frac{d^2\Delta\varphi}{dt^2} + \omega_{s0}^2\Delta\varphi = \frac{\omega_{s0}^2}{V_{\mathrm{rf}}\cos\varphi_s}V_{\mathrm{ind}}(\Delta\varphi)$$

- On the other hand, for long-range wake fields and constant beam spectrum [16]

$$V_{\mathrm{ind}}(\Delta\varphi) = -\frac{N_p q\omega_{\mathrm{rev}}}{2\pi}\sum_{p=-\infty}^{\infty}\left(Z(p\omega_{\mathrm{rev}})\Lambda(p\omega_{\mathrm{rev}})e^{jp\omega_{\mathrm{rev}}t}\right)$$

- Expanding the exponential and the phase around φ_s results in a shift of the synchronous phase of

$$\Delta\varphi_s = -\frac{N_p q\omega_{\mathrm{rev}}}{2\pi V_{\mathrm{rf}}\cos\varphi_s}\sum_{p=-\infty}^{\infty}\left(\Re(Z(p\omega_{\mathrm{rev}}))\,|\,\Lambda(p\omega_{\mathrm{rev}})\,|^2\right)$$

- And a shift in synchrotron frequency of

$$\frac{\Delta\omega_s}{\omega_{s0}} \approx \frac{N_p q\omega_{\mathrm{rev}}}{4\pi V_{\mathrm{rf}}\cos\varphi_s}\Im(Z/n)\sum_{p=-\infty}^{\infty}\left(p^2\Lambda(p\omega_{\mathrm{rev}})\right)$$

- In addition, a bunch lengthening is observed for $\gamma > \gamma_T$ and $\Im(Z/n) > 0$ w.r.t. the zero-intensity bunch length τ_0

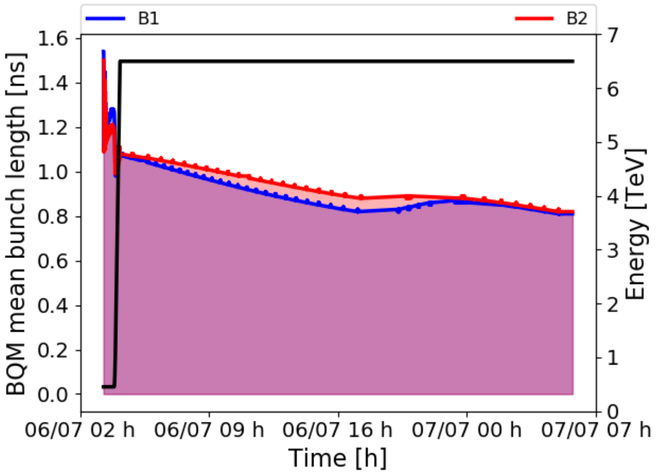
$$\left(\frac{\tau}{\tau_0}\right)^2 = \frac{\omega_{s0}}{\omega_s}\sqrt{\frac{\cos\varphi_s}{\cos\varphi_s + \Delta\varphi_s}}$$

[16] J. L. Laclare: ‘Bunched beam coherent instabilities’, Proc. CAS: Accelerator Physics, pp. 264–326,1985.

Loss of Landau damping

Landau damping in a nutshell

- Originally derived for plasmas by Landau [17]
- Principle: need a spread in synchrotron frequencies between different bunch particles to keep the bunch stable, for otherwise the bunch would oscillate in a coherent manner
- E.g. for a constant energy, voltage, emittance, if the bunch intensity increases, at some point a coherent line separates from the continuous, incoherent bunch spectrum
  - The intensity at which this separation happens is the threshold of loss of Landau damping
  - Driven by $\mathfrak{F}(Z/n)$
- Can be single/multi-bunch effect



Slow emittance blow-up in LHC during physics due to loss of Landau damping

In particle tracking

- Loss of Landau damping can be tested by giving a small phase kick to a bunch/beam in steady state
- If bunch oscillations are undamped, we talk of loss of Landau damping
  - N.B. in practise it might be difficult to find a suitable threshold of oscillation amplitude to distinguish background or numerical oscillations from undamped oscillations

[17] L. Landau: 'On the vibration of the electronic plasma', JETP **16**, 1946.

Synchrotron radiation and quantum excitation

Synchrotron radiation: emission of EM radiation for high-energy particles on a bent trajectory

- Shrinks the bunch emittance
- Energy loss applied per turn or in a sub-cycled manner
  - For machines with large energy loss per turn, one has to avoid large discrete energy kicks leading to fake debunching
  - Can use sub-cycling with SR kick + drift, even where there is no RF cavity in the machine

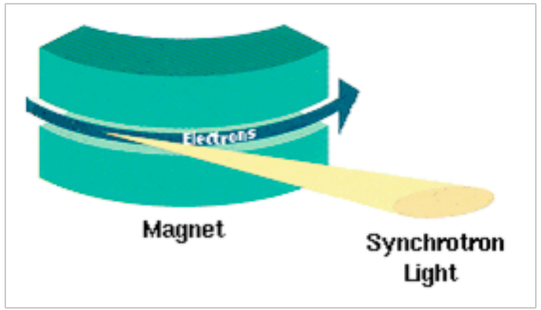
$$E_{\mathrm{other},(n)} = -U_0 - \frac{2}{\tau_z}\Delta E_{(n)}$$

$\Delta E_{(n)}$: Difference in energy loss for each particle

U_0: Average energy loss

  - Where the average energy loss is:

$$U_0 = \frac{4\pi}{3}\frac{r_{\mathrm{cl}}}{m_p^3 c^6}\frac{1}{\rho}E_{d,(n)}^4\frac{R}{C}$$



ρ	magnet bending radius
$\sigma_{\Delta E}$	equilibrium energy spread
τ_z	damping time [turns]
m_p	particle mass
RANDN	normal (Gaussian) random number in (0,1)
$r_{\mathrm{cl}} = \frac{1}{4\pi\varepsilon_0}\frac{q^2}{m_p c^2}$	classical particle radius

Quantum excitation: a collective effect for short bunches [18]

- Counteracting emittance blow-up due to statistical emission of photons

$$E_{\mathrm{other},(n)} = 2\frac{\sigma_{\Delta E}}{\sqrt{\tau_z}}E_{d,(n)}\mathrm{RANDN}$$

[18] F. E. Müller: 'Modification of the simulation code BLonD for lepton rings', https://zenodo.org/record/7675649, (2017)

Coherent synchrotron radiation

Coherent synchrotron radiation can also be modelled as an impedance

- The free-space model gives [19]

$$\frac{\Re Z}{Z_0} = \frac{\sqrt{3}\gamma}{4}\frac{f}{f_{\mathrm{crit}}}\int_{f/f_{\mathrm{crit}}}^{\infty} K_{5/3}(y)\,dy$$

$$\frac{\Im Z}{Z_0} = \frac{-\gamma f}{f_{\mathrm{crit}}}\left[\int_0^1 e^{-\frac{f}{f_{\mathrm{crit}}}y}\left(\frac{-2\tilde{y}}{4y\tilde{y}} + \frac{\left(i\,y+\tilde{y}\right)^{5/3} + \left(i\,y+\tilde{y}\right)^{-5/3}}{4y\tilde{y}}\right)dy - \int_1^{\infty} e^{-\frac{f}{f_{\mathrm{crit}}}y}\left(\frac{8i\tilde{y}}{8yi\tilde{y}} - \frac{\left(y+i\tilde{y}\right)^{5/3} - \left(y+i\tilde{y}\right)^{-5/3}}{8yi\tilde{y}}\right)dy\right]$$

$$f_{\mathrm{crit}} \equiv \frac{3}{4\pi}\gamma^3\frac{c}{\rho}$$

$\tilde{y} \equiv \sqrt{1-y^2}$

ρ magnet bending radius

[19] J. Murphy, S. Krinsky, and R. Gluckstern, Part.Accel. **57**, 9 (1997).

Content

Beam tracking basics

- Tracking of beams
- Discretisation

Longitudinal tracking

- Reference frame
- Equations of motion
- Periodicity

Effects on particle energy

- Collective effects and impedance
- Multi-turn wake
- Synchrotron radiation

RF modelling

- Tune and beam loading
- Global and local control loops

Particle distribution

- Observables and parametrisation
- Matching distributions

6D effects

- Some examples

Code design, optimisation and benchmarks

- Good practises

Cavity-transmitter-beam interaction

Superconducting resonant cavities

- Can be modelled as a resonator

$$Z(f) = \frac{R_s}{1 + jQ\left(\frac{\omega}{\omega_r} - \frac{\omega_r}{\omega}\right)}$$

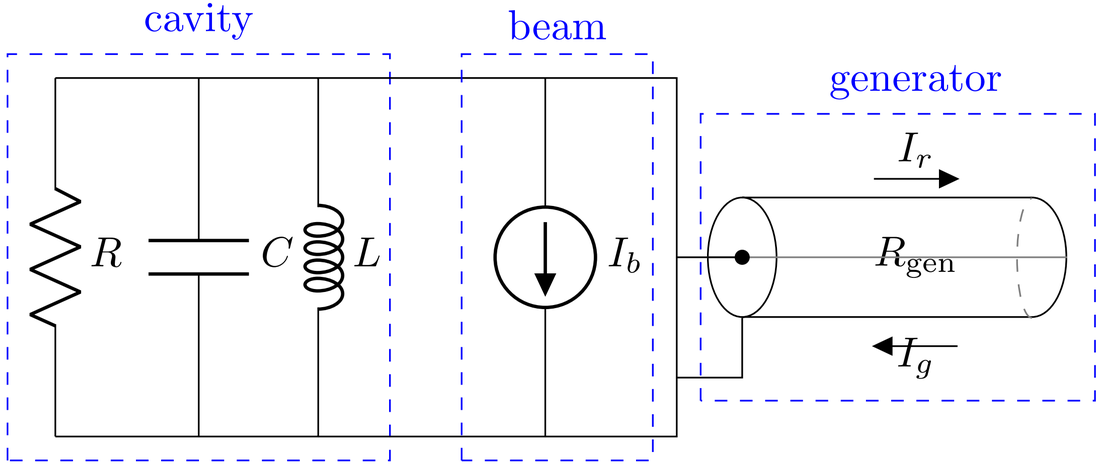
Cavity-transmitter-beam interaction can be described using a circuit model [20]

- Cavity: RLC circuit, beam: current source, generator: transmission line

$$I_{\mathrm{gen}}(t) = \frac{V_{\mathrm{ant}}(t)}{2R/Q}\left(\frac{1}{Q_L} - 2i\frac{\Delta\omega}{\omega}\right) + \frac{dV_{\mathrm{ant}}(t)}{dt}\frac{1}{\omega R/Q} + \frac{1}{2}I_{\mathrm{b,rf}}(t)$$

- Or, in time discrete form,

$$V_{\mathrm{ant}}^{(n)} = \frac{R}{Q}\omega T_{\mathrm{rev},(n)}\, I_{\mathrm{gen}}^{(n-1)} + \left(1 - \frac{\omega T_s}{2Q_L} + i\Delta\omega T_{\mathrm{rev},(n)}\right) V_{\mathrm{ant}}^{(n-1)} - \\ - \frac{1}{2}\frac{R}{Q}\omega T_{\mathrm{rev},(n)}\, I_{\mathrm{b,rf}}^{(n-1)}$$



[20] J. Tückmantel: 'Cavity-beam-transmitter interaction formula collection with derivation', CERN-ATS-Note-2011-002 TECH, 2011.

RF tune and detuning

The cavity tune is chosen usually to minimise the RF power

- In the absence of beam, the frequency is tuned to the centre of the resonance ω_r (max. V, min. P)

Half-detuning beam-loading compensation scheme [21]: optimum average power keeping $\vec{V}_{\text{ant}} = \text{const.}$

$$P_{\text{gen}} = \frac{1}{8} R/Q Q_L \left(\frac{V_{\text{ant}}}{R/Q} \frac{1}{Q_L} + \cancel{\Re(I_{b,\text{rf}})}^{\,0} \right)^2 + \frac{1}{8} R/Q Q_L \left(-2 \frac{V_{\text{ant}}}{R/Q} \frac{\Delta\omega}{\omega_r} + \Im(I_{b,\text{rf}}) \right)^2$$

$$P_{\text{gen}} = \frac{1}{8} \frac{V_{\text{ant}}^2}{R/Q Q_L} + \frac{1}{32} R/Q Q_L I_{b,\text{rf}}^2 = \frac{1}{8} V_{\text{ant}} I_{b,\text{rf}} \quad \text{with the optimum detuning of} \quad \Delta\omega_{\text{HD}} = \frac{1}{4} R/Q \frac{I_{b,\text{rf}}}{V_{\text{ant}}} \omega_r$$

Full-detuning beam-loading compensation scheme [22]: only keep $|\vec{V}_{\text{ant}}| = \text{const.}$

$$P_{\text{gen}} = \frac{1}{8} \frac{V_{\text{ant}}^2}{R/Q Q_L} + \frac{1}{2} R/Q Q_L \left(\frac{V_{\text{ant}}}{R/Q} \frac{\dot{\varphi}}{\omega_r} - \frac{V_{\text{ant}}}{R/Q} \frac{\Delta\omega}{\omega_r} + \frac{1}{2} \Im(e^{-j\varphi} I_{b,\text{rf}}) \right)^2$$

- Compensates the bunch-by-bunch variation of the beam current along the ring via $\dot{\varphi}$

[21] D. Boussard: 'RF power requirements for a high intensity proton collider, parts 1 and 2', Proc. PAC'91, CERN-SL-91-20-DI-16, 1991.
[22] P. Baudrenghien and T. Mastoridis: 'Proposal for an RF roadmap towards ultimate intensity in the LHC', Proc. IPAC'12, 2012.

Phase and frequency modulation

Affect the RF phase and/or frequency at a given turn n

- A change in the RF phase results in a change in the frequency and vice versa

$$\Delta\varphi_{\mathrm{rf}} = 2\pi h \frac{\omega_{\mathrm{rf}}}{\omega_{\mathrm{rf},d}} = 2\pi h \frac{\omega_{\mathrm{rf},d} + \Delta\omega_{\mathrm{rf}}}{\omega_{\mathrm{rf},d}}$$

  - Phase and frequency modulation are in theory interchangeable
  - In practise, for the hardware implementation, one or the other might be preferred

- When a phase modulation $\Delta\varphi_{\mathrm{rf}}$ is applied, the RF frequency has to be corrected with the equivalent frequency change w.r.t. reference clock:

$$\Delta\omega_{\mathrm{rf},(n)} = \frac{d\Delta\varphi_{\mathrm{rf},(n)}}{dt} = \delta\Delta\varphi_{\mathrm{rf},(n)} \frac{\omega_{\mathrm{rf},d,(n)}}{2\pi h}$$

- Vice versa, when a frequency modulation $\Delta\omega_{\mathrm{rf}}$ is applied, the RF phase has to be corrected with:

$$\Delta\varphi_{\mathrm{rf}} = \frac{2\pi h \Delta\omega_{\mathrm{rf}}}{\omega_{\mathrm{rf,d}}}$$

Accumulated phase error

RF phase w.r.t. reference frame

- A phase error is accumulated when
  - The RF frequency is decoupled from the revolution frequency $\omega_{\mathrm{rf}} \neq h\omega_{\mathrm{rev}}$
  - There is explicitly an offset/modulation/noise added to the RF phase
- The phase at a given moment in time is

$$\varphi_{\mathrm{rf},k}(t_{(n)}) = \int_0^{t_{(n)}} \omega_{\mathrm{rf}}(\tau)\, d\tau + \varphi_{\mathrm{offset},k,(n)} + \varphi_{\mathrm{noise},k,(n)} = \sum_{i=1}^{n} \omega_{\mathrm{rf},k,(i)} T_{\mathrm{rev},(i)} + (t_{(n)} - t_{\mathrm{ref},(n)})\omega_{\mathrm{rf},k,(n)} + \varphi_{\mathrm{offset},k,(n)} + \varphi_{\mathrm{noise},k,(n)}$$

  - Subtracting multiples of 2π:

$$\varphi_{\mathrm{rf},k}(\Delta t_{(n)}) = \sum_{i=1}^{n} \frac{\omega_{\mathrm{rf},k,(i)} - h_{k,(i)}\omega_{\mathrm{rev},(i)}}{h_{k,(i)}\omega_{\mathrm{rev},(i)}} 2\pi h_{k,(i)} + \omega_{\mathrm{rf},k,(n)}\Delta t_{(n)} + \phi_{\mathrm{offset},k,(n)} + \phi_{\mathrm{noise},k,(n)}$$

- The RF phase is determined at $\Delta t = 0$:

$$\varphi_{\mathrm{rf},k} = \sum_{i=1}^{n} \frac{\omega_{\mathrm{rf},k,(i)} - h_{k,(i)}\omega_{\mathrm{rev},(i)}}{h_{k,(i)}\omega_{\mathrm{rev},(i)}} 2\pi h_{k,(i)} + \phi_{\mathrm{offset},k,(n)} + \phi_{\mathrm{noise},k,(n)}$$

Phase modulation at a single frequency

Resonant excitation using sine waves [23-25]

- Phase modulation added to the RF phase [26]:

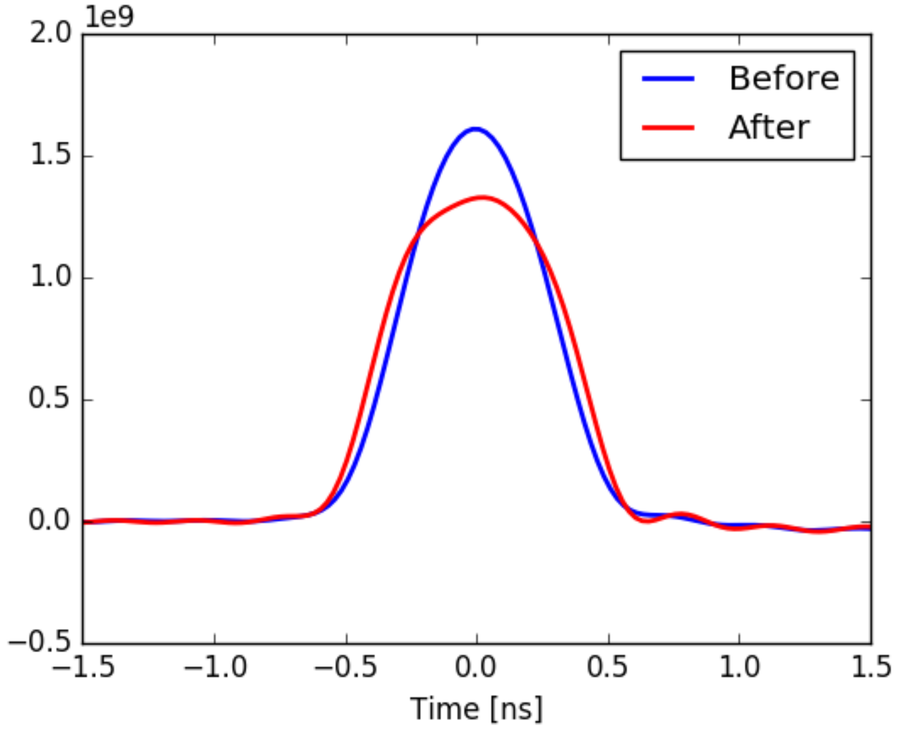
$$\Delta\varphi_{\mathrm{rf},(n)} = A \sin\left(2\pi \sum_{k=0}^{n} f_{\mathrm{mod},(k)} T_{\mathrm{rev},(k)}\right) + \varphi_{\mathrm{off},(n)}$$

- To correctly simulate the modulation, the RF frequency has to follow as:

$$\Delta\omega_{\mathrm{rf},(n)} = \frac{d\Delta\varphi_{\mathrm{rf},(n)}}{dt} = \delta\Delta\varphi_{\mathrm{rf},(n)} \frac{\omega_{\mathrm{rf},d,(n)}}{2\pi h}$$

Example: bunch flattening during LHC collisions

- Increase in bunch length to counteract SR w/o losses
- For this, modulate close to the core $0.98 f_{s0}$ with 0.6°



Measured bunch profile in the LHC before and after flattening

[23] S. Y. Lee: 'Accelerator Physics', World Scientific, 3rd Ed., 2012.

[24] C. Y. Tan and A. Burov: 'Phase modulation of the bucket stops bunch oscillations at the Fermilab Tevatron', PRAB **15**, 044401, (2012).

[25] E. Shaposhnikova *et al.*: 'Flat Bunches in the LHC', Proc. IPAC'14, Dresden, Germany, (2014).

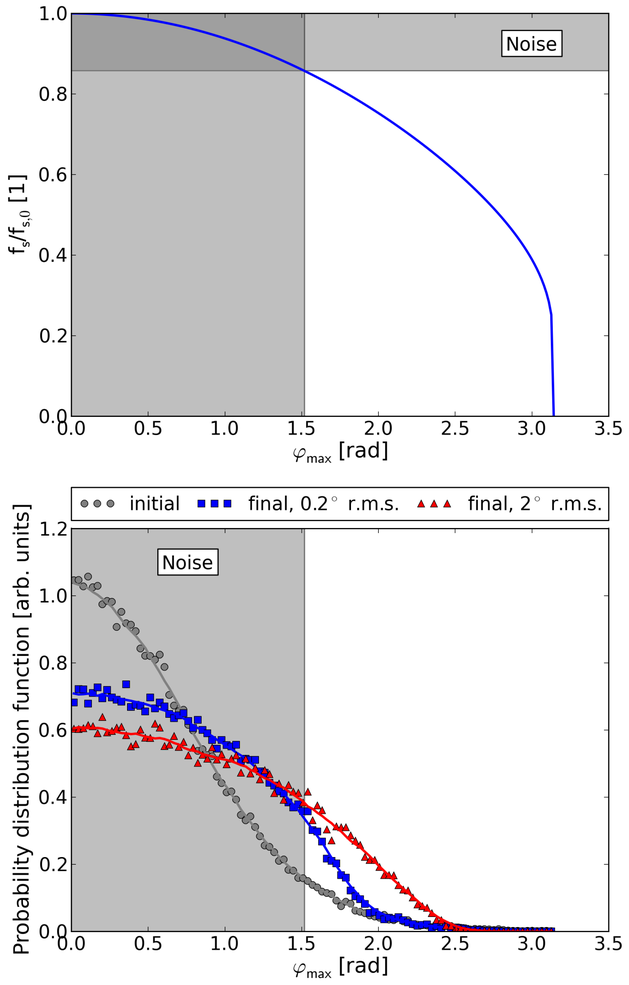
[26] S. Albright and D. Quartullo: Journal of Physics: Conference Series 1350, 012144 (2019).

RF phase noise

Controlled emittance blow-up can be achieved by injecting RF phase noise into the system [27,28]

- RF phase is changed turn by turn with a noise sample:
  $\Delta\varphi_{\mathrm{rf}} = \varphi_{\mathrm{noise}}(t_{(n)})$
  - Alternative: as a sum of sine-wave modulations
- Band-limited white noise can be injected to shape the bunch [29]
  - The spectrum of the noise determines which part of the bunch is affected
  - Should target the core region to avoid losses from the tails
  - The average phase noise is: $< \varphi_{\mathrm{noise}}(t_{(n)}) > = 0$
  - The r.m.s. phase noise depends on the spectral density of the noise

$$\varphi_{\mathrm{noise}}^{\mathrm{rms}} = \sqrt{\int S_{\varphi}(f)df}$$



Diffusion due to band-limited white noise

[27] S. Krinsky, J.M. Wang: 'Bunch diffusion due to RF noise', Part. Accel. 12, 107–117 (1982).

[28] G. Dôme: 'Diffusion due to RF noise', CERN Accelerator School '85, Oxford, UK, 370–401 (1985).

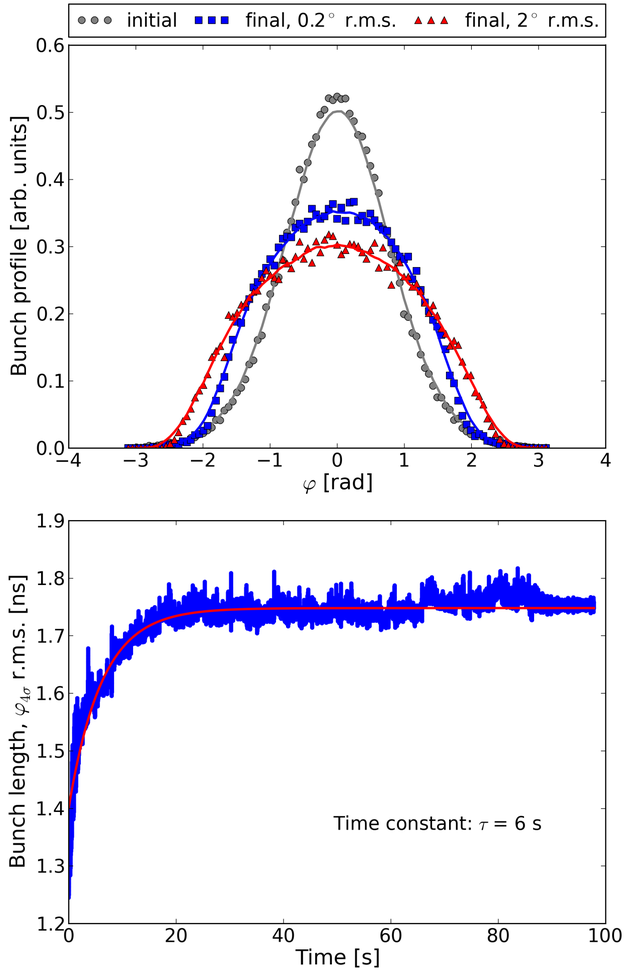
[29] T.Toyama: 'Uniform bunch formation by RF voltage modulation with a band-limited white signal', NIM A, 447, 317–327 (2000).

Emittance blow-up using RF phase noise

Diffusion due to RF phase noise

- In general, has to be simulated with particle tracking
- For some simplified cases, can use semi-analytical solvers
  - Determines final bunch profile and bunch length
- For short bunches, the r.m.s. bunch length evolution can be calculated applying the diffusion theory [30]

$$\sigma(t) = \sqrt{\sigma_0^2 + \int S_\varphi(f)df}$$



Diffusion due to band-limited white noise

[30] S.V. Ivanov: 'Longitudinal diffusion of a proton bunch under external noise', IHEP Preprint 92-43, Protvino (1992).

Momentum slip stacking

Reduction of bunch spacing by “interleaving” two batches [31-34]

- Capture two beams with two RF systems that have a slight frequency difference

$$V_{\mathrm{rf}} = V_{\mathrm{rf},1}\sin(\omega_{\mathrm{rf},1}t + \varphi_{\mathrm{rf},1}) + V_{\mathrm{rf},2}\sin(\omega_{\mathrm{rf},2}t - \varphi_{\mathrm{rf},2})$$

  - One RF system captures one beam each and the other RF system perturbs the other beam
- The small frequency difference w.r.t. the design frequency will result in a phase error

$$\Delta\varphi_{\mathrm{rf}} = \frac{2\pi h\Delta\omega_{\mathrm{rf}}}{\omega_{\mathrm{rf,d}}}$$

- Which at constant magnetic field translates to a slippage (drift) of

$$\frac{\Delta\omega_{\mathrm{rf}}}{\omega_{\mathrm{rf,d}}} = -\eta_0\frac{\Delta p}{p_{\mathrm{d}}}$$

  - The two beams slip inside the same beam pipe in opposite directions
- Once at the desired longitudinal and azimuthal position, the two beams are recaptured with a higher RF voltage, at the common frequency

[31] J. P. Burnet *et al.:* ‘Fifty years of the CERN Proton Synchrotron: Volume 1’, CERN Yellow Reports: Monographs, 2011.

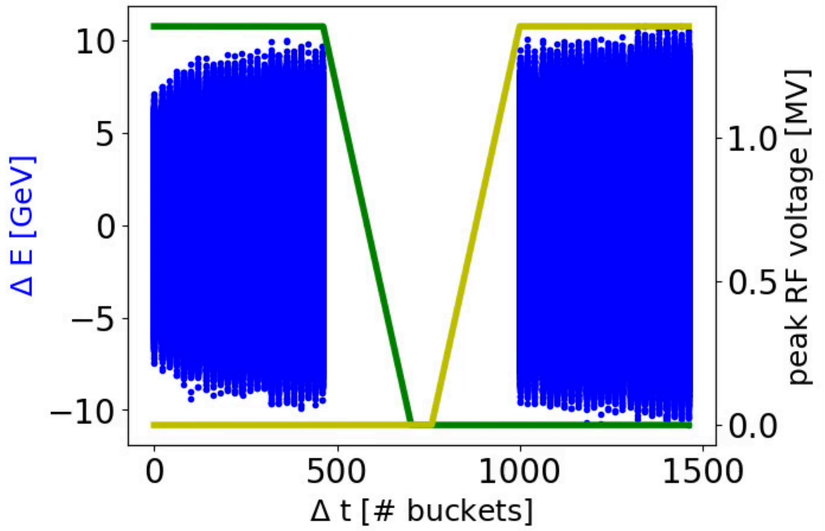
[32] D. Boussard and Y. Mizumachi: ‘Production of beams with high line-density by azimuthal combination of bunches in a synchrotron’, CERN-SPS-ARF-79-11, 1979.

[33] K. Seiya *et al.:* ‘Finalizing the Roadmap for the Upgrade of the CERN & GSI Accelerator Complex’, Proc. BEAM’07, 2007.

[34] R. Ainsworth *et al.,* Phys. Rev. Accel. Beams **22**, 020404 (2019).

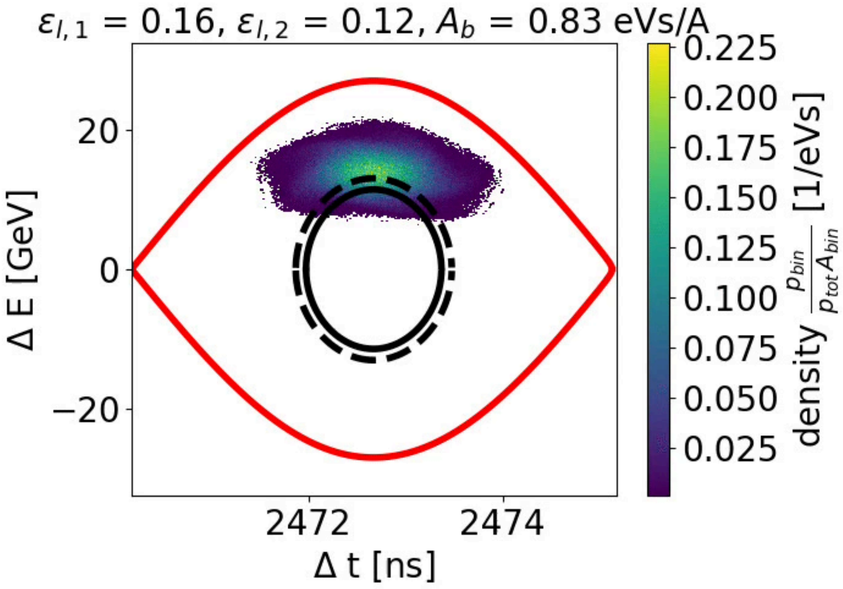
Momentum slip stacking example

Slip stacking for ion beams in SPS [35] in simulations



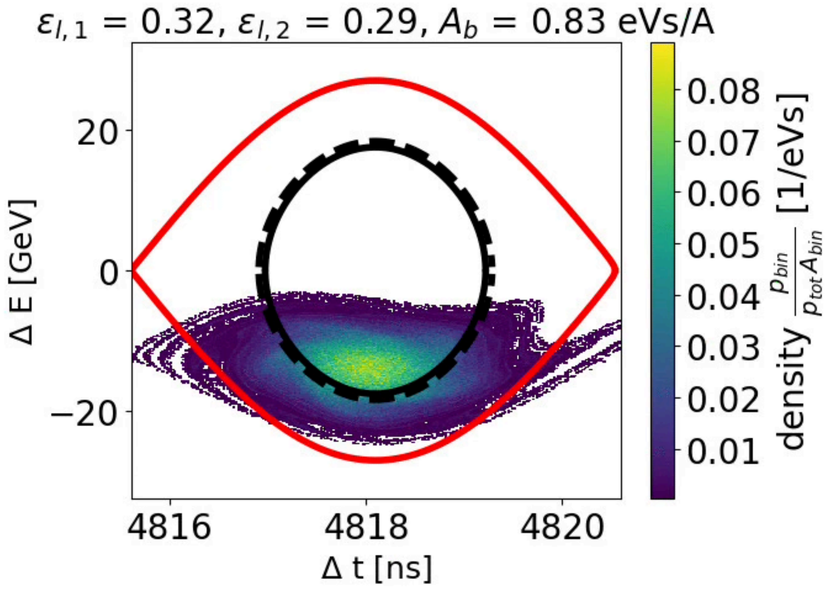
Slip stacking of 2 batches of 24 bunches

Courtesy of D. Quartullo



Re-capture and ramp:

first bunch



Re-capture and ramp:

last bunch

[35] J. Coupard, *et al.:* 'LHC Injectors Upgrade', Technical Design Report, CERN-ACC-2016-0041, 2016.

Global control loops (1)

Principle: take a beam measurable and compare it to its design value

- Correct the RF frequency $\Delta\omega_{\mathrm{TOT}} \equiv \omega_{\mathrm{rf}} - \omega_{\mathrm{rf,d}}$ (can be a sum of corrections from different loops)
- Beam phase loop
  - Measures the beam phase w.r.t. synchronous (design) phase $\Delta\varphi_{\mathrm{PL}} = \varphi_b - \varphi_d$
  - Used to reduce injection errors and undesired RF noise (improvement of beam lifetime)
- Synchronisation or frequency loop
  - Measures the RF frequency $\Delta\omega_{\mathrm{SL}} = \omega_{\mathrm{rf}} - \omega_{\mathrm{rf,d}} = \omega_{\mathrm{rf}} - h\omega_{\mathrm{rev}}$
  - Used for injection/extraction and to keep the RF frequency at its design value
- Radial loop (for transition-crossing machines)
  - Measures the radial position of the beam $\dfrac{\Delta R_{\mathrm{RL}}}{R_d} = \dfrac{\Delta\omega_{\mathrm{rf}}}{\omega_{\mathrm{rf,d}}}\dfrac{\gamma^2}{\gamma_T^2 - \gamma^s}$
  - Used to keep the beam centred and the RF frequency at its design value
- Longitudinal damper
  - Measures the beam phase w.r.t. RF phase $\Delta\varphi_{\mathrm{PL}} = \varphi_b - \varphi_d$
  - Used to damp phase errors bunch by bunch or batch by batch, depending on bandwidth
  - Counteracts coupled-bunch instabilities

Global control loops (2)

Measure at a given harmonic, apply corrections on all RF systems (if several)

- Updated RF frequency of system k:

$$\Delta\omega_{\mathrm{rf,k}} = \frac{h_k}{h_{\mathrm{meas}}}\Delta\omega_{\mathrm{TOT}}$$

- Updated RF phase of system k:

$$\Delta\varphi_{\mathrm{rf,k}} = 2\pi h_k \frac{\omega_{\mathrm{rf,k}}}{\omega_{\mathrm{rf,d,k}}}$$

- For each loop, the frequency correction is calculated from the measurable via a concrete transfer function
  - E.g. a simple gain (LHC PL)

    $\Delta\omega_{\mathrm{PL}} = -\, g_{\mathrm{PL}}\Delta\varphi_{\mathrm{PL}}$
  - E.g. transfer function (PSB PL) given in z-domain

    $H(z) = g\frac{b_0 + b_1 z^{-1}}{1 + a_1 z_{-1}}$
  - translates in time domain to

    $\Delta\omega_{\mathrm{PL,(n+1)}} = -\, a_1\omega_{\mathrm{PL,(n)}} + g(b_0\Delta\varphi_{\mathrm{PL,(n+1)}} + b_1\Delta\varphi_{\mathrm{PL,(n)}})$

Measurement of RF phase

The RF phase is measured by extracting the RF component from the beam profile

- Convolve the beam profile with the RF wave
  - Averaging over the whole beam

$$f(t) = \int_{\lambda_{\min}}^{\lambda_{\max}} \cos(\omega_{\mathrm{rf}}(t-\tau) - \varphi_{\mathrm{rf}})\lambda(\tau)d\tau = \cos(\omega_{\mathrm{rf}}t)\int_{\lambda_{\min}}^{\lambda_{\max}} \cos(\omega_{\mathrm{rf}}\tau + \varphi_{\mathrm{rf}})\lambda(\tau)d\tau + \sin(\omega_{\mathrm{rf}}t)\int_{\lambda_{\min}}^{\lambda_{\max}} \sin(\omega_{\mathrm{rf}}\tau + \varphi_{\mathrm{rf}})\lambda(\tau)d\tau$$

- The beam phase is determined from the coefficients of the sine and cosine components

$$\varphi_b \equiv \arctan\left(\frac{\int_{\lambda_{\min}}^{\lambda_{\max}} \sin(\omega_{\mathrm{rf}}\tau + \varphi_{\mathrm{rf}})\lambda(\tau)d\tau}{\int_{\lambda_{\min}}^{\lambda_{\max}} \cos(\omega_{\mathrm{rf}}\tau + \varphi_{\mathrm{rf}})\lambda(\tau)d\tau}\right)$$

Example: LHC phase and synchro loop

Exact algorithms are machine dependent. A concrete example: LHC, where PL & SL work together [36]

- Correction of beam phase loop

  $\Delta\omega_{\mathrm{PL}} = -\, g_{\mathrm{PL}} \Delta\varphi_{\mathrm{PL}}$

  - Reaction time: 5 turns $g_{\mathrm{PL}} = 1/(5T_{\mathrm{rev}})$
- Correction of synchronisation loop

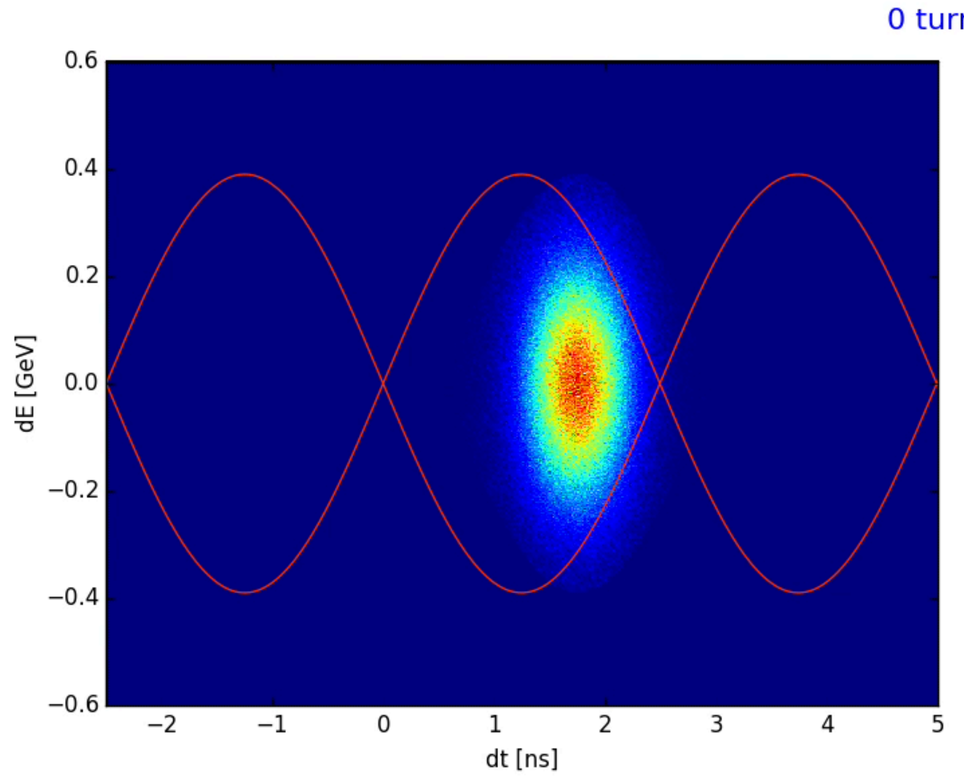
  $\Delta\omega_{\mathrm{SL}} = -\, g_{\mathrm{SL}}(y + a\,\Delta\varphi_{\mathrm{rf}})$

  - Reaction time: 50 turns $g_{\mathrm{SL}} = 1/(50T_{\mathrm{rev}})$
  - Here y is a recursive function

    $y_{(n+1)} = (1-\tau)y_{(n)} + (1-a)\tau\Delta\varphi_{\mathrm{rf}}$ with $y_{(0)} = 0$
  - and τ, a are functions of the synchrotron frequency

    $$a(\omega_s) \equiv 5.25 - \frac{\omega_s}{2\pi\ 20\ \mathrm{Hz}}$$

    $$\tau(Q_s) \equiv 2\pi Q_s \sqrt{\frac{a}{1 + \frac{g_{\mathrm{PL}}}{g_{\mathrm{SL}}}\sqrt{\frac{1+1/a}{1+a}}}}$$



LHC capture with phase error

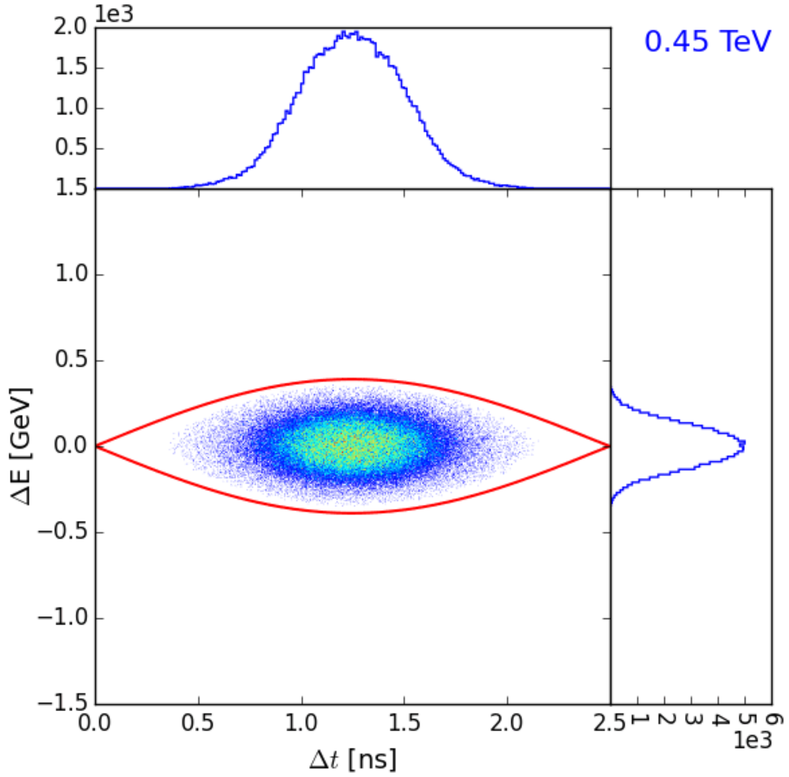
Phase loop and synchro loop acting

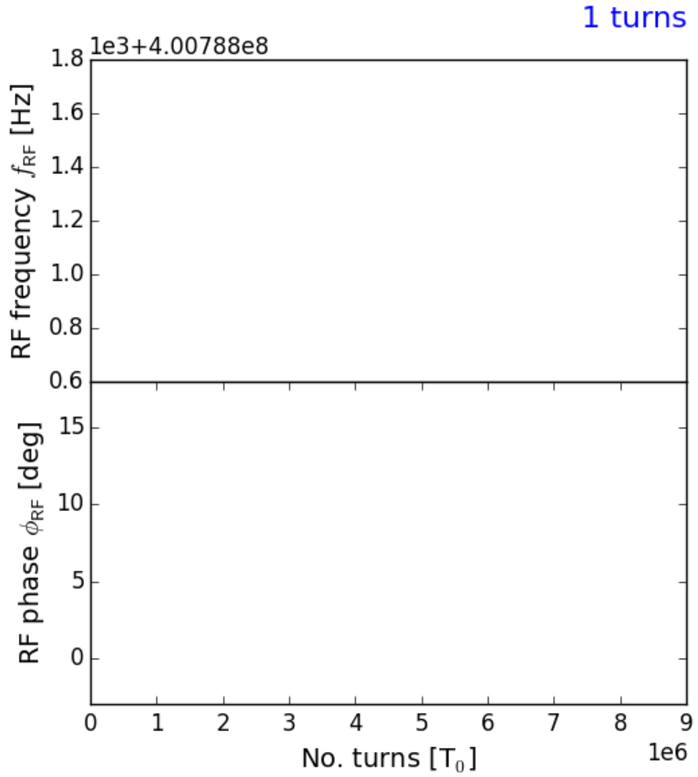
[36] P. Baudrenghien: 'The LHC Low Level Loops', unpublished, 2008.

Coupling control loops and noise

A concrete example: LHC blow-up during the ramp

- RF phase noise injection with PL and SL on
- Bunch length feedback regulating the strength (r.m.s. amplitude) of the noise





Local control loops

Control an RF chain from transmitter to cavity

- Principal function: regulate the RF voltage amplitude and phase $\vec{V}_{\text{ant}}$ to be the design (set point) value $\vec{V}_{\text{rf,d}}$
  - Voltage amplitude and phase are **now arrays** over one turn!
- Transmitter regulation
  - Sends an input gain to the amplifier chain; final amplifier can be klystron, tetrode, IOT…
  - Polar loop: regulates the phase and amplitude of the transmitter
- Clamping/protection loops
  - Protect from overdriving e.g. a klystron beyond saturation

From the beam point of view

- Reduce the cavity impedance at the fundamental (RF) frequency
  - Reduction of beam- (and generator-) induced voltage
  - Filter design machine dependent
- If necessary, damp coupled-bunch instability
  - Coupled bunch feedbacks
  - E.g. reduce the side-bands of the cavity impedance at $nf_{\text{rev}} + kf_s$ (comb filters)

Example: SPS cavity controller

- Normal-conducting travelling wave cavities, fixed tune
- Antenna voltage: beam and generator part [37]

$$V_{\mathrm{ant}} = I_{\mathrm{b}} Z_{\mathrm{b}} + I_{\mathrm{gen}} Z_{\mathrm{gen}}$$

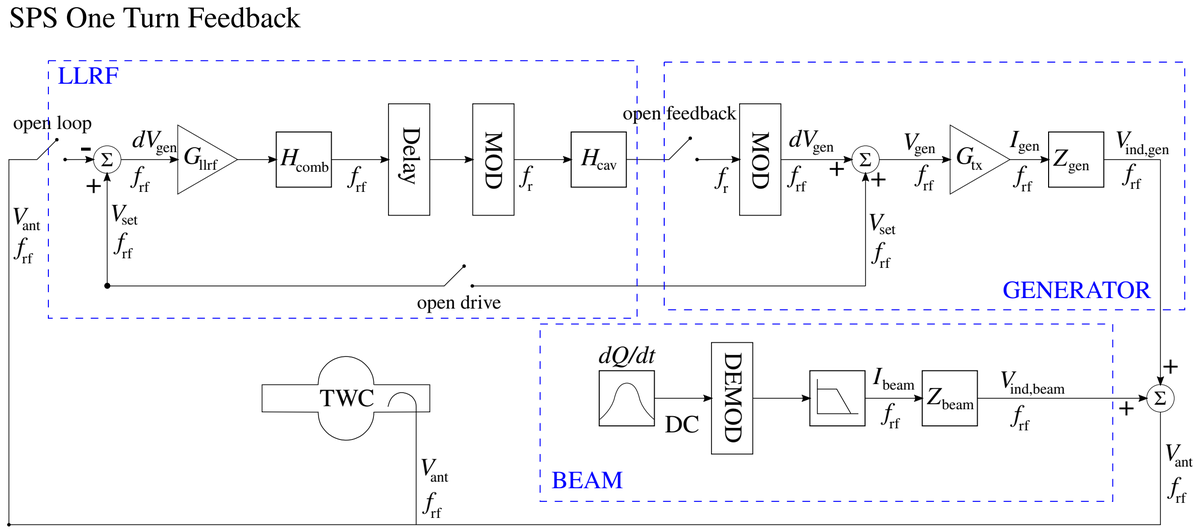
- Beam-induced voltage

$$Z_{\mathrm{b}}(\omega) = \frac{\rho l^2}{8}\left[\left(\frac{\sin\left(\frac{1}{2}\tau(\omega-\omega_r)\right)}{\frac{1}{2}\tau(\omega-\omega_r)}\right)^2 - 2i\frac{\tau(\omega-\omega_r)-\sin(\tau(\omega-\omega_r))}{(\tau(\omega-\omega_r))^2}\right] +$$

$$+\frac{\rho l^2}{8}\left[\left(\frac{\sin\left(\frac{1}{2}\tau(\omega+\omega_r)\right)}{\frac{1}{2}\tau(\omega+\omega_r)}\right)^2 + 2i\frac{\tau(\omega+\omega_r)-\sin(\tau(\omega+\omega_r))}{(\tau(\omega+\omega_r))^2}\right]$$

- Generator-induced voltage

$$Z_{\mathrm{gen}}(\omega) = l\sqrt{\frac{\rho Z_0}{2}}\left[\frac{\sin\left(\frac{1}{2}\tau(\omega-\omega_r)\right)}{\frac{1}{2}\tau(\omega-\omega_r)} + \frac{\sin\left(\frac{1}{2}\tau(\omega+\omega_r)\right)}{\frac{1}{2}\tau(\omega+\omega_r)}\right]$$



Model of the SPS cavity controller after [38]

Sampling time of the system: $T_s = 25ns$ (3564 points / turn)

[37] G. Dôme: 'The SPS acceleration system travelling wave drift-tube structure for the CERN SPS', CERN-SPS-ARF-77-11, 1977.

[38] P. Baudrenghien and T. Mastoridis: 'I/Q model of the SPS 200 MHz travelling wave cavity and feedforward design', CERN-ACC-NOTE-2020-0032, 2020.

Example: LHC cavity controller

- Tuneable superconducting cavities
  - One klystron of 300 kW feeding a cavity
- Antenna voltage: as in the cavity-transmitter-beam model
- Direct RF feedback composed of
  - Analog high-pass branch: gain G_a, delay τ_a

$$y^{(n)} = \left[1 - \frac{T_s}{\tau_a}\right] y^{(n-1)} + G_a(x^{(n)} - x^{(n-1)})$$

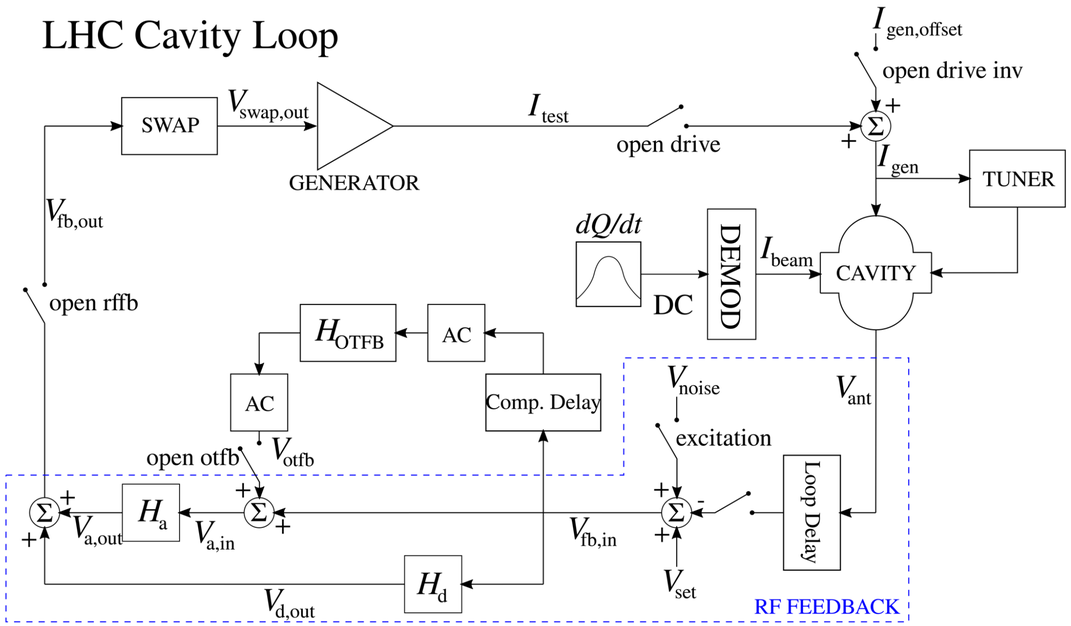
  - Digital low-pass branch: gain G_d, delay τ_d

$$y^{(n)} = \left[1 - \frac{T_s}{\tau_d}\right] y^{(n-1)} + G_a G_d e^{i\Delta\varphi_{\mathrm{ad}}} \frac{T_s}{\tau_d} x^{(n-1)}$$

- One-turn delay feedback (comb filter)
  - Boosts the analog gain: gain G_o, scaling factor α, samples per turn N

$$y^{(n)} = \alpha y^{(n-N)} + G_o(1-\alpha)x^{(n-N)}$$

- Other: tuning and clamping loops



Model of the LHC cavity controller after [39]

Sampling time of the system: $T_s = 25\mathrm{ns}$ (3564 points / turn)

[39] J. Holma: 'The model and simulations of the LHC 400 MHz cavity controller', CERN-AB-Note-2007-012, 2007.

Coupling global and local control loops

General problem

- Local control loops sample one turn usually with a fixed integer amount of samples

  $$T_s = \frac{T_{\mathrm{rev}}}{N}$$

- Global control loops act on the RF frequency and unlock it from the revolution frequency

  $$\omega_{\mathrm{rf}} \neq h\omega_{\mathrm{rev}}$$

Solution to make sampling continuous

- Modulate signals to $N\omega_{\mathrm{rev}}$ for tracking the local control loops
- Demodulate signals back to $\frac{N}{h}\omega_{\mathrm{rf}}$ for tracking the beam EOMs

Content

Beam tracking basics

- Tracking of beams
- Discretisation

Longitudinal tracking

- Reference frame
- Equations of motion
- Periodicity

Effects on particle energy

- Collective effects and impedance
- Multi-turn wake
- Synchrotron radiation

RF modelling

- Tune and beam loading
- Global and local control loops

Particle distribution

- Observables and parametrisation
- Matching distributions

6D effects

- Some examples

Code design, optimisation and benchmarks

- Good practises

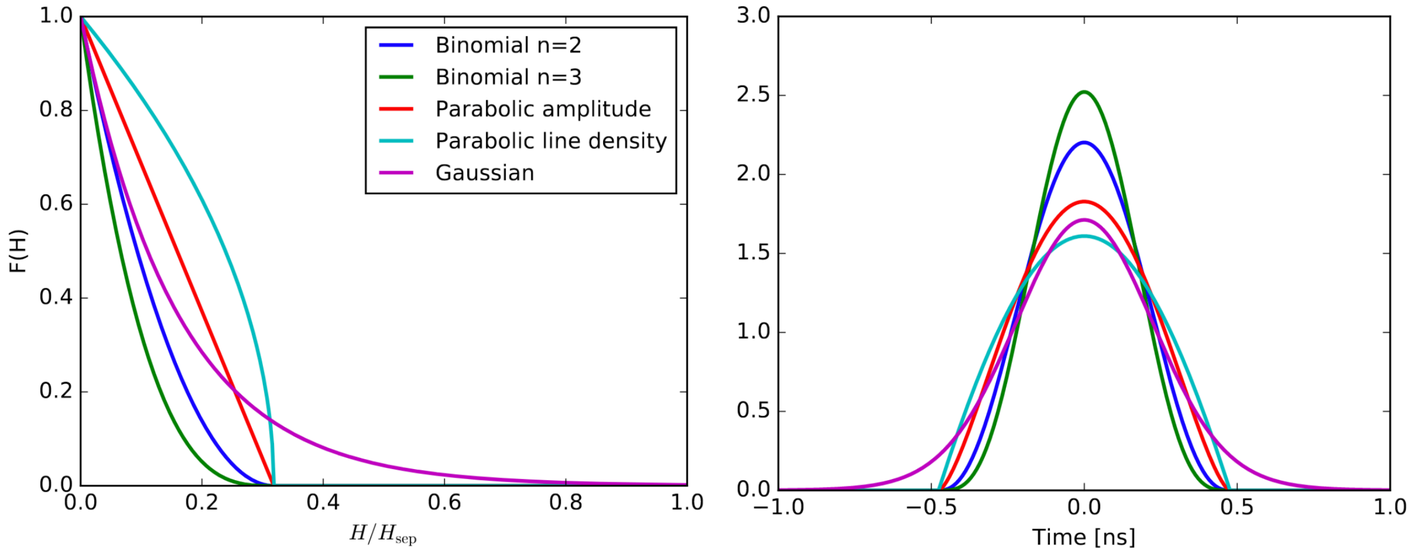
Beam distribution

In a conservative system

- The distribution function is conserved along a given trajectory (Liouville's theorem) $F(\Delta t, \Delta E) = \text{const.}$
- So the phase-space distribution function is function of the Hamiltonian $F(\Delta t, \Delta E) = F(H)$

Some distributions useful for (proton) bunches [7]

- Binomial: $F(H) = F_0 \left(1 - \frac{H}{H_0}\right)^n$
- Parabolic amplitude: $F(H) = F_0 \left(1 - \frac{H}{H_0}\right)$
- Parabolic line density: $F(H) = F_0 \left(1 - \frac{H}{H_0}\right)^{1/2}$
- Gaussian: $F(H) = F_0 e^{-\frac{2H}{H_0}}$



Some basic distribution functions (left) and line densities (right) from [34]

[7] J. Esteban Müller: 'Longitudinal intensity effects in the CERN Large Hadron Collider ', CERN-THESIS-2016-066, 2016.

Beam observables

Beam profile

- A basic measurable; tails are harder to measure though. Some basic analytical profiles are
  - Gaussian: $\lambda(\Delta t) = \lambda_0 e^{\frac{-\Delta t^2}{\sigma_t^2}}$
  - Binomial with a full bunch length of τ_{full} and a coefficient of μ: $\lambda(\Delta t) = \lambda_0 \left(1 - 4\left(\frac{\Delta t}{\tau_{\text{full}}}\right)^2\right)^{\mu+\frac{1}{2}}$

Bunch length

- Full bunch length τ_{full}: difficult to measure in the machine
- R.m.s. bunch length σ_t: prone to wrong results when applying r.m.s. calculation, even if only one particle drifts far from the bunch; best to be calculated using Gaussian fit
- Practical: 4-sigma equivalent FWHM bunch length: $\tau_{4\sigma} \equiv \frac{2}{\sqrt{2\ln 2}}\tau_{\text{FWHM}}$
  - For a binomial bunch: $\tau_{\text{full}} = \left(1 - \frac{1}{2^{1/(\mu+\frac{1}{2})}}\right)^{-1/2} \frac{\sqrt{2\ln 2}}{2}\tau_{4\sigma}$

Bunch emittance

- Corresponding to a given bunch length definition
- The area inside the phase-space trajectory at the bunch length $\varepsilon \equiv \oint_{\Delta t=\tau/2} \Delta E(\Delta t)\, d(\Delta t)$

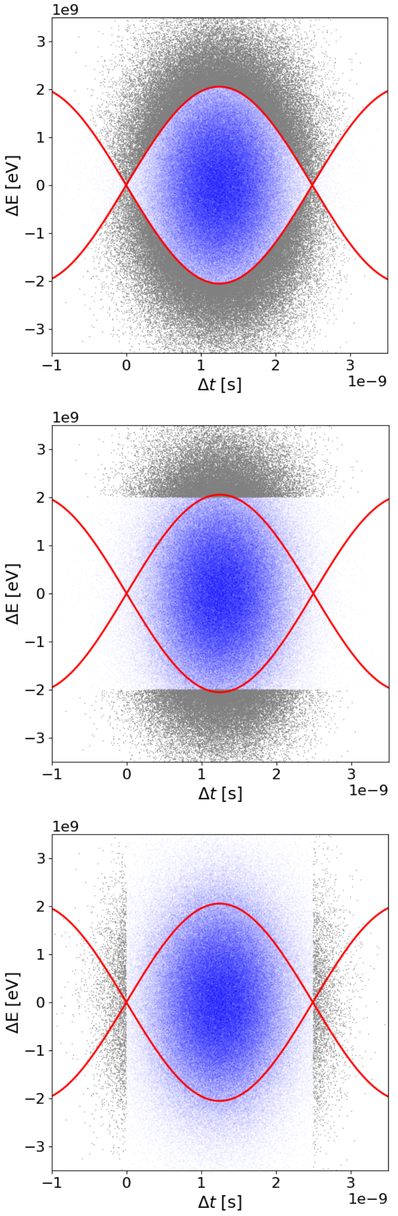
Beam losses

A few common ways to determine beam losses in simulations

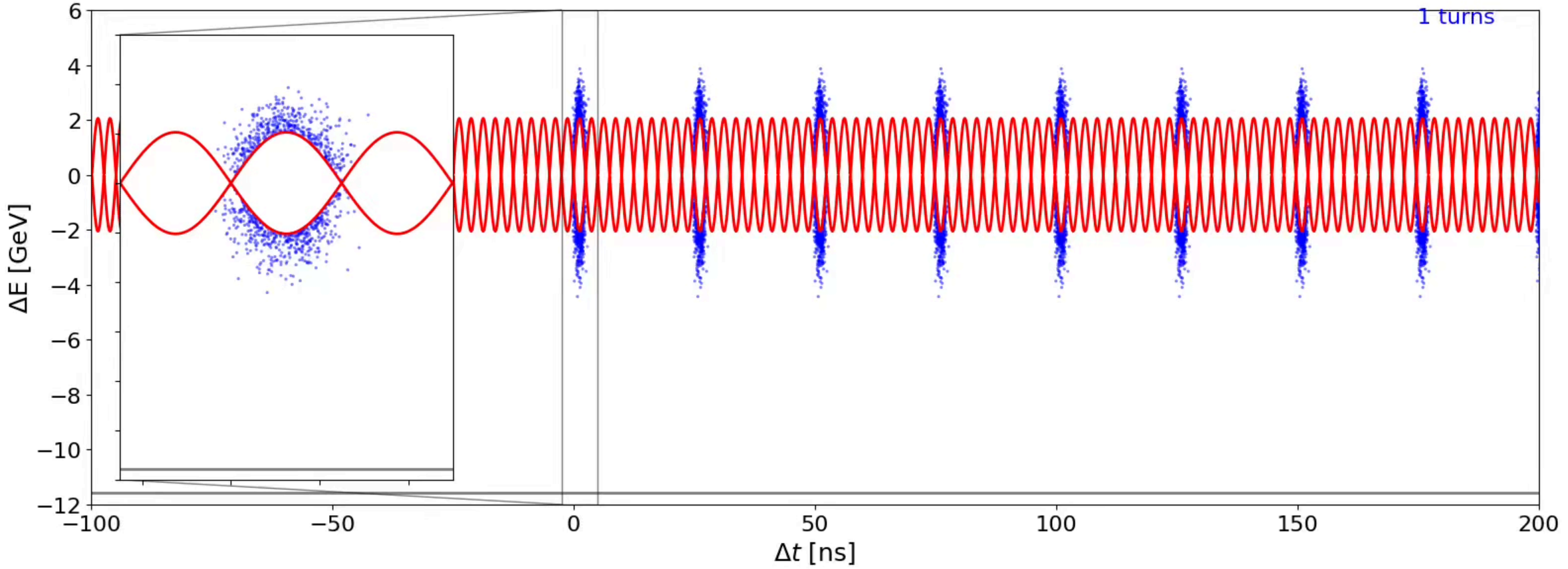
- By separatrix, with or without intensity effects
  - This is the strict definition of longitudinal losses, hard to measure though in a real machine
  - Computationally heavy, especially with induced voltage added
- By particle energy, corresponding to a certain aperture of the machine
  - Simple and easy to compare with the machine
- By particle arrival time, corresponding to a longitudinal region, e.g. abort gap
  - Simple and easy to compare with the machine
- Beam profile
  - Integrated beam profile
  - Useful when high-quality measured profiles are available (with correction of cable transfer function, if needed)
  - Can be tricky to apply cuts on the tails to remove measurement noise → not suitable for very low losses

Tip: use an attribute of the beam to mark it lost

- Sometimes it can be useful to still track the lost particles



LHC losses with synchrotron radiation



Off-momentum losses in LHC at 6.8 TeV

Debunched beam loses energy through synchrotron radiation

Generating a bunch distribution

Example: generation of a bi-Gaussian bunch with N_m marco-particles (without intensity effects)

- Initialise the seed of the random number generator (RNG)

  Tip: use a fixed seed to make sure that next time you can **re-generate** exactly the same random distribution!

- Generate two independent arrays of random numbers with a normal (Gaussian) distribution of $\sigma = 1$

  $\mathtt{RAND}_{1,k}$ and $\mathtt{RAND}_{2,k}$ where $k = 0,1,\ldots, N_{m-1}$

  - The time coordinates of the bunch with a user-given bunch length of σ_t are

    $$\Delta t_k = \sigma_t \mathtt{RAND}_{1,k} + \frac{\varphi_s - \varphi_{\mathrm{rf}}}{\omega_{\mathrm{rf}}}$$

  - The “matched" energy coordinates of the bunch are proportional to the bucket height, which for single RF:

    $$\Delta E_k = \sigma_E r_{2,k} = \sqrt{\frac{V_{\mathrm{rf}} E_d \beta_d^2}{\pi h \,|\, \eta_0 \,|}(\cos\varphi_b - \cos\varphi_s + (\varphi_b - \varphi_s)\sin\varphi_s)}\,\mathtt{RAND}_{2,k}, \text{ with } \varphi_b = \omega_{\mathrm{rf}}\sigma_t + \varphi_s$$

- Generate a normal distribution from random numbers $\mathrm{RAND}_3 \in (1,1)$ using the cumulative distribution fct:

  $$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \rightarrow F_{\mathrm{CDF}} = \int_0^r r e^{-r^2/2} dr = 1 - e^{-r^2/2} \in (0,1) \text{ for } r \in [0,\infty)$$

  $$\Rightarrow x_k = \sqrt{-2\ln(\mathtt{RAND}_{3,k})}$$

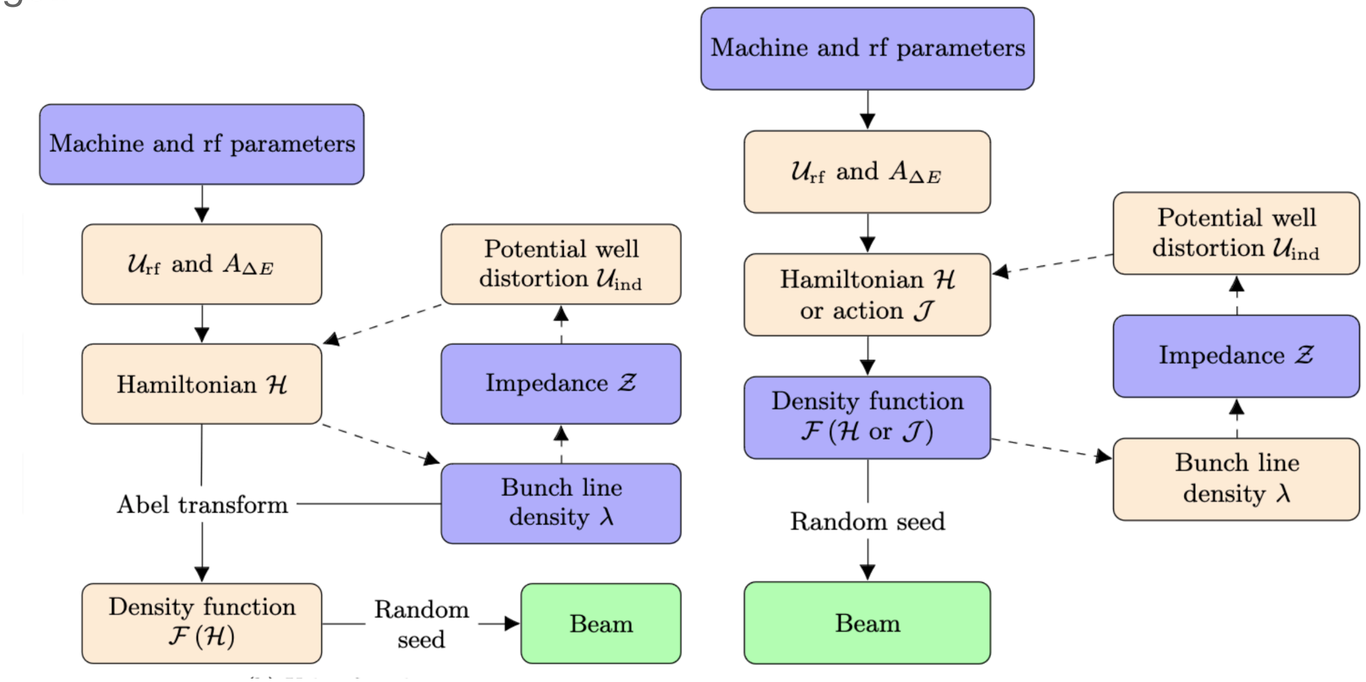
Matching with intensity effects

Recursive algorithms

- Need to define a function to minimise
  - Can match, e.g. density function or beam profile, up to the desired accuracy
- In an iterative process, one needs to calculate potential well and distribution that act back on each other
  - Also need to find the minimum of the potential well to place the bunch, which can be tricky with strong induced voltage or many RF harmonics → always check what you get!

Example: flow chart of BLonD algorithms to match line density (left) or density function (right)

Courtesy of A. Lasheen

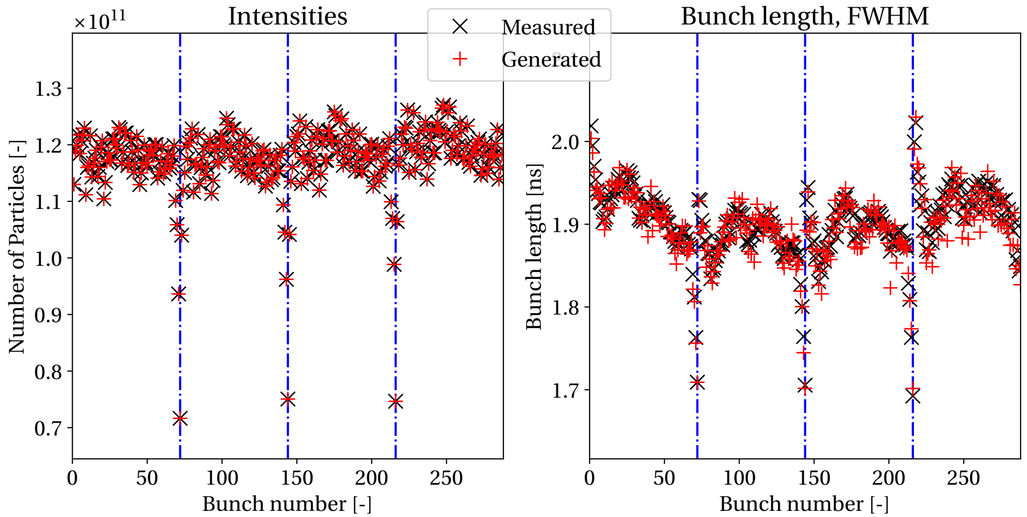


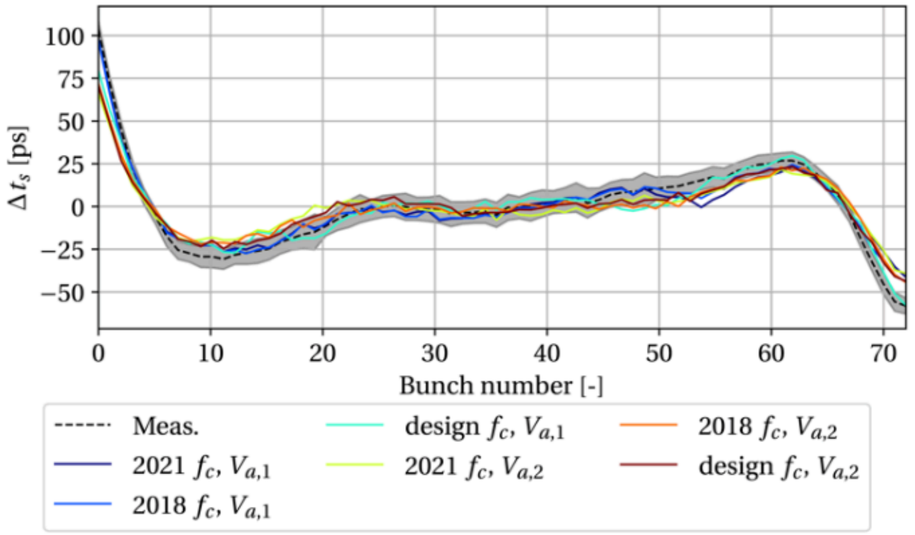
Matching with intensity effects and loops

Matching a beam distribution gets even trickier when loops are acting

- Example: SPS flattop distribution with intensity effects and SPS one-turn delay feedback
  - Matching measured beam profile, with measured bunch-by-bunch bunch length and intensity
  - First matched with intensity effects, then tracked with feedback

Tip: can use an adiabatic ramp of feedback gain to better match the beam





Left: matching of bunch-by-bunch intensities and bunch lengths based on measured data

Right: simulating the one-turn delay feedback to get the right bunch-by-bunch longitudinal offset from beam loading

Courtesy of B. Karlsen-Bæck

Content

Beam tracking basics

- Tracking of beams
- Discretisation

Longitudinal tracking

- Reference frame
- Equations of motion
- Periodicity

Effects on particle energy

- Collective effects and impedance
- Multi-turn wake
- Synchrotron radiation

RF modelling

- Tune and beam loading
- Global and local control loops

Particle distribution

- Observables and parametrisation
- Matching distributions

6D effects

- Some examples

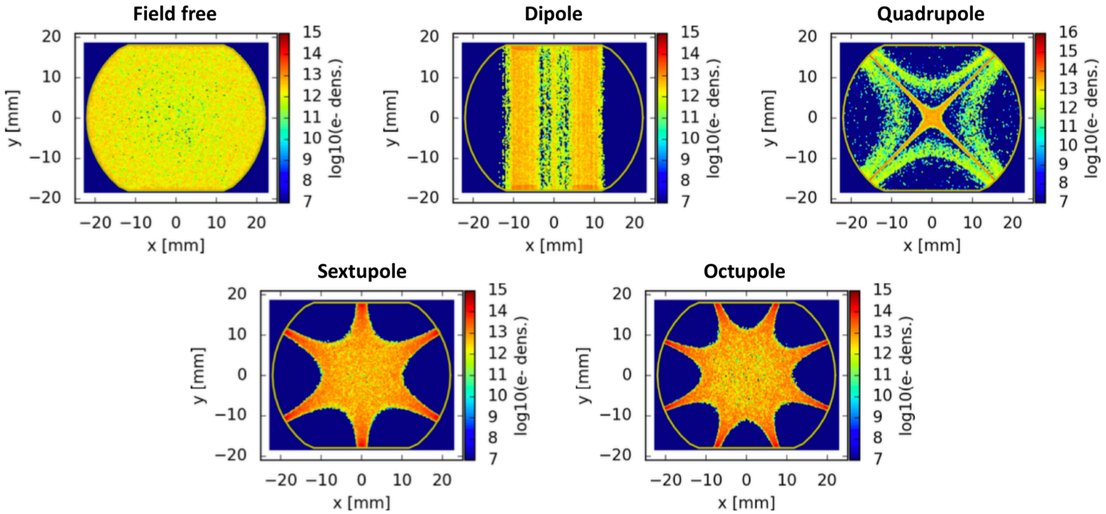
Code design, optimisation and benchmarks

- Good practises

Electron cloud

In proton machines, electrons that are ripped off the wall of the machine elements through the repetitive passage of protons

- Its simulation requires two particle types
  - Electrons: oscillating around a fixed location
- Modelling requirements
  - Modelling surface properties (secondary emission yield)
  - Requires full 6D simulations of interaction between particles
  - Basically a two-component plasma → PIC simulation needed
  - Overall space charge
  - Scattering and collisions
- Example: PyECLOUD [40] with PyHEADTAIL [41]



Electron distribution in LHC arc components with different magnetic field configurations from [42]

[40] PyECLOUD simulation suite, CERN, https://github.com/PyCOMPLETE/PyECLOUD/wiki

[41] PyHEADTAIL simulation suite, CERN, https://github.com/PyCOMPLETE/PyHEADTAIL/wiki

[42] G. Iadarola and G. Rumolo: 'Electron cloud effects', CERN-2018-003-CP, 2018.

Intra-beam scattering

Coulomb scattering among beam particles

- Single scattering with large change of momentum (Touschek effect) and multiple scatterings
  - Similar to PIC codes
  - Slightly different treatment in electron (short bunch) and hadron (long bunch) machines
- Leads to emittance blow-up in all three coordinates (longitudinal, horizontal, vertical)
  - Emittance evolution can be calculated numerically for many cases [43]
- Ideally, a full 6D, stochastic simulation required
  - Some approximations can be done to include the effect as an energy kick [44]

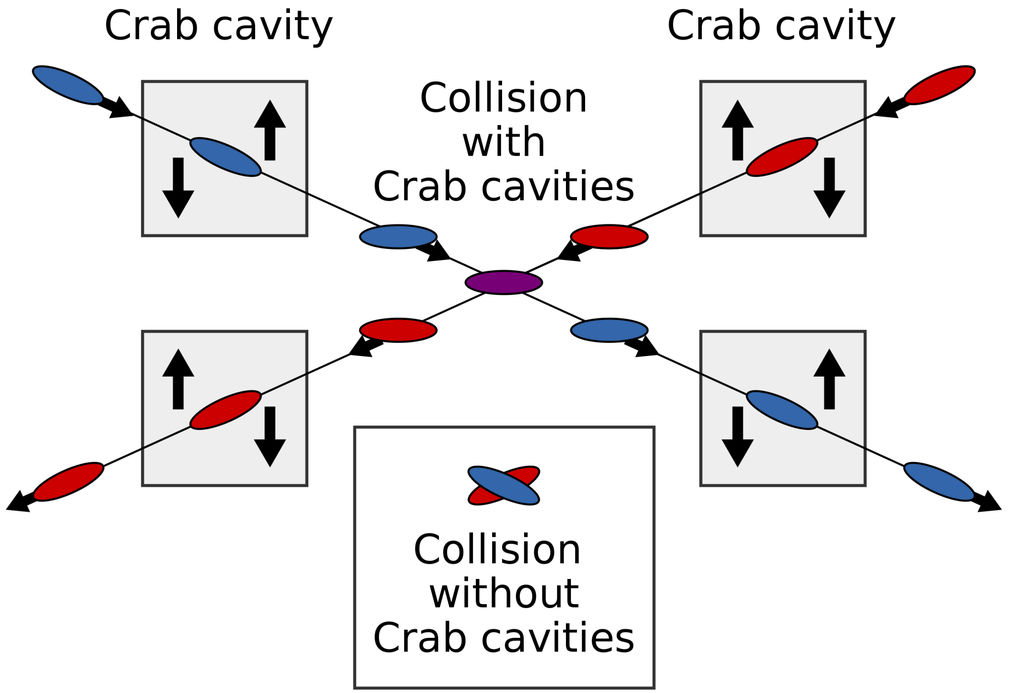
[43] S. Nagaitsev: 'Intrabeam scattering formulas for fast numerical evaluation', Phys. Rev. Accel. Beams **8**, 064403, 2005.

[44] R. Bruce *et al.*: 'Time evolution of the luminosity of colliding heavy-ion beams in BNL Relativistic Heavy Ion Collider and CERN Large Hadron Collider', Phys. Rev. Accel. Beams **113**, 091001, 2010.

Crab cavities

Crab crossing: time-dependent transverse kick

- Passing RF deflecting cavity at zero phase → bunch head and tail are deflected in opposite directions
- Couples all planes, full 6D beam dynamics simulator needed
  - E.g. BLonD with XSuite, PyHEADTAIL or MADX
- Several numeric tools have been developed for this purpose [45]



Collision scheme for HL-LHC [45] with crab cavities from [46]

[45] A. Alekou *et al.*: 'Numerical tools for crab cavity simulations', CERN-ACC-NOTE-2020-0042, 2020.
[46] O. Aberle *et al.*: 'High-Luminosity Large Hadron Collider (HL-LHC): Technical design report', CERN-2020-010, 2020.

Content

Beam tracking basics

- Tracking of beams
- Discretisation

Longitudinal tracking

- Reference frame
- Equations of motion
- Periodicity

Effects on particle energy

- Collective effects and impedance
- Multi-turn wake
- Synchrotron radiation

RF modelling

- Tune and beam loading
- Global and local control loops

Particle distribution

- Observables and parametrisation
- Matching distributions

6D effects

- Some examples

Code design, optimisation and benchmarks

- Good practises

Code design: structure

Choosing the right code structure for what you want to simulate

- Always the same pipeline?
  - Can limit to an input file with parameters
  - Can heavily optimise the sequence of operations for a given hardware platform
  - Need to fix the possibilities of interaction with the computational core (when to output etc.)
- Flexible design? Modularity?
  - Has the advantage of being easily extensible for new applications
  - More difficult to optimise the computational core
  - Need to make sure that computational sequence is physically correct for all possible scenarios
- What platform are you going to simulate on?
  - For a wide range of platforms (from office PC to multi-CPU and multi-GPU clusters), separate as much as possible physics code from platform-dependent code
  - This will help to avoid rewriting physics code every time you use a new hardware for your simulations

Code design: language

Choosing the right coding language

- High-level language
  - Fast prototyping, easier-to-read code, more “equation-like”
  - More forgiving (e.g. can mix float and int), but more hidden caveats too
  - Comes potentially with some modules for CPU and GPU acceleration
  - E.g. Python
- Low-level language
  - Can be more optimised, if written correctly
  - Slow prototyping, requires compiling
  - Less forgiving when written, but clearer what’s happening under the hood
  - Code needs to be adapted to the hardware architecture
  - E.g. C++, CUDA

```python
def kick(dt: np.ndarray, dE: np.ndarray, voltage: np.ndarray,
         omega_rf: np.ndarray, phi_rf: np.ndarray,
         charge: float, n_rf: int, acceleration_kick: float) -> None:
    '''
    Function to apply RF kick on the particles with sin function
    '''

    voltage_kick = charge * voltage

    for j in range(n_rf):
        dE += voltage_kick[j] * np.sin(omega_rf[j] * dt + phi_rf[j])

    dE[:] += acceleration_kick
```

```cpp
extern "C" void kick(const double * __restrict__ beam_dt,
                     double * __restrict__ beam_dE, const int n_rf,
                     const double * __restrict__ voltage,
                     const double * __restrict__ omega_RF,
                     const double * __restrict__ phi_RF,
                     const int n_macroparticles,
                     const double acc_kick){
int j;

// KICK
for (j = 0; j < n_rf; j++)
#pragma omp parallel for
        for (int i = 0; i < n_macroparticles; i++)
                beam_dE[i] = beam_dE[i] + voltage[j]
                        * fast_sin(omega_RF[j] * beam_dt[i] + phi_RF[j]);

// SYNCHRONOUS ENERGY CHANGE
#pragma omp parallel for
    for (int i = 0; i < n_macroparticles; i++)
        beam_dE[i] = beam_dE[i] + acc_kick;

}
```

Kick equation implemented in Python (top) and C++ (bottom)

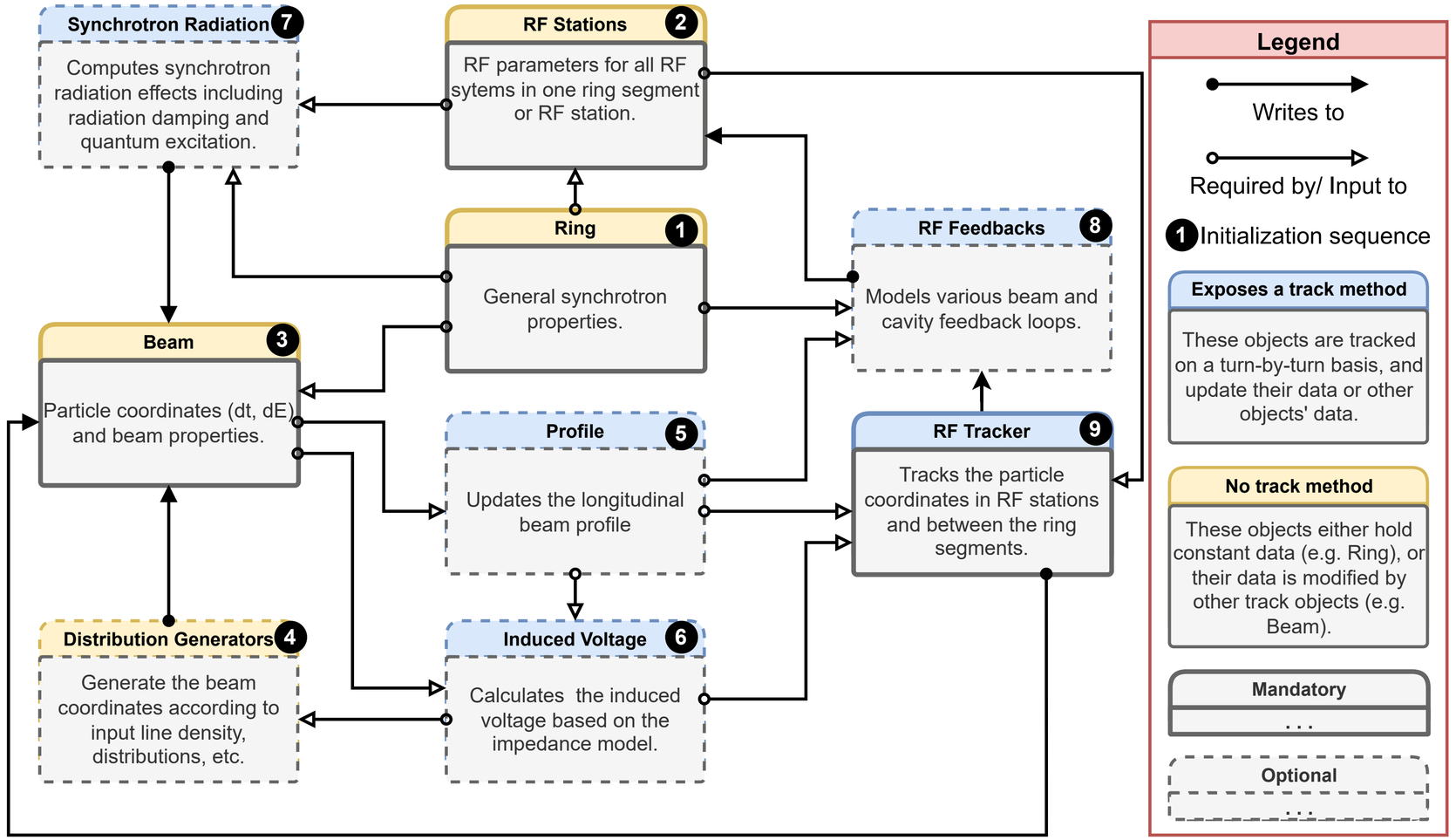
Example: BLonD structure

BLonD structure

- Python with C++ and CuPy core
- Three modules as data containers (Ring, RF Station, Beam)
- Trackable objects acting on the data containers
- Modular, object-oriented structure

BLonD code structure

Courtesy of K. Iliakis



Good practises for writing code

Please no spaghetti code!

- Good practises for readability
  - Object oriented, using inheritance, breaking down into functions
  - Adding comments and documentation

Simulation suite, input files, argument parser, scripts, …

- Separate what's
  - Really computation → simulation suite
  - Configuration, interaction with the simulation or special usage → input files
  - Options that should be input from the command line → argument parser
  - Tools for jobs submission on a cluster → submission scripts

Input/output: store what you need

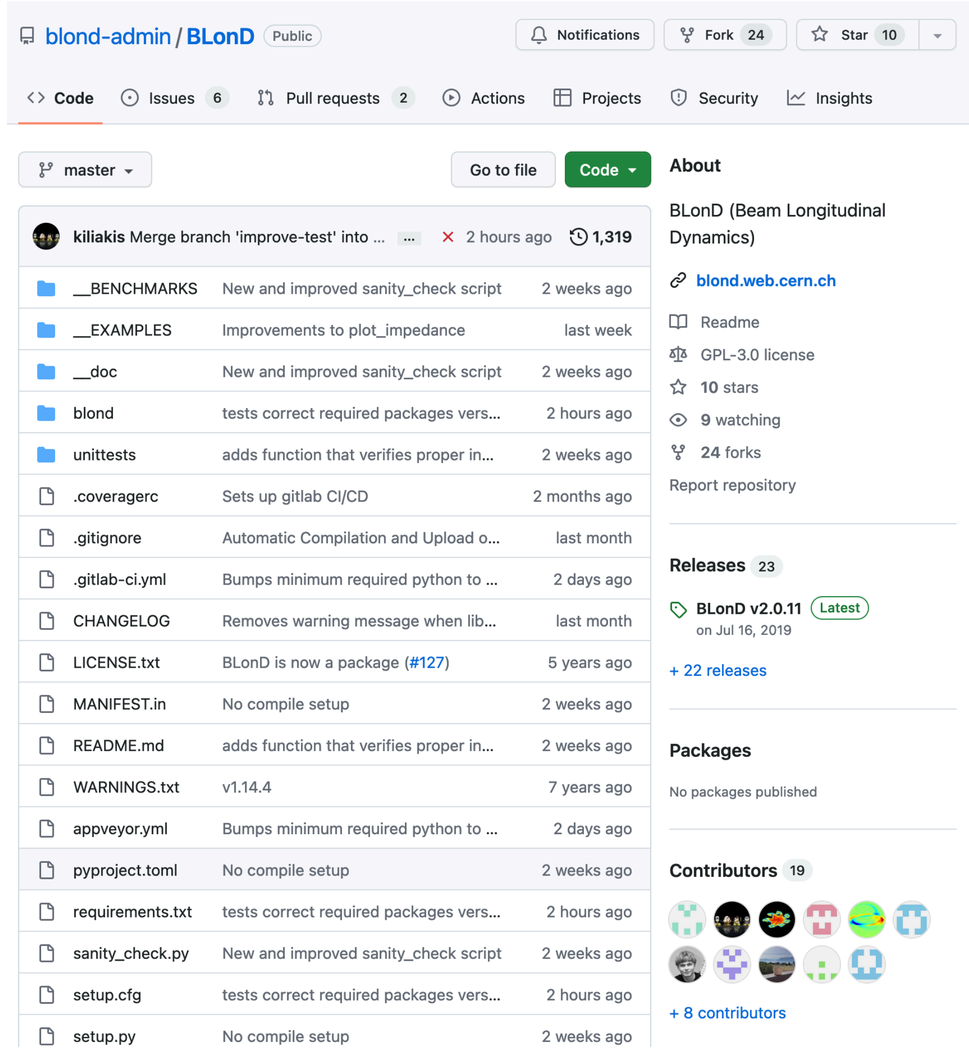
- Reading and writing into files as well as plotting takes a lot of time

Tip: Minimise the frequency of I/O and think beforehand what output you really need from your simulation

Coding in teams

Repository usage

- Main repository with production version
  - Versioning and tagging (git, svn)
    - Avoid using “my_script_final_version_2.py”
  - Documentation (e.g. github pages)
  - Installation procedures
- Forks for different people
- Branches for beta versions
- Pull requests to make a new version being production
  - Gives time for team members to test and react
- Issues for bug reports, suggestions and improvements



Computationally heavy simulations

Optimised core code

- Profile and separate core code where most computation is done
- Optimise core code for hardware
- Minimise interaction with core (plots etc.), separate computation from analysis
- Prefer a compiled language
  - Python can be interfaced with a compiled, high-performance language
- Revise data structures and algorithms → e.g. reducing complexity from N^2 to $N \log N$ (FFT vs DFT)

Tools and techniques

- Vectorisation: modern processors have vector units that can do e.g. 4 operations in parallel
  - Compilers take automatically care of it (in most cases)
- Multi-threading: can be used even on your office PC (e.g. OpenMP library)
- Hardware accelerators (GPUs, FPGAs…)
- Data types: single (32-bit) vs double (64-bit) precision floating point numbers
  - Can you afford lower precision? Test it before you use
- Look for third party libraries (e.g. FFT), don't reinvent the wheel

Memory-heavy simulations

More common nowadays, as processing speed increases faster than memory access speed

- Platform dependent
- Saving a lot of files, doing a lot of plots: minimise exchange of data and try to fit data in memory
- Large arrays to operate on (hundreds of millions of coordinates)

Good practises

- Minimise writing to files
  - In clusters, network file systems are used (NFS), which is slower than writing to SSD or HDD
  - Compression levels (.txt, .csv vs .h5, .tar, .gz) - depends on application and hardware
- Tile your input (operate on a subset of input), if possible: fitting into different cache memory levels
  - Analogy: fetching information from a library - shelf - book - brain
- Change access pattern and data structures (regular better than random)
  - E.g. histogram of multi-bunched beam: better organise by bunches

```
struct pointlist3D {
    float x[N];
    float y[N];
    float z[N];
};
struct pointlist3D points;
float get_point_x(int i) { return points.x[i]; }
```

```
struct point3D {
    float x;
    float y;
    float z;
};
struct point3D points[N];
float get_point_x(int i) { return points[i].x; }
```

Example of access pattern difference with data structures

Top: structure of arrays

Bottom: array of structures

Submission to clusters

How and what to take care with

- Interactive
  - For debugging, developing, or small studies
- Batch mode (offline)
  - For long studies, scans of simulations

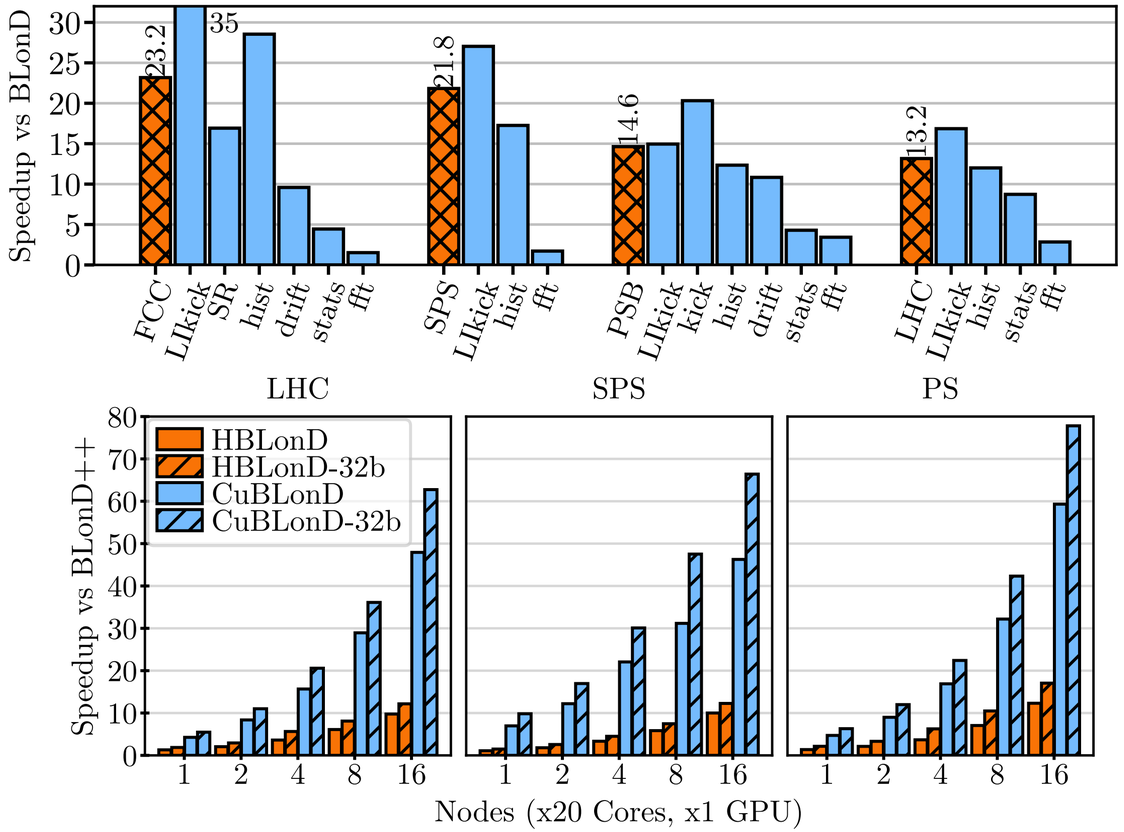
Schedulers

- Take care of priorities, quotas, queues based on resources, memory, expected runtime
- Some common schedulers: slurm, htcondor, PBS, LSF…
  - Different set of commands to be used with each
- Basic linux knowledge necessary
  - Visualisation shouldn’t happen on the cluster

From BLonD over HBLonD to CuBLonD

An optimisation example: BLonD optimisation [47]

- BLonD (Python only) → BLonD++
  - Move computational core to C++
  - Optimise time-consuming code regions
    - Use of highly efficient libraries (e.g. sine function)
  - Runtime reduction: 1 day → 80 minutes
- BLonD++ → HBLonD (x10 speed-up)
  - From single-core to distributed version
  - Using the Message Passing Interface (MPI)
  - High-performance computing techniques
    - Dynamic load balancing, data- and task parallelism
- BLonD++ → CuBLonD (x10-17 speed-up)
  - GPU-accelerated version
  - CUDA and CuPy versions available



Top: speed-up of BLonD++ vs BLonD

Bottom: speed-up of HBLonD and CuBLonD vs BLonD++

[47] K. Iliakis: 'Large-scale software optimization and micro-architectural specialization for accelerated high-performance computing', PhD Thesis, 2022.

Courtesy of K. Iliakis

Testing, benchmarking, comparing

Essential for the trust in the code: continuous verification of the correctness of the results

- No 100 % guarantee that the results are correct, but our confidence in simulation results increases over time if we test constantly [48]
  - There is a difference between test, benchmark, and comparison!
- Test
  - A small piece of the entire code
  - Input and output can be well defined
- Benchmark
  - A global problem simulated with the code
  - What should be the outcome is exactly known either from theory or from reality (measurements)
- Code-to-code comparison
  - In the absence of benchmark possibilities (no known solution for the problem)
  - Can simulate the same problem with two different codes that potentially use different computational methods to model the same thing

[48] H. Timko *et al.*: 'Why perform code-to-code comparisons: a vacuum arc discharge simulation case study', Contrib. Plasma Phys. **52**, 4, 2012.

Unit tests

Test basic functions

- Against theoretical expected value
- Against a value obtained with earlier versions of the code (i.e. correctness not guaranteed)
- Python has a module dedicated to unit tests
  - Can be invoked on-demand or integrated to a repository

```python
def test_losses_longitudinal_cut(self):

    longitudinal_tracker = RingAndRFTracker(self.rf_params, self.beam)
    full_tracker = FullRingAndRF([longitudinal_tracker])
    try:
        matched_from_distribution_function(self.beam,
                                           full_tracker,
                                           distribution_exponent=1.5,
                                           distribution_type='binomial',
                                           bunch_length=1.65e-9,
                                           bunch_length_fit='fwhm',
                                           distribution_variable='Hamiltonian')
    except TypeError as te:
        self.skipTest("Skipped because of known bug in deepcopy. Exception message %s"
                      % str(te))
    self.beam.losses_longitudinal_cut(0., 5e-9)
    self.assertEqual(len(self.beam.id[self.beam.id == 0]), 0,
                     msg='Beam: Failed losses_longitudinal_cut, first')

    self.beam.dt += 10e-9
    self.beam.losses_longitudinal_cut(0., 5e-9)
    self.assertEqual(len(self.beam.id[self.beam.id == 0]),
                     self.beam.n_macroparticles,
                     msg='Beam: Failed losses_longitudinal_cut, second')
```

```python
class testBeamClass(unittest.TestCase):

    # Run before every test
    def setUp(self):

        # Bunch parameters
        # ----------------

        N_turn = 200
        N_b = 1e9  # Intensity
        N_p = int(2e6)  # Macro-particles

        # Machine parameters
        # ------------------
        C = 6911.5038  # Machine circumference [m]
        p = 450e9  # Synchronous momentum [eV/c]
        gamma_t = 17.95142852  # Transition gamma
        alpha = 1. / gamma_t**2  # First order mom. comp. factor

        # Define general parameters
        # -------------------------
        self.general_params = Ring(C, alpha, p, Proton(), N_turn)

        # Define beam
        # -----------
        self.beam = Beam(self.general_params, N_p, N_b)

        # Define RF section
        # -----------------
        self.rf_params = RFStation(self.general_params, [4620], [7e6], [0.])

    # Run after every test
    def tearDown(self):

        del self.general_params
        del self.beam
        del self.rf_params
```

Example of python unit tests: several functions can be tested within the same test case

Code-to-code comparisons

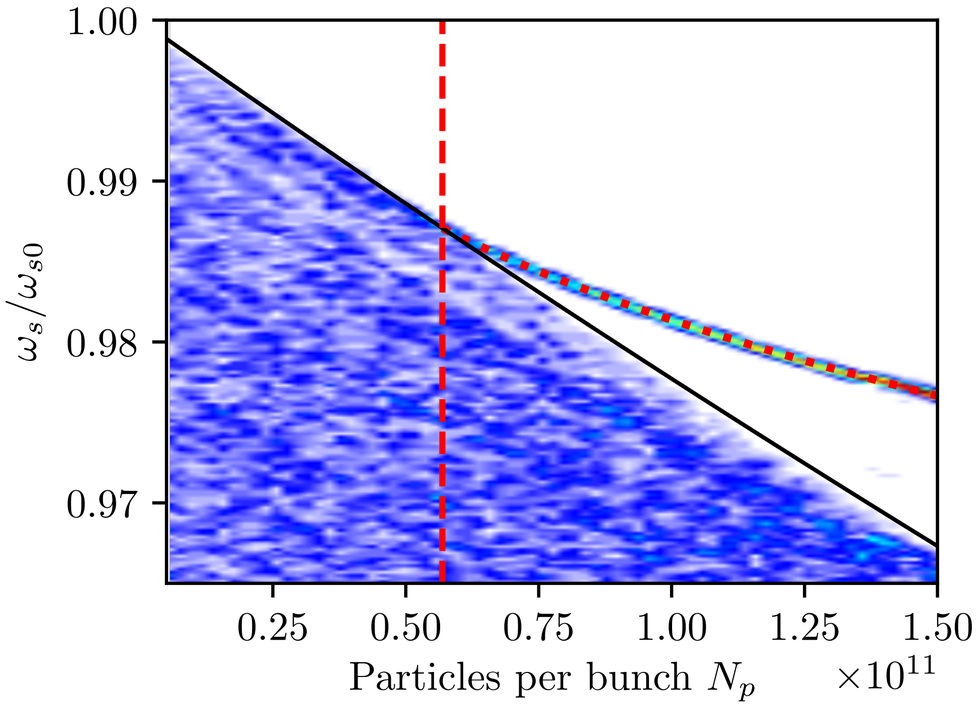
Test a given physics problem with two independent simulation suites

- May often use different physics assumptions
- Make sure you're actually simulating the same
  - Parameters, assumptions…
- Getting agreement is reassuring
  - Even if both codes could be wrong!

Example of loss of Landau damping simulation with the macroparticle suite BLonD (blue) vs the semi-analytical solver MELODY (red)

The comparison assumes a single broad-band resonator of $\Im(Z/n) = 0.07\Omega, Q = 1, f_r = 4\text{GHz}$ *in the LHC*

Courtesy of I. Karpov



Benchmark

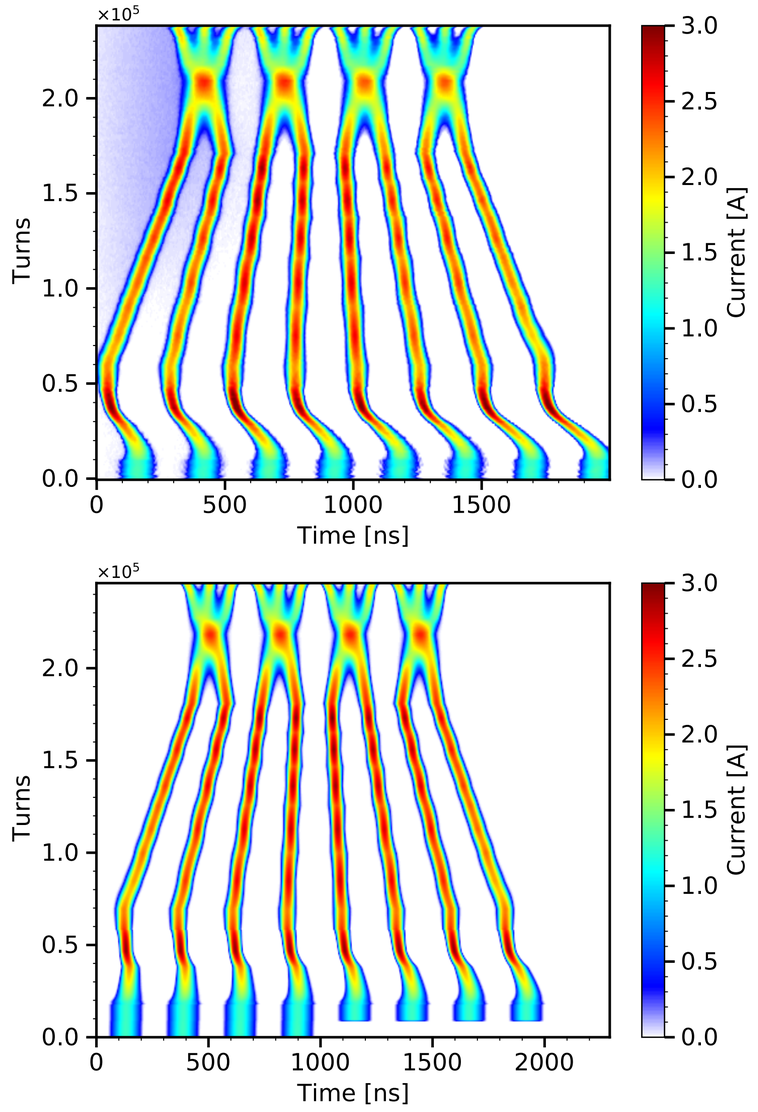
Test a physics problem against a known theoretical solution or measurements

- The good thing about accelerator physics is we can test our theories!
- Measurements vs simulations
  - Try to make the observables as similar as possible
  - Use the same data analysis methods
  - Try to model measurement imperfections in simulations

Benchmarking bunch profiles (shown in waterfall plots) in the PS during RF manipulations

Top: measured profiles, bottom: simulated profiles

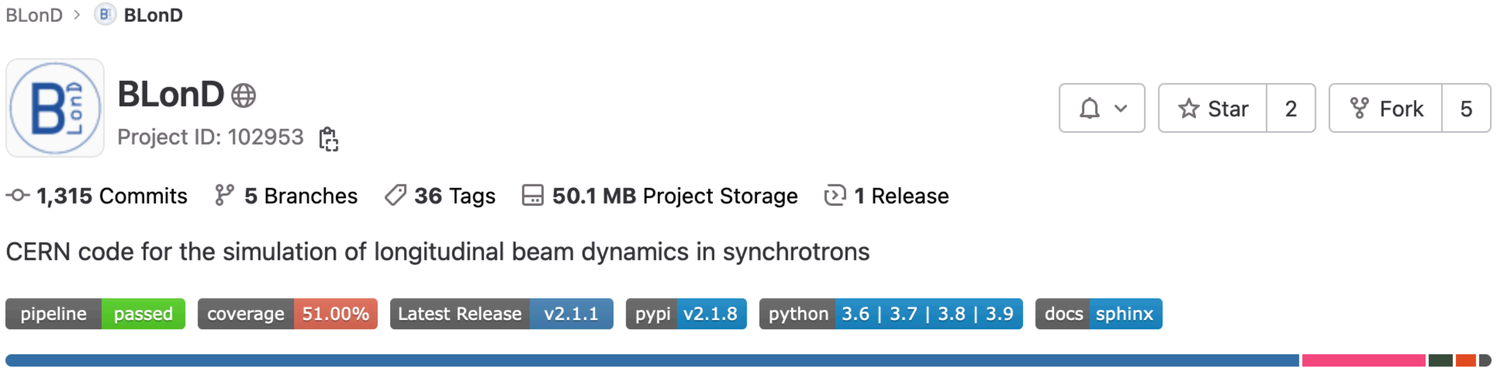
Courtesy of A. Lasheen



Continuous integration

Maintain trust in the code: make sure what worked yesterday still works tomorrow

- Coverage: add as many unit tests and complex tests as possible to cover all possible regions of the code
- Define platforms and software versions the code should be compatible with
- Make sure the code compiles (build passes)
- Tools available
  - Automatic pipeline execution on github/gitlab whenever a new version is deployed (package index like PyPI)
    - E.g. Github actions, Gitlab pipelines, Travis, AppVeyor…
  - Coverage
    - Based on the scripts ran in the pipeline
    - Or using a tool like codecov, dotCover (JetBrains)



Continuous integration tools active for BLonD

Courtesy of K. Iliakis

Summary

Beam tracking basics

- Tracking of beams
- Discretisation

Longitudinal tracking

- Reference frame
- Equations of motion
- Periodicity

Effects on particle energy

- Collective effects and impedance
- Multi-turn wake
- Synchrotron radiation

RF modelling

- Tune and beam loading
- Global and local control loops

Particle distribution

- Observables and parametrisation
- Matching distributions

6D effects

- Some examples

Code design, optimisation and benchmarks

- Good practises

Now you have the basic tools to build your own simulator of longitudinal beam dynamics!

Happy coding and thank you for your attention!