RF-Track particle tracking code

Andrea Latina
andrea.latina@cern.ch

28th April 2023

Table of contents

RF-Track highlights

- It handles 3-D field maps of static and radio-frequency electro-magnetic fields:
  - with the capability of simulating of standing waves, backward- and forward-travelling waves, as well as static fields
- It's fully relativistic; it allows backtracking
- Can transport any particle, and was successfully used with: electrons, positrons, protons, antiprotons, ions, at various energies, recently photons, and muons
- Implements high-order integration algorithms
- Implements collective effects: space-charge effects, wakefields, beam-loading, ...
- Tracks mixed-species beams, with collective effects
- It's flexible and fast

Reference: `https://zenodo.org/record/4580369`

Download: `https://gitlab.cern.ch/rf-track`

RF-Track: minimalistic and physics-oriented

RF-Track only contains C++ code that provides accelerator-physics concepts:

- Flexible accelerator and beam models
- Accurate integration of the equations of motion
- Interpolation of field maps which is Maxwell's-equations-compliant
- Collective effects

For "all the rest" (e.g., integration algorithms, random number generation, etc.), it relies on two robust and renowned open-source libraries:

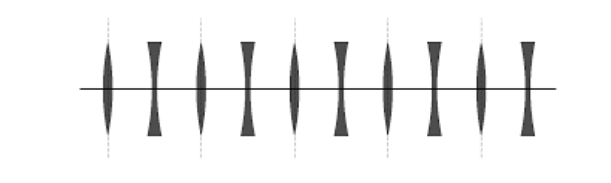
- GSL, "Gnu Scientific Library", provides a wide range of mathematical routines such as high-quality random number generators, ODE integrators, linear algebra, and much more
- FFTW, "Fastest Fourier Transform in the West", arguably the fastest open-source library to compute discrete Fourier transforms

RF-Track tracking environments

Lattice: integration in space

- A list of elements
- Tracks the particles element-by-element, along the longitudinal direction

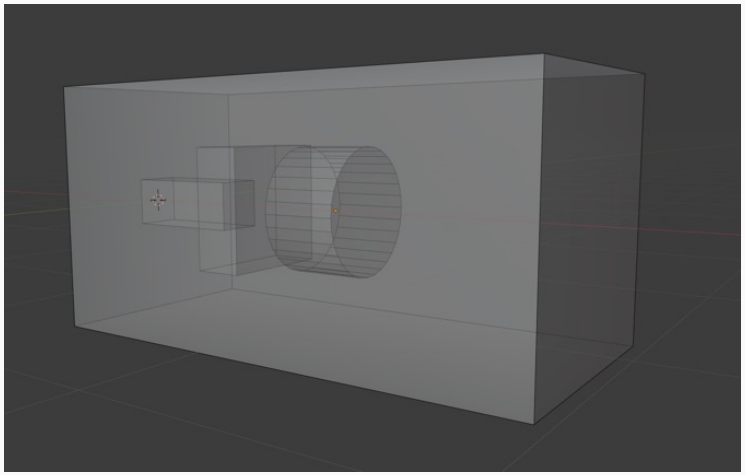
Lattice



Volume: integration in time

- A portion of 3D space
- Elements can be placed at arbitrary locations in the volume
- Element misalignment can be specified using the Euler angles (raw, pitch, yaw)
- Allows element overlap
- Allows particles creation
- Can simulate cathodes and field emission
- Considers the effect of mirror charges at the cathode

Volume



Beam tracking in space and in time

1. Beam moving in space: `Bunch6d()`
   - All particles have the same S position
   - Each particle's phase space is

$$(x\ [\text{mm}],\ x'\ [\text{mrad}],\ y\ [\text{mm}],\ y'\ [\text{mrad}],\ t\ [\text{mm/c}],\ P\ [\text{MeV/c}])$$

   Integrates the equations of motion in dS: $S \rightarrow S + dS$ (moves the beam element by element

2. Beam moving in time: `Bunch6dT()`
   - All particles are considered at same time t
   - Each particle's phase space is

$$(X\ [\text{mm}],\ Y\ [\text{mm}],\ S\ [\text{mm}],\ P_x\ [\text{MeV/c}],\ P_y\ [\text{MeV/c}],\ P_z\ [\text{MeV/c}])$$

   - Particles can have $P_z < 0$ or even $P_z = 0$: particles can move backward

   Integrates the equations of motion in dt: $t \rightarrow t + dt$

In both models each particle also stores

m : mass [MeV/c^2], Q : charge [e^+]

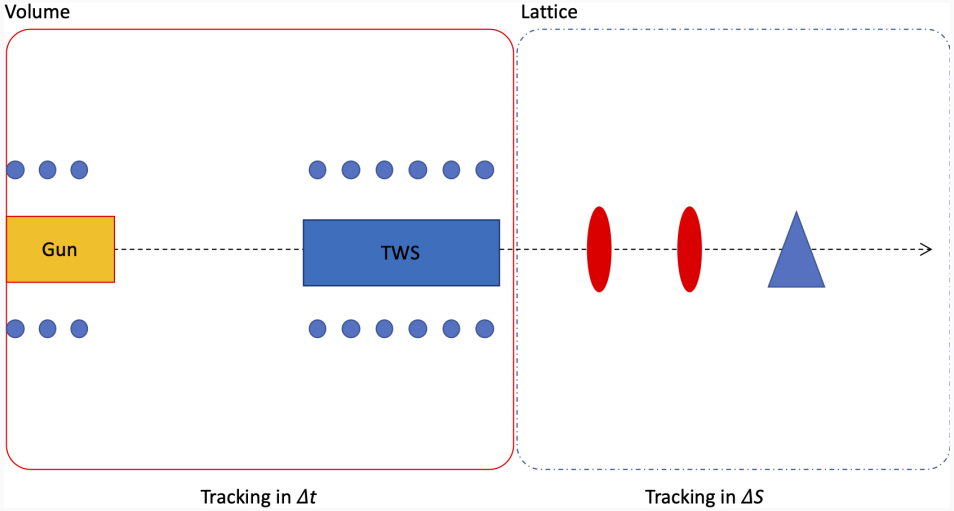
N : nb of particles / macroparticle, t_0 : creation time$^{(\star)}$ τ : lifetime **[NEW!]**

$^{(\star)}$ only for beams moving in time

RF-Track can simulate mixed-specie beams

Working with these environments

Lattice and Volume are meant to be used together:



(SC-dominated regimes)

Example:

```
V = Volume();
V.add (Element, X, Y, Z, roll, pitch, yaw, "reference");
```

- X, Y, Z: arbitrary position in the 3-D space
- roll, pitch, yaw: Euler angles
- reference point: "center", "entrance", "exit"

Tracking: Integration algorithms

- The default is: "**leapfrog**": fast, second-order accurate, "symplectic"

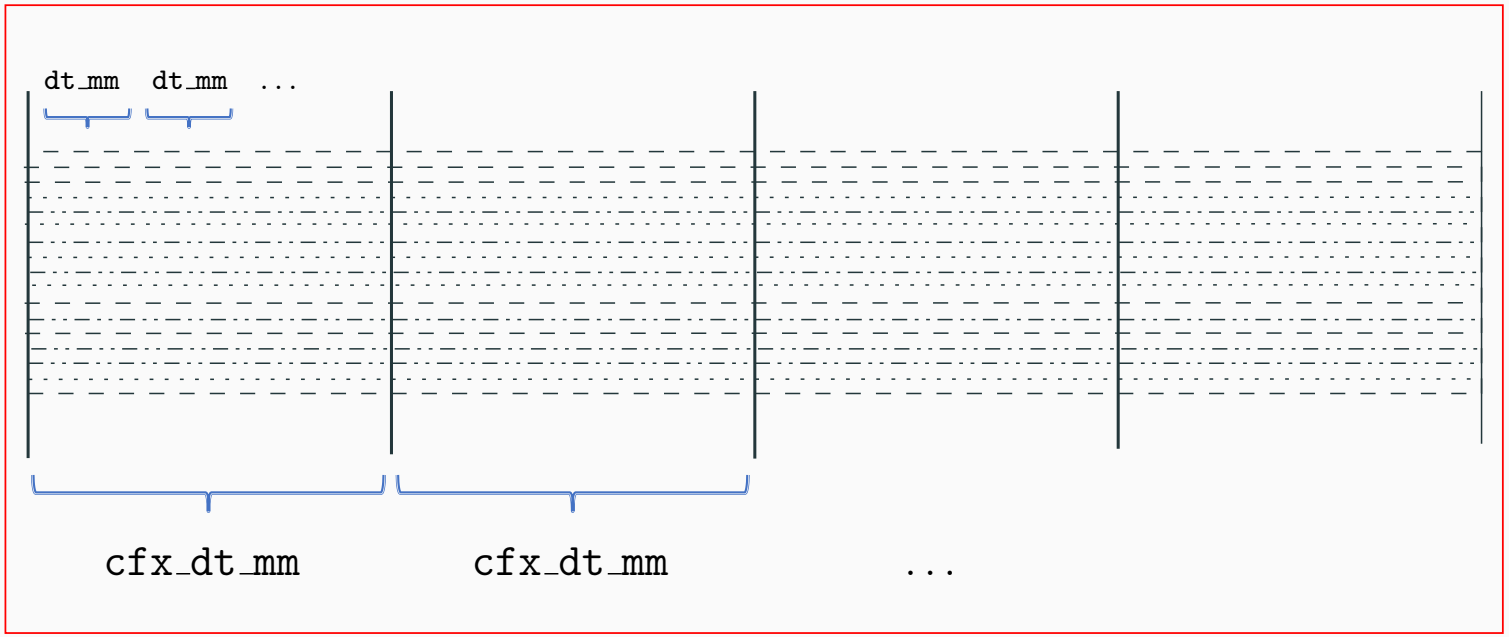
- Higher-order, adaptive algorithms provided by GSL:
  - ⋆"**rk2**" Runge-Kutta (2, 3)
  - ⋆"**rk4**" 4th order (classical) Runge-Kutta
  - ⋆"**rkf45**" Runge-Kutta-Fehlberg (4, 5)
  - ⋆"**rkck**" Runge-Kutta Cash-Karp (4, 5)
  - ⋆"**rk8pd**" Runge-Kutta Prince-Dormand (8, 9)
  - ⋆"**msadams**" multistep Adams in Nordsieck form; order varies dynamically between 1 and 12

- Analytic algorithm:
  - ⋆"**analytic**" integration of the equations of motion assuming a locally-constant EM field.

The beam can be tracked backward in time. Including collective effects.

Volume: Staggered integration

Beam is tracked in parallel between consecutive kicks of collective effects:



Volume

Staggered integration:

- Fine: Integration in 'dt_mm' is performed in parallel using high-order adaptive integration: 'rk2', 'rk4', 'rkf45', 'rk8pd', 'msadams', 'leapfrog'
- Coarse: Integration in 'cfx_dt_mm' is performed on the whole beam, using fixed-step-size leapfrog, and it's meant for collective effects

Beamline elements

1. **Standard** set of matrix-based symplectic elements:

- Sector Bends (standard matrix-based)
  - Quadrupole (standard matrix-based)
  - Drift (with an optional constant electric and magnetic fields, can be used to simulate e.g., rbends, or solenoids)

Beamline elements

1. **Standard** set of matrix-based symplectic elements:

- Sector Bends (standard matrix-based)
  - Quadrupole (standard matrix-based)
  - Drift (with an optional constant electric and magnetic fields, can be used to simulate e.g., rbends, or solenoids)

1. **Field maps** (see next slides)

Beamline elements

1. **Standard** set of matrix-based symplectic elements:

- Sector Bends (standard matrix-based)
  - Quadrupole (standard matrix-based)
  - Drift (with an optional constant electric and magnetic fields, can be used to simulate e.g., rbends, or solenoids)

1. **Field maps** (see next slides)

2. **Special elements**:
   - 3-D Analytic Coils
   - 3-D Analytic Solenoid
   - 3-D Analytic Standing wave and Traveling wave structures
   - Adiabatic matching devices, Toroidal Harmonics, LaserBeams (ICS)
   - Twiss table: tracks through an arbitrary lattice, given a table of Twiss parameters, phase advances, momentum compaction, 1^{st} and 2^{nd} order chromaticity

Field maps

RF-Track supports several types of field maps:

- 1-D (on-axis) field maps of oscillating traveling-wave electric fields
  - It uses the Maxwell's equations to reconstruct the 3-D fields: E_r, E_ϕ as well as B_r and B_ϕ in points off-axis
- 2-D fields maps: field on a plane and applies cylindrical symmetry
- 3-D field maps of oscillating electro-magnetic fields
  - It accepts 3-D meshes of complex numbers (see next slide for more details)
- StaticElectric and StaticMagnetic fields
  - Dedicated implementation to provide curl-free (electric) and divergence-free (magnetic) interpolation of the field map

Collective and single-particle effects

"Effects" can be attached to any element. An arbitrary number of effects can be attached.
Collective effects:

- Space-charge
  - Mirror charges at cathode
- Short-range wakefields:
  - Karl Bane's approximation
  - Arbitrary Spline
- Long-rage wakefields
  - Frequency, amplitude, and Q factor
- Self-consistent Beam-loading effect in TW structures (SW in progress)

Single-particle Effects:

- Incoherent Synchrotron Radiation (in any fields)
- Multipole kicks for magnetic-imperfection studies
- Multiple Coulomb Scattering (NEW!)

3-D Space-charge: P2P and PIC

RF-Track provides self-consistent 3-D solvers, with two optional methods:

1. Particle-2-particle:
   - computes the electromagnetic interaction between each pair of particles
   - numerically-stable summation of the forces (Kahan summation)
   - fully parallel: $O\left(N_{\text{particles}}^2\right)/N_{\text{cores}}$ operation

2. 3-D Particle-in-cell code: $\rightarrow$ fast
   - uses 3-D Integrated Green functions
   - computes E and B fields directly from ϕ and $\vec{A}$ (this ensures $\nabla \cdot \vec{B} = 0$)
   - can save E and B field maps on file, and use them for fast tracking
   - implements continuous beams
   - fully parallel: $O\left(N_{\text{mesh points}} \log N_{\text{mesh points}}\right)/N_{\text{cores}}$ operations

- No approximations such as "small velocities", or $\vec{B} \ll \vec{E}$, or rigid gaussian bunch, are made.
- It can simulate beam-beam forces
- New beam creation procedure: Simulation of photocathode
  - Quasi-random number generators added
- Strengthen the implementation of single-particle and collective effects
  - Multipole kicks for magnetic-imperfections studies

Beam creation

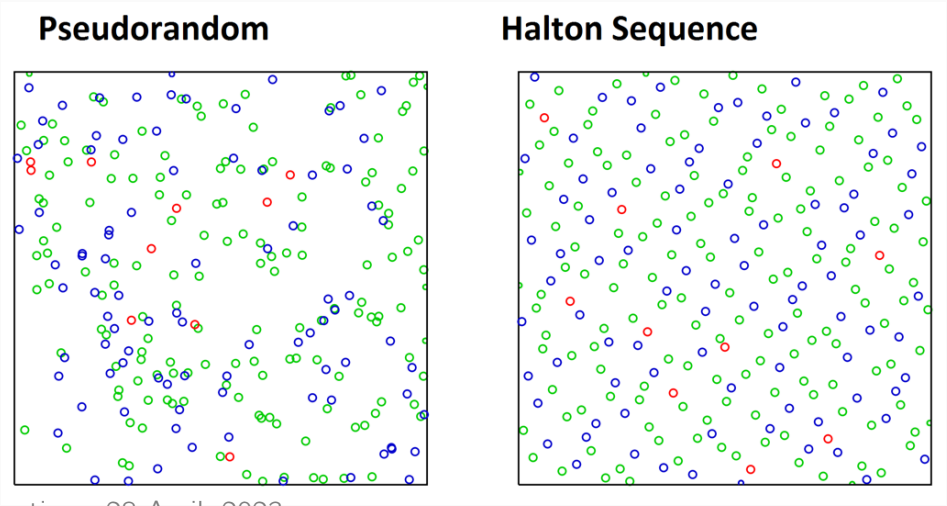
Credits: Avni Aksoy

- The emission process determines the beam properties the cathode
  - The emittance can't be smaller than the value one gets at the injector
  - Some 3D-distribution parameters (i.e., uncorrelated energy spread..)
- Usually, macro particles are generated with some physical constrains «randomly»
- In order to have good statistical agreement one needs to create «enough» number of macro-particle
  - We have «pseudo-random» numbers rather than «real» random numbers
  - if we do not have enough random number we might be struggling with «noises»
    - Large computing power
- Having «quasi»-random numbers let us to cover the domain of interest quickly

Beam creation

Credits: Avni Aksoy

- The emission process determines the beam properties the cathode
  - The emittance can't be smaller than the value one gets at the injector
  - Some 3D-distribution parameters (i.e., uncorrelated energy spread..)
- Usually, macro particles are generated with some physical constrains «randomly»
- In order to have good statistical agreement one needs to create «enough» number of macro-particle
  - We have «pseudo-random» numbers rather than «real» random numbers
  - if we do not have enough random number we might be struggling with «noises»
    - Large computing power
- Having «quasi»-random numbers let us to cover the domain of interest quickly



Beam creation/II

Example of photocathode simulation:

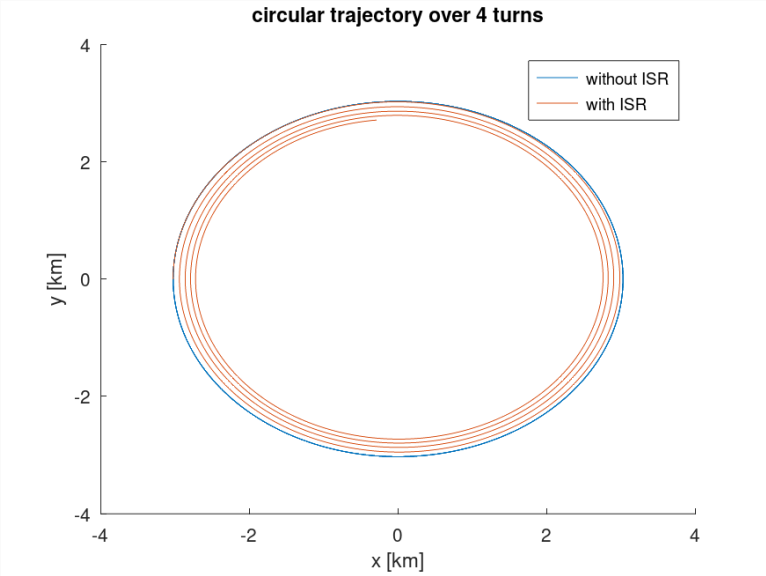
```
RF_Track;

G = Bunch6dT_Generator();
G.species = "electrons";          % species
G.cathode = true;                 % cathode, true or false
G.noise_reduc = true;             % noise reduction (quasi-random)
G.q_total = 0.285;                % nC bunch charge
G.ref_ekin = 0;                   % MeV energy of reference particle
G.ref_zpos = 0;                   % m position of reference particle
G.ref_clock = 0;                  % ns clock of reference particle
G.phi_eff = 3.5;                  % eV, effective work function
G.e_photon = 4.73;                % eV, photon energy for Fermi-Dirac distribution.
G.dist_x = 'g';                   % Specifies the transverse particle distribution in the horizontal direction.
G.dist_y = 'g';                   % Specifies the transverse particle distribution in the vertical direction.
G.dist_z = 'g';                   % Specifies the longitudinal particle distribution
G.sig_x = 1.600;                  % mm, rms bunch size in the horizontal direction. Also the vertical bunch size if Dist_x = radial.
G.sig_y = 1.600;                  % mm, rms bunch size in the vertical direction.
G.sig_t = 0.001;                  % ns, rms value of the emission time, i.e. the bunch length if generated from a cathode
G.c_sig_x = 5;                    % cuts off a Gaussian horizontal distribution
G.c_sig_y = 5;                    % cuts off a Gaussian vertical distribution
G.c_sig_t = 5;                    % cuts off a Gaussian longitudinal distribution
G.dist_pz = 'fd_300';             % Emission from cathode following a Fermi-Dirac dist. (for z)

%% Create bunch
Nparticles = 100000;
B = Bunch6dT(G, Nparticles);
M = B.get_phase_space();
```

New effect: Incoherent Synchrotron Radiation

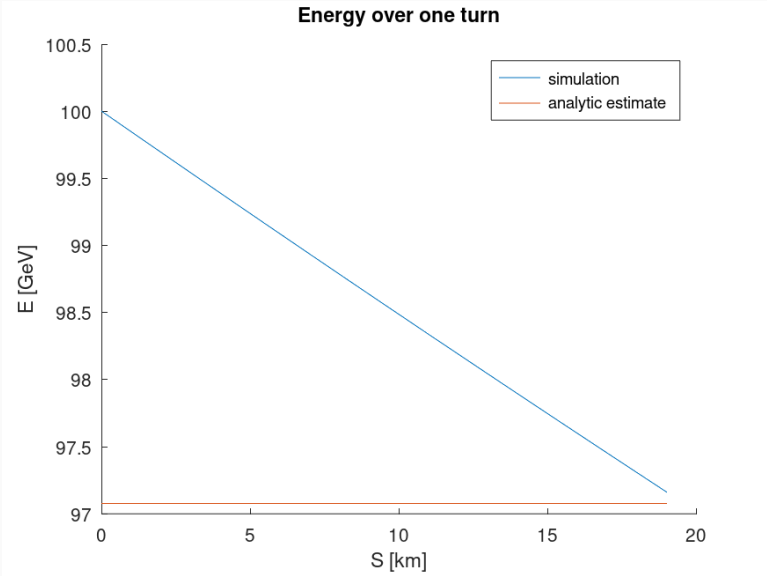
- Implements both average energy loss or stochastic emission of photos (optional)
- The radiation spectrum is based on the code written by Helmut Burkhardt (used in Geant4, PLACET, ...)
- It can be enabled in any elements
- It computes the effects of radiation due to any fields (even the accelerating one)

New effect: Incoherent Synchrotron Radiation

- Implements both average energy loss or stochastic emission of photos (optional)
- The radiation spectrum is based on the code written by Helmut Burkhardt (used in Geant4, PLACET, ...)
- It can be enabled in any elements
- It computes the effects of radiation due to any fields (even the accelerating one)



LEP-like parameters. Tracking of Bunch6dT in Volume()

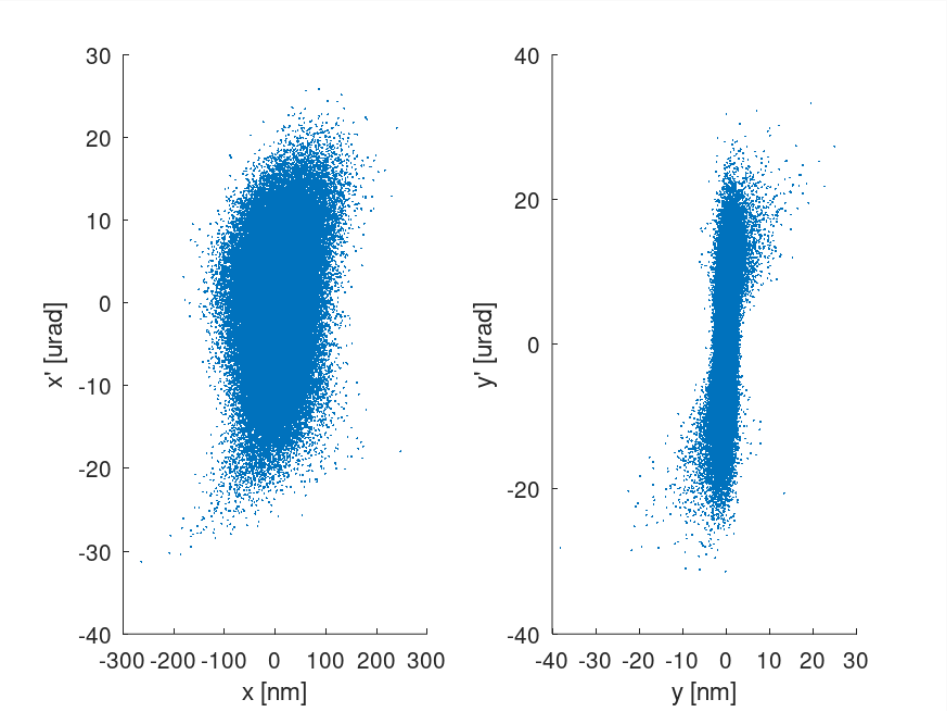


Energy profile over the first turn, compared with an analytic estimate of the final energy

Incoherent Synchrotron Radiation

CLIC: ISR in the Final-Focus System

Beam at the interaction point, with radiation effects



Beam size without radiation effects:

$\sigma_x = 40.6$ nm

$\sigma_y = 1.09$ nm

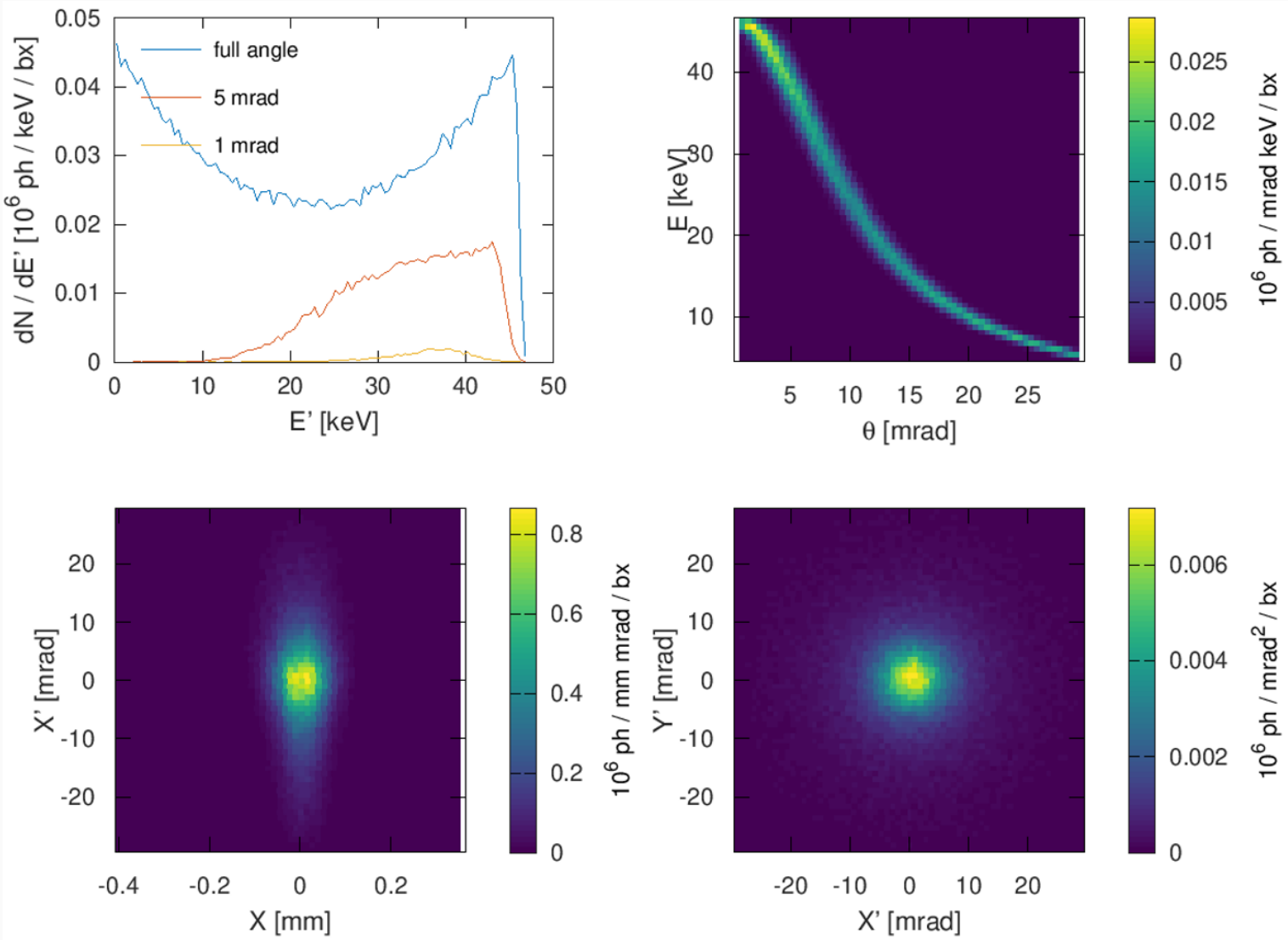
Beam size without radiation effects:

$\sigma_x = 42.0$ nm

$\sigma_y = 1.44$ nm

Special element: LaserBeam

To simulate Inverse-Compton Scattering: (ThomX example)



New element: Space-charge Field

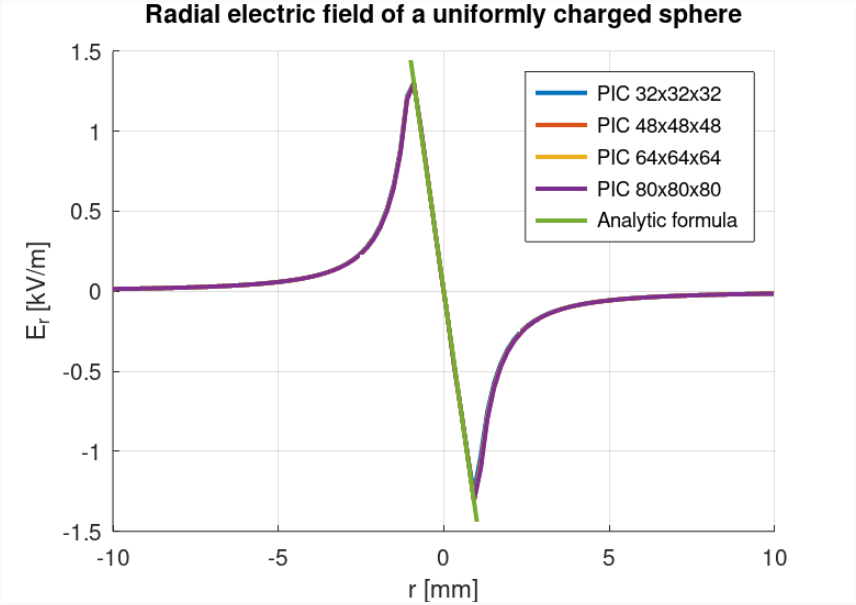
A new element was created: `SpaceCharge_Field()`

It contains the beam's excited electromagnetic field, extended to the whole space

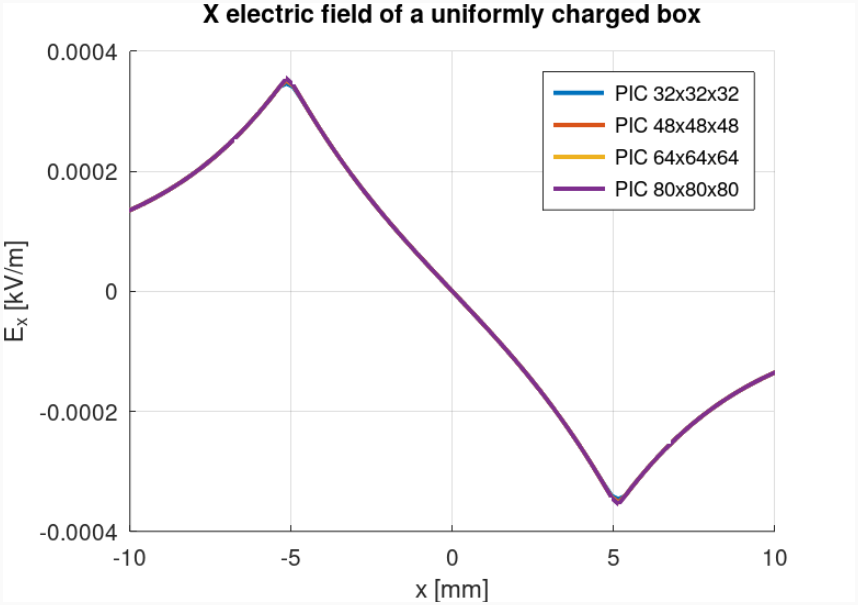
- For points inside the bunch:
  - It uses the fast space-charge routines and interpolates the field mesh
- For points outside the bunch:
  - It performs a summation of the fields due to each mesh cell (can be improved)
  - Both electric and magnetic contributions are considered
  - It uses 3D "Integrated Coulomb fields" (each mesh cell is considered as uniformly filled with charge)
  - Retardation is considered (with some approximations)
  - 2D Integrated Coulomb fields are used for ultra-relativistic energies

Full overlap of the fields is possible. This allows the use of one `SpaceCharge_Field()` for each different species. Each species with its own appropriate mesh.

Space-charge Field



SpaceCharge calculation. Fields are computed inside the bounding box. It matches well the analytic estimate.



SpaceCharge_Field: the field exists also outside of the charge distribution.

Update: LEMMA Scheme with Super-positron Bunch

Work developed with: Manuela Boscolo, Mario Antonelli, Frank Zimmermann, and John Farmer in April/May 2022

Plasma: Argon ions

Nominal parameters

- Plasma length = 100 m
- Positron bunch length = 5 m (longitudinally uniform)

Simulation parameters

- Plasma length = 10 mm
- Positron bunch length = 5 mm (longitudinally uniform)

Integration step = 1 $\mu\mathrm{m}/c$

Positrons = 10'000 macro particles

Ions / electrons = 100'000 macro particles

e^+, e^-, Ag, μ^+, μ^- are tracked simultaneously.
Each species excites an electromagnetic field and feels the electromagnetic fields excited by the other species as well as its own self field.

Simulation setup

Parameter	Value	Remark
Plasma length	10 mm	Nominal: 100 m
Plasma density	$10^{23}\ \mathrm{m}^{-3}$	
Plasma distribution	$1 \times 1 \times 10\ \mathrm{mm}^3$	uniform cylinder $R_x \times R_y \times L$
Positron population	10^{14}	core density $10^{27}\ \mathrm{m}^{-3}$
Positron bunch length	5 mm	Nominal: 5 m
Positron longitudinal profile	uniform	
Positron transverse profile	Gaussian	
Positron geometric emittance	$135\ \mathrm{pm}$	$11.8\ \mathrm{mm} \cdot \mathrm{mrad}$ normalised
$\beta^\star$ at entrance	7 mm	
Plasma mesh	32x32x32	
Positron mesh	8x8x64	
Muon mesh	8x8x64	
Plasma macro-particles	100'000	100'000 electrons, 100'000 ions
Positron macro-particles	10'000	

Main simulation script

```
for i = 1:N_steps
    %% compute fields
    if do_EE || do_PE || do_ME;  SC_ele = SpaceCharge_Field (B0e, 32, 32, 32); end
    if do_EP || do_PP || do_MP;  SC_pos = SpaceCharge_Field (B0p, 8, 8, 64); end
    if do_EM || do_PM || do_MM;  SC_muo = SpaceCharge_Field (B0m, 8, 8, 64); end

    %% compute forces
    if do_EE;  Fee = compute_force(B0e, SC_ele); end
    if do_EP;  Fep = compute_force(B0e, SC_pos); end
    if do_EM;  Fem = compute_force(B0e, SC_muo); end
    if do_PP;  Fpp = compute_force(B0p, SC_pos); end
    if do_PE;  Fpe = compute_force(B0p, SC_ele); end
    if do_PM;  Fpm = compute_force(B0p, SC_muo); end
    if do_MP;  Fmp = compute_force(B0m, SC_pos); end
    if do_ME;  Fme = compute_force(B0m, SC_ele); end
    if do_MM;  Fmm = compute_force(B0m, SC_muo); end

    %% apply forces
    B0e.apply_force(Fee + Fep + Fem, dt_mm);
    B0p.apply_force(Fpe + Fpp + Fpm, dt_mm);
    B0m.apply_force(Fme + Fmp + Fmm, dt_mm);

    %% generate muons
    M0  = B0e.get_phase_space ('%X %Px %Y %Py %S %Pz %m %Q %N'); % e+i
    M0e = M0(M0(:,8) == -1,:);
    M0p = B0p.get_phase_space ('%X %Px %Y %Py %S %Pz %m %Q %N');

    M0m = [ M0m ; generate_muons(M0p, M0e, xs_enhancement * dt_mm) ];
    B0m = Bunch6dT(M0m);
end
```

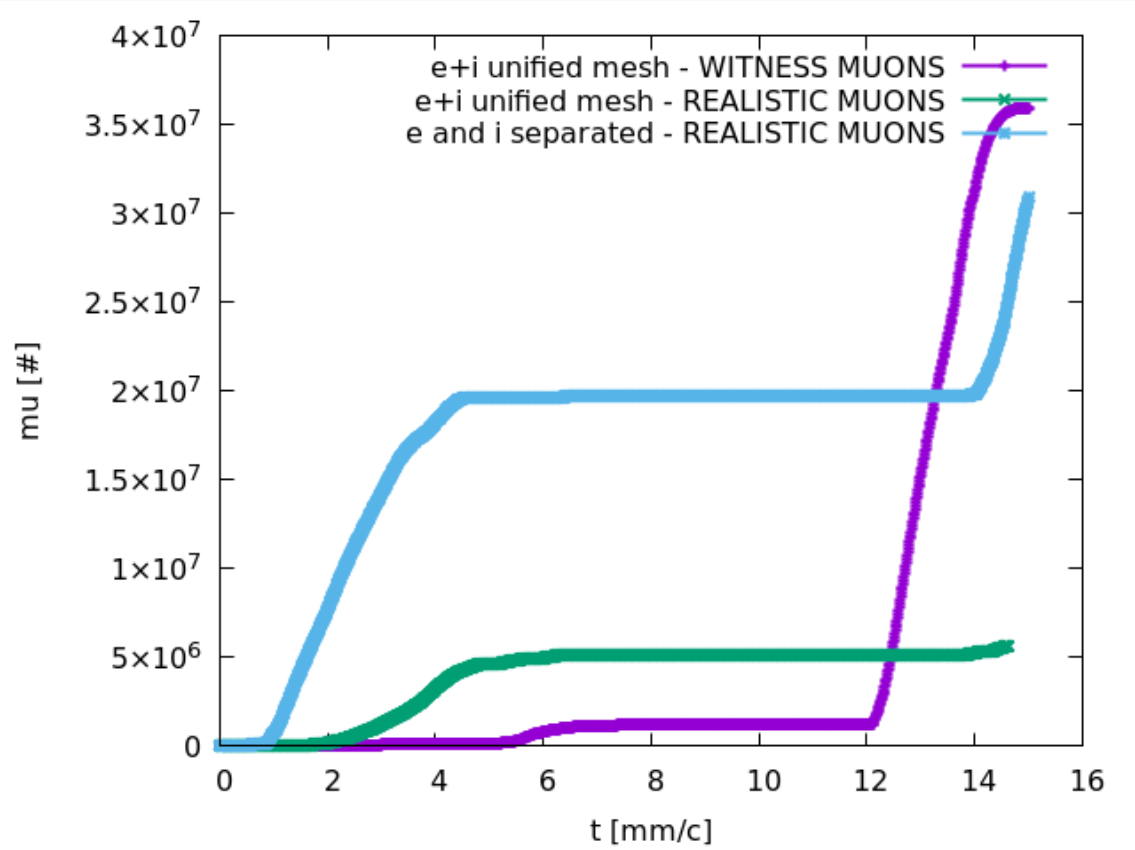
Main simulation script

```
for i = 1:N_steps
    %% compute fields
    if do_EE || do_PE || do_ME;  SC_ele = SpaceCharge_Field (B0e, 32, 32, 32); end
    if do_EP || do_PP || do_MP;  SC_pos = SpaceCharge_Field (B0p, 8, 8, 64); end
    if do_EM || do_PM || do_MM;  SC_muo = SpaceCharge_Field (B0m, 8, 8, 64); end

    %% compute forces
    if do_EE;  Fee = compute_force(B0e, SC_ele); end
    if do_EP;  Fep = compute_force(B0e, SC_pos); end
    if do_EM;  Fem = compute_force(B0e, SC_muo); end
    if do_PP;  Fpp = compute_force(B0p, SC_pos); end
    if do_PE;  Fpe = compute_force(B0p, SC_ele); end
    if do_PM;  Fpm = compute_force(B0p, SC_muo); end
    if do_MP;  Fmp = compute_force(B0m, SC_pos); end
    if do_ME;  Fme = compute_force(B0m, SC_ele); end
    if do_MM;  Fmm = compute_force(B0m, SC_muo); end

    %% apply forces
    B0e.apply_force(Fee + Fep + Fem, dt_mm);
    B0p.apply_force(Fpe + Fpp + Fpm, dt_mm);
    B0m.apply_force(Fme + Fmp + Fmm, dt_mm);

    %% generate muons
    M0  = B0e.get_phase_space ('%X %Px %Y %Py %S %Pz %m %Q %N'); % e+i
    M0e = M0(M0(:,8) == -1,:);
    M0p = B0p.get_phase_space ('%X %Px %Y %Py %S %Pz %m %Q %N');

    M0m = [ M0m ; generate_muons(M0p, M0e, xs_enhancement * dt_mm) ];
    B0m = Bunch6dT(M0m);
end
```

Muon creation as a result of $e^+ - e^-$ collisions was also implemented, as an Octave function

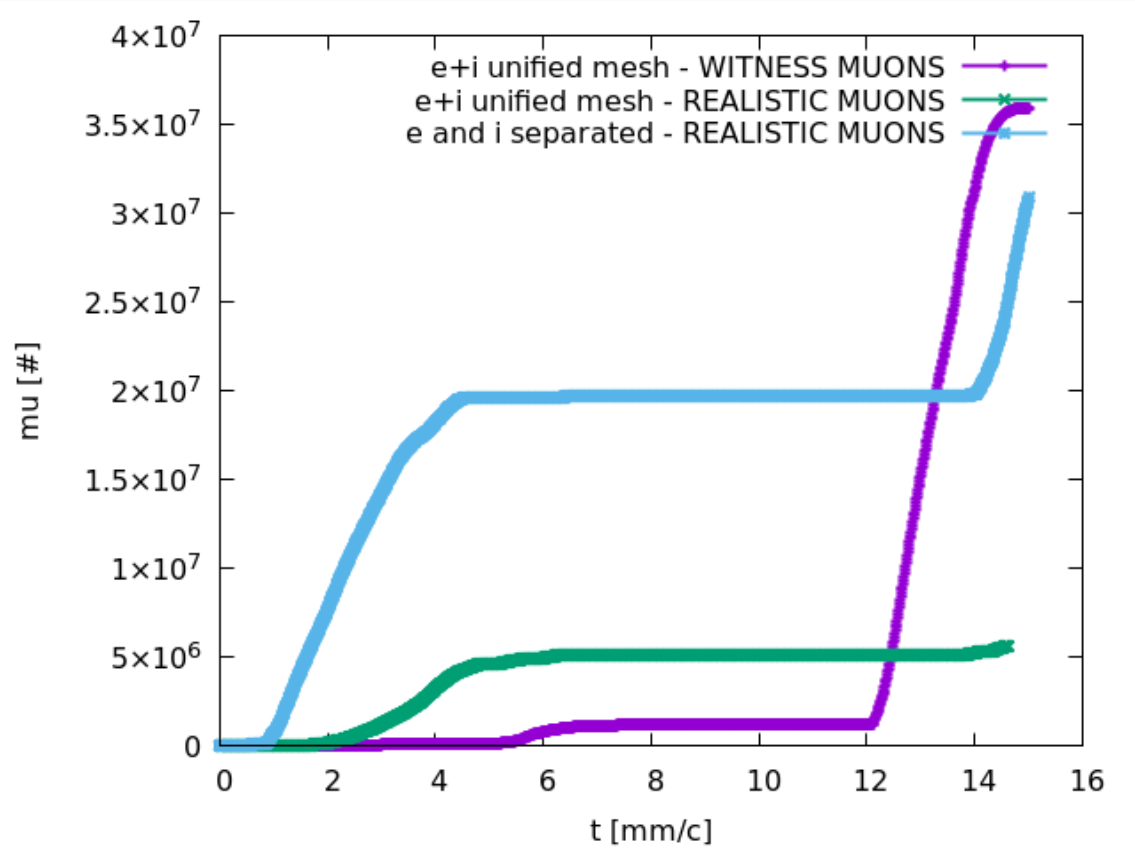
Simulation results

In May 2022 we showed that muon build-up occurs



Simulation results

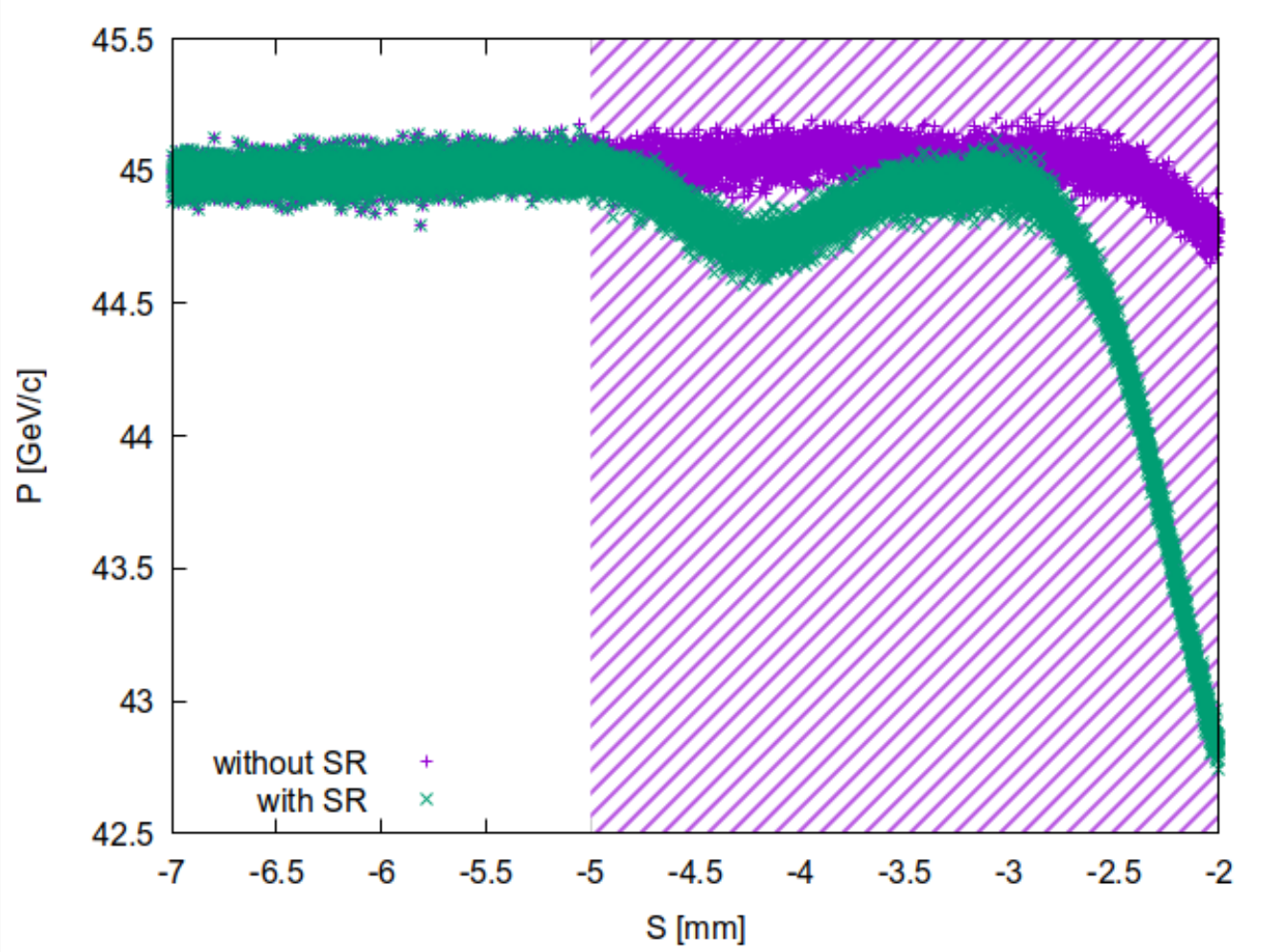
In May 2022 we showed that muon build-up occurs



At the time, Daniel pointed out that effects of incoherent synchrotron radiation (a.k.a. betatron radiation, in the PLASMA wakefield accelerators' jargon) and Bhabha scattering would rapidly consume all the positrons's energy.

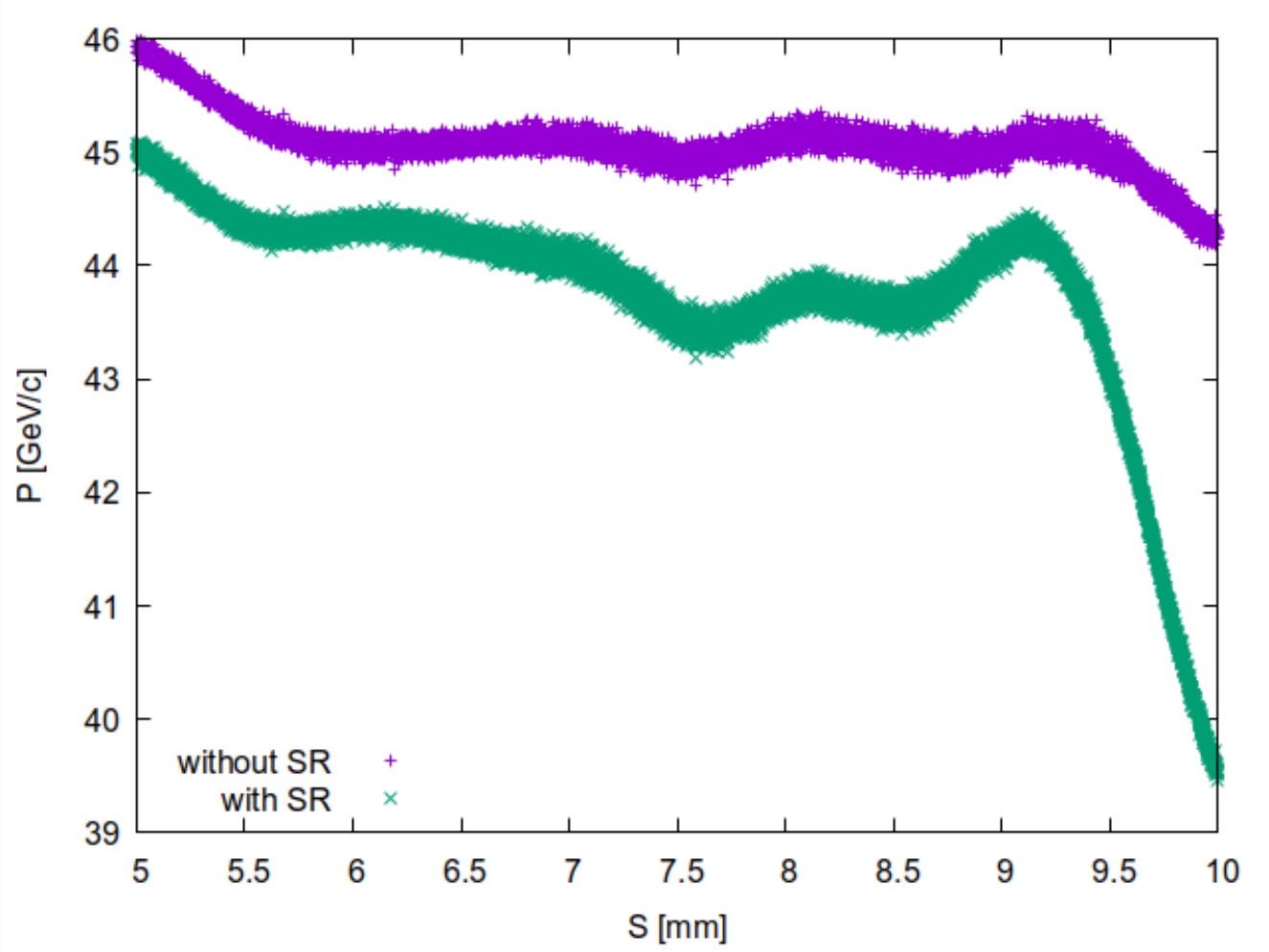
Simulation results

As the positron bunch enters the plasma (plasma starts at $S = -5$ mm)



Simulation results

As the positron bunch exits the plasma (plasma ends at $S = 5$ mm)



Indeed, the positron beam loses a considerable amount of energy...

New element: Absorber

It's basically a drift that implements **Multiple Coulomb Scattering**:

```
Absorber(length, radiation_length, Z, A, density, mean_excitation_energy);
Absorber(length, material_name );
```

Pre-defined materials include: 'air', 'water', 'beryllium', 'lithium', 'liquid_hydrogen', 'tungsten'.

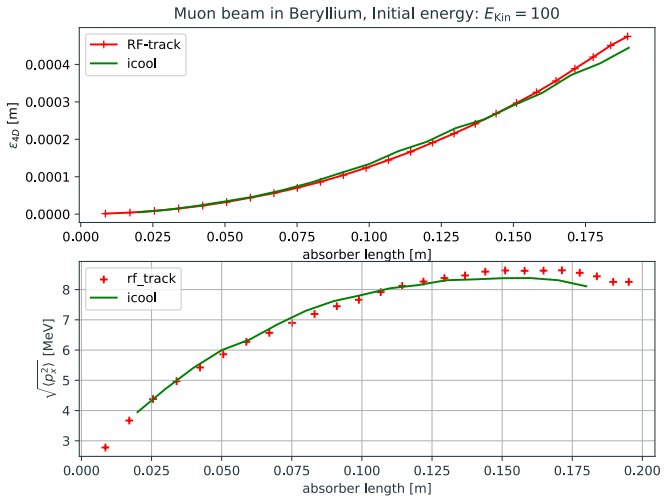
The target can have circular or rectangular shape.

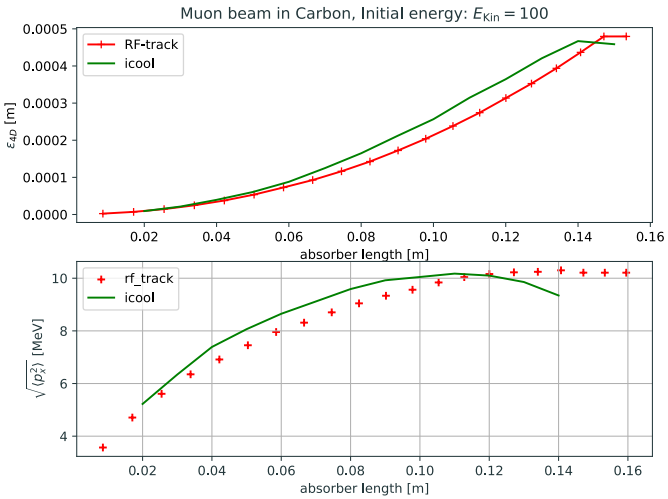
NOTICE:

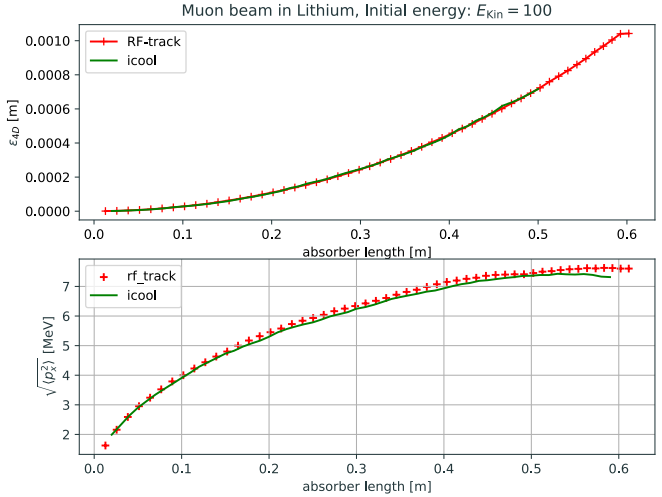
- As any other collective effect in RF-Track, **Multiple Coulomb Scattering** can be added to any element.

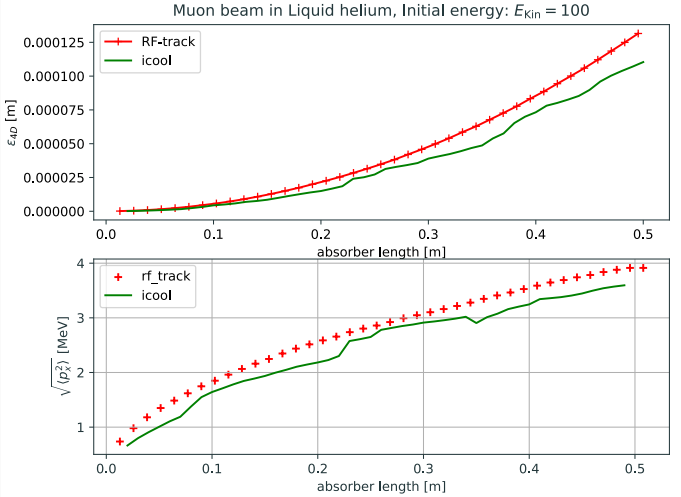
Absorber: Benchmark against ICOOL

Credits: Bernd Stechauner



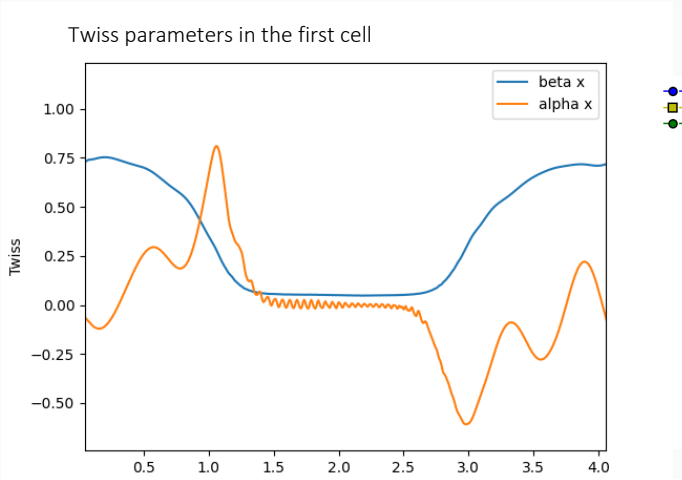


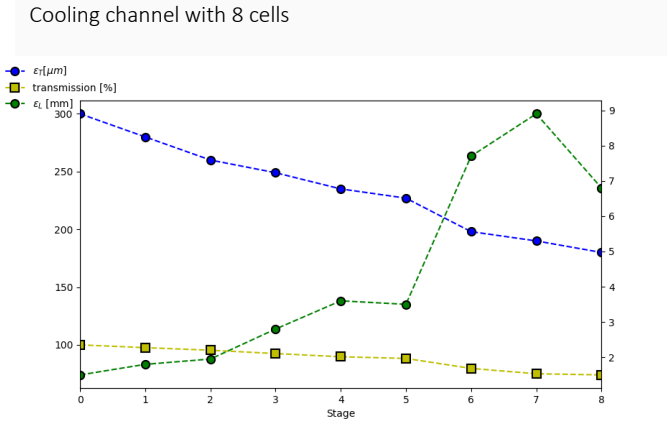


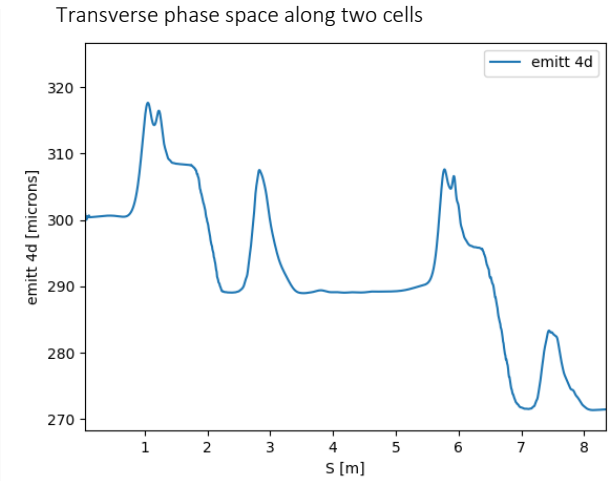


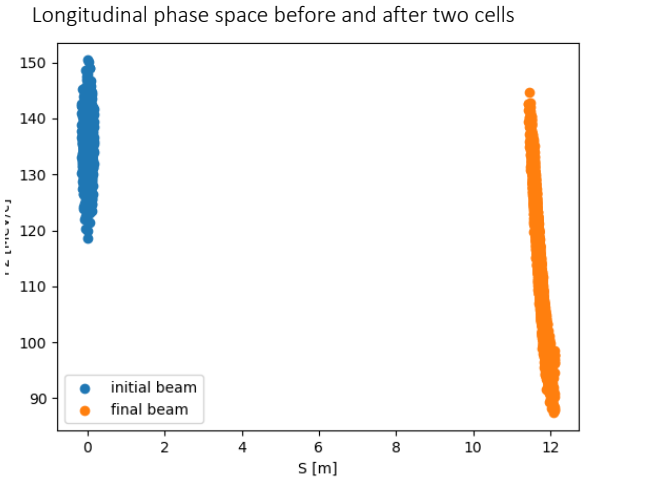
Absorber: Examples of muon cooling channel optimisation

Credits: Elena Fol









This simulation includes: a constant solenoid field, realistic 3D solenoid, several standing-wave structures, and the absorber, simultaneously together and overlapping.

New feature: Orbit Correction

Usage example

```
sigmaX    = 0.100; % mm rms
sigmaXP   = 0.100; % mrad rms
sigmaROLL = 0.100; % mrad rms
sigmaBPMS = 0.010; % mm, bpm resolution

________
%% Main

% Lin is a linac

Res = Lin.get_response_matrix(P0);

Lin.set_bpm_resolution(sigmaBPMS);

nmachines = 100;
for machine = 1:nmachines

    Lin.scatter_elements('quadrupole', sigmaX, sigmaX, 0.0, sigmaROLL, sigmaXP, sigmaXP, 'center');
    Lin.scatter_elements('rf_element', sigmaX, sigmaX, 0.0, sigmaROLL, sigmaXP, sigmaXP, 'center');
    Lin.scatter_elements('bpm', sigmaX, sigmaX, 0.0, sigmaROLL, sigmaXP, sigmaXP, 'center');

    Lin.reset_correctors();

    ________________________
    %% uncorrected machine
    B1 = Lin.track(B0);
    T_Unc = Lin.get_transport_table('%S %emitt_x %emitt_y');

    ________________
    %% correct machine
    B1 = Lin.orbit_correction(Res, B0); % return the corrected bunch
    T_Cor = Lin.get_transport_table('%S %emitt_x %emitt_y');

    T_Sum += T_Cor;
end

T_Sum /= nmachines;
```

Orbit Correction in FCC-ee injector common linac

Credits: Simona Bettoni (PSI)

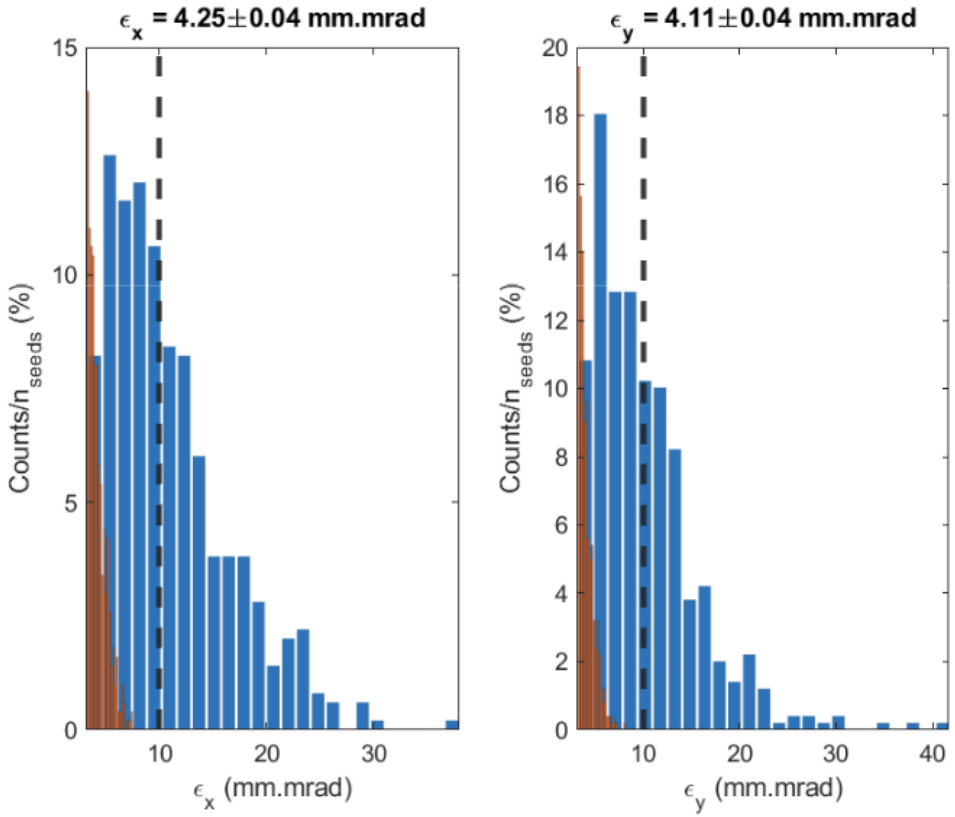


Figure 14: Common linac: emittance at the end of the common linac due to Gaussian randomly misaligned quadrupoles (50 µm), rf structures (50 µm), and BPM (30 µm). Gaussian randomly distributed. The distribution is determined over 500 seeds. In this case the mean and rms refer to the case of the steered orbit.

Orbit Correction in FCC-ee injector linacs

Credits: Simona Bettoni (PSI). We found a bug in ELEGANT.

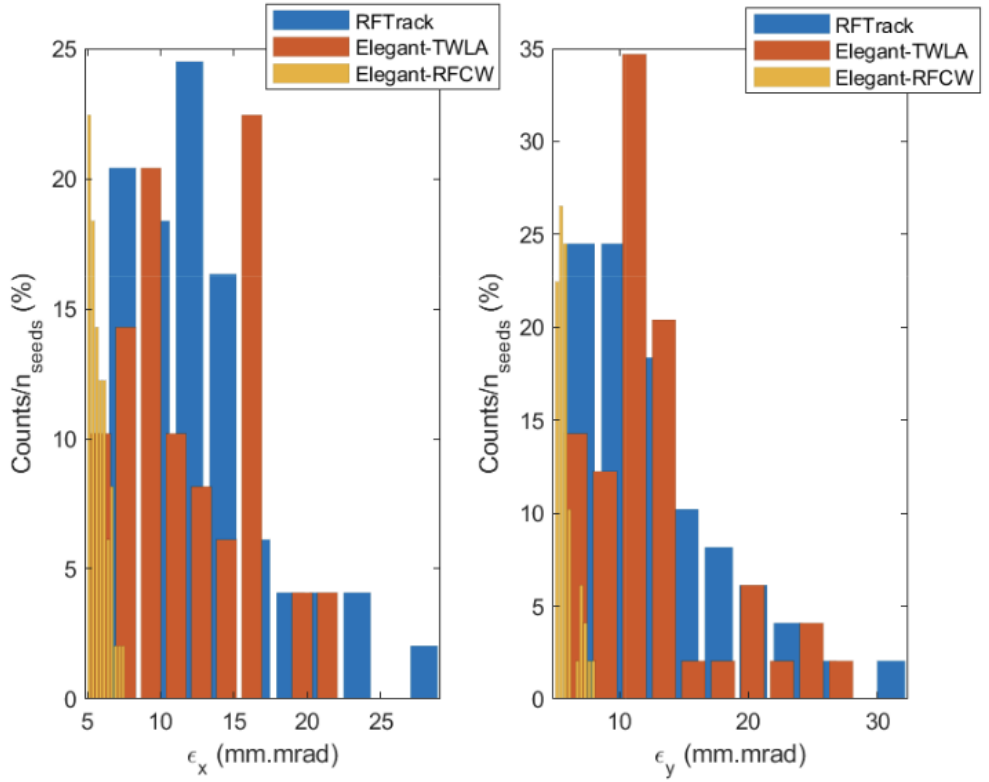


Figure 9: Comparison Elegant versus RF-Track using the different elements available to simulate rf structures in Elegant and RF-Track.

Now RF-Track is the reference code for the FCC-ee injectors.

Next steps...

Collective effects:

- Complete benchmark of Absorber (TOPAS, Geant4, FLUKA)
- Finalise Beam-Loading effects (Javier Olivares)
- Coherent Synchrotron radiation with 3D retardation effects
- Intra-beam Scattering as Monte Carlo scattering

New elements:

- Taylor Maps in Lattice

Structural changes:

- Just implemented the possibility to nest Lattices inside Lattice
- Allow insertion of Volume in Lattice directly

Summary and How to get it

RF-Track:

- A new code: minimalistic, parallel, fast
- Friendly and flexible, it uses Octave and Python as user interfaces
- Currently used for: FCC-ee, DEFT, Muons cooling, positron sources, RFQ, ...

In-progress documentation:

- `https://zenodo.org/record/4580369`

Pre-compiled binaries and more up-to-date documentation are available here:

- `https://gitlab.cern.ch/rf-track`

Acknowledgements: many thanks to all contributors.