RF-Track: a minimalistic multipurpose tracking code featuring space-charge

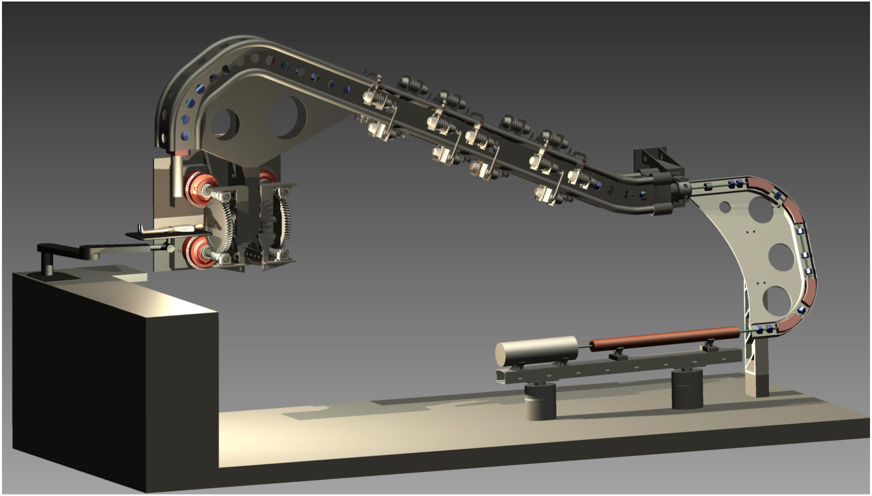
Andrea Latina
(BE-ABP-LAT)

ABP-CWG - 24 May 2017

Table of Contents

Motivations for RF-Track: The TULIP Project

High-gradient proton linac for medical applications (hadron therapy)



Motivations for RF-Track

- TULIP Project: the design of a linac based on a new high-gradient (50 MV/m) S-band backward travelling-wave accelerating structure, with initial $\beta = 0.38$, dictated the need to develop a new code capable of tracking particles through such linac
- We needed 6d tracking in 3d EM field maps of bwTW structures
- We needed the possibility to maximise the transmission tuning **any** parameters: RF input power, quads strengths, quads positions, input distribution, etc.
- We needed to track protons as well as carbon ions
- We needed something flexible and fast
- We decided to develop RF-Track

RF-Track Highlights

- Can handle $\mathbb{C}$omplex 3d field maps of Electric + Magnetic fields: static fields, as well as field maps of RF backward/forward travelling waves
- It's fully relativistic
  - no approximations like $\beta \ll 1$ or $\gamma \gg 1$
  - successfully tested with: electrons, positrons, protons, antiprotons, ions, at various energies
- Can track mixed-species beams
- Implements high-order integration algorithms
- Implements space-charge
- Implements electron cooling [new!]
- It is indeed flexible and fast

RF-Track Internals

- RF-Track is a C++ library:
  - fast, optimised code
  - modern C++11, natively parallel
  - great care for numerical stability
- It works on Linux, Mac, and Windows (cygwin / Windows 10)
- Physics-oriented: it's a minimalistic code, relies on two robust and well known open-source libraries for *"all the rest"*
  - GSL, "Gnu Scientific Library", provides a wide range of mathematical routines such as random number generators, ODE integrators, linear algebra e.g. BLAS, and more
  - FFTW, "Fastest Fourier Transform in the West", probably the fastest library to compute discrete Fourier transforms ever made
- About 12,000 lines of code

Overview: user interface

RF-Track is a library, loadable from

- Octave, *a high-level language for numerical computations, mostly compatible with Matlab, open-source*
- Python, *general-purpose, high-level programming language, open-source*

Both languages offer powerful toolboxes for numerical experimentation: multidimensional optimisations, fitting routines, data analysis, control tools, ...

Example (octave interface)

```
% load RF-Track
RF_Track;

% setup simulation
TL = setup_transferline;
B0 = setup_beam;

% track
B1 = TL.track(B0);

% inquire the phase space
T1 = B1.get_phase_space("%x %xp %y %yp");

% plot
plot(T1(:,1), T1(:,2), "*");
xlabel("x [mm]");
ylabel("x' [mrad]");
```

Overview: example of beam line definition

- Example of beam line definition:

```
% quad integrated strength S = Q [e] * G [T/m] * L [m] * c [m/s] = [MeV/m]
QF = Quadrupole(L_quad, S_QF);
QD = Quadrupole(L_quad, S_QD);
DD = Drift(L_drift);

QF.set_aperture(0.1, 0.1, "circular");
QD.set_aperture(0.1, 0.1, "circular");
DD.set_aperture(0.1, 0.1, "rectangular");

FODO = Lattice(); % 1 cell
FODO.append(QF);
FODO.append(DD);
FODO.append(QD);
FODO.append(DD);

L = Lattice(); % 3 cells
F.append(FODO);
F.append(FODO);
F.append(FODO);
```

Overview: input / output

- Input / output:
  - Input / output through Octave: ASCII files, binary files, HDF5
  - RF-Track can save beam data in DST format (PlotWin)
  - On-the-fly compression / decompression of files through Octave (e.g. field maps)

- Retrieving particles phase space can be done with great flexibility:
  - PLACET style

    ```
    T1 = B1.get_phase_space("%E %x %y %dt %xp %yp");
    ```

  - MAD-X style

    ```
    T1 = B1.get_phase_space("%x %px %y %py %Z %pt");
    ```

  - TRANSPORT style

    ```
    T1 = B1.get_phase_space("%x %xp %y %yp %dt %d");
    ```

Overview: inquiring the phase space

One can retrieve the phase space with great flexibility: e.g.

```
P1 = B1.get_phase_space(%x %Px %y %Py %deg@750 %K);
```

%x	horiz. position at S	mm	%X	horiz. position at $t=t_0$	mm
%y	vert. position at S	mm	%Y	vert. position at $t=t_0$	mm
%xp	horiz. angle	mrad	%t	proper time	mm/c
%yp	vert. angle	mrad	%dt	delay = $t - t_0$	mm/c
%Vx	velocity	c	%z	$S/beta_0 - c.t = c(t_0 - t)$	mm
%Vy	velocity	c	%Z	–%dt * %Vz	mm
%Vz	velocity	c	%S	S + %Z	m
%Px	momentum	MeV/c	%deg@f	degrees *@freq* [MHz]	deg
%Py	momentum	MeV/c	%d	relative momentum	per mille
%Pz	momentum	MeV/c	%pt	$(\%E - E_0) / P_0 c$	per mille
%px	$\%Px/P_0$	mrad	%P	total momentum	MeV/c
%py	$\%Py/P_0$	mrad	%E	total energy	MeV
%pz	$\%Pz/P_0$	mrad	%K	kinetic energy	MeV

Overview: tracking global beam properties

After tracking over a lattice "L", one can retrieve the average beam properties:

```
T1 = L.get_transport_table("%S %beta_x %beta_y %mean_K %N");
```

%sigma_x|xp|y|yp|t|P	Std(#)	various
%sigma_xxp|yyp|tP	Cov(#,#)	various
%mean_x|xp|y|yp|t|P|K|E	Mean(#)	various
%alpha_x|y|z	Twiss	1
%beta_x|y|z	Twiss	see below
%rmax	envelope	mm
%emitt_x|y|z	RMS emittance	see below
%S	position	m
%N	Transmission	percent

β_x [m/rad]	$\epsilon_x = \epsilon_{x,\text{ geom}} \beta_{\text{rel}} \gamma_{\text{rel}}$ [mm.mrad]	$\epsilon_{x,\text{ geom}} = \sigma_x \sigma_{x'}$	$\sigma_x = \sqrt{\epsilon_{x,\text{ geom}} \beta_x}$ [mm]
β_y [m/rad]	$\epsilon_y = \epsilon_{y,\text{ geom}} \beta_{\text{rel}} \gamma_{\text{rel}}$ [mm.mrad]	$\epsilon_{y,\text{ geom}} = \sigma_y \sigma_{y'}$	$\sigma_y = \sqrt{\epsilon_{y,\text{ geom}} \beta_y}$ [mm]
β_z [m]	$\epsilon_z = \epsilon_{z,\text{ geom}} \beta_{\text{rel}} \gamma_{\text{rel}}$ [mm.permil]	$\epsilon_{z,\text{ geom}} = \sigma_Z \sigma_\delta$	$\sigma_z = \sqrt{\epsilon_{z,\text{ geom}} \beta_z}$ [mm]

Tracking: RF-Track implements two beam models

1. Beam moving in space:
   - 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:

$$S \rightarrow S + dS$$

2. Beam moving in time:
   - All particles are taken at same time t
   - Each particle's phase space is

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

   - It can handle particles with $P_z < 0$ or even $P_z = 0$
   - Integrates the equations of motion in dt:

$$t \rightarrow t + dt$$

- Each particle also stores

$$m:\ \text{mass [MeV/c}^2\text{]},\quad Q:\ \text{charge [}e^+\text{]}$$

$$N:\ \text{nb of particles / macroparticle},\quad t_0:\ \text{creation time}^{(\star)}$$

so that RF-Track can simulate mixed-species beams, cathodes, and field emission.

$^{(\star)}$ type 2 only

Tracking: Integration algorithms

By default, RF-Track uses:

- "leapfrog" integration algorithm: fast, second-order, symplectic

In some cases leapfrog might be not accurate enough. RF-Track offers the following algorithms as an alternative:

- GSL explicit algorithms:
  - ⋆"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
- GSL implicit algorithms:
  - ⋆"rk1imp", "rk2imp", "rk4imp" implicit Runge-Kutta
  - ⋆"bsimp" Bulirsch-Stoer method of Bader and Deuflhard
  - ⋆"msbdf" multistep backward differentiation formula (BDF) method in Nordsieck form
- Exact algorithm:
  - ⋆"analytic" integration of the equations of motion in a locally constant EM field

Example:

```
L = Lattice();
L.append(RFQ);
L.set_odeint_algorithm("msadams");

B1 = L.track(B0, 1.0); % tracks in time, using integration step dt = 1 mm/c
```

Beam line elements

Minimal set of elements:

- RF / Static Field map
- Quadrupole
- Drift

- ...Field maps and drifts can embed a static B field

- Each element:
  - can be tracked in several steps, to capture phase space evolution along the element
  - can have an aperture, which is checked at <u>any</u> integration step

  - If a particle hits the aperture, it is flagged as lost
    - User can retrieve its full phase space components, as well as its 3d position and the time at which is lost

- The user can identify what particles are lost in the initial distribution

Elements: RF Field map

- Complex EM maps of RF fields
  - forward traveling / backward traveling / static fields
  - tricubic interpolation
- Accept quarter / half field maps
  - automatic mirroring of the fields
  - accepts cartesian and cylindrical maps
- Can change dynamically input power
  - Provide P_{map}, set P_{actual}
- Not-a-Numbers are considered as walls
  - allows to precisely track losses in the 3d volume
- One can retrieve the fields at any point:

  e.g.

```
[E,B] = RFQ.get_field(x,y,z,t);
```

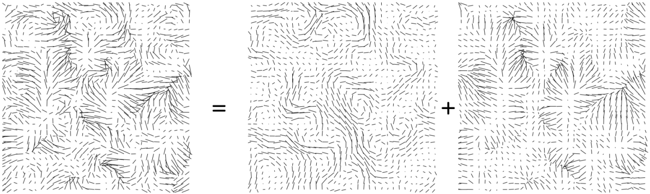
Example:

```
load 'field.dat.gz';

RFQ = RF_Field( ...
    field.Ex, ... % Efield [V/m]
    field.Ey, ...
    field.Ez, ...
    field.Bx, ... % Bfield [T]
    field.By, ...
    field.Bz, ...
    field.xa(1), ... % x0,y0 [m]
    field.ya(1), ...
    field.hx, ... % mesh size [m]
    field.hy, ...
    field.hz, ...
    field.za(end), ... % length [m]
    field.frequency, ... % [Hz]
    field.direction, ... % +1, -1, 0
    field.P_map, ... % [W]
    field.P_actual);
```

Elements: Static field map [in progress]

- Special care is given to static 3d field maps of magentic and electric fields
- Any field map can be seen as the sum of a mass-conserving field (divergence-free) and a gradient field (curl-free):



- In the paper:

> Brackbill, J.U. and Barnes, D.C.,*"The effect of nonzero $\nabla \cdot B$ on the numerical solution of the magneto-hydrodynamic equations"*, Journal of Computational Physics, Volume: 35 Issue: 3 Pages: 426-430, 1980

  the authors demonstrate that $\nabla \cdot \mathbf{B} \neq 0$ results in numerical errors in the solution of the equation of motion that appear as a force parallel to the field.

- RF-Track performs such a decomposition and zeroes the unwanted mass-source terms

Physics: Space-charge interaction

RF-Track solves the basic differential laws of magneto and electro-statics for any distribution of particles. Full 3d solver, with two optional methods:

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

2. cloud-in-cell: $O\left(n_{\text{particles}} \cdot n_{\text{grid}} \cdot \log n_{\text{grid}} \,/ n_{\text{cpus}}\right)$ computations $\rightarrow$ much faster

$$\vec{A}(x) = \frac{\mu_0}{4\pi} \iiint \frac{\vec{j}(x')}{|x - x'|} d^3x' \quad \Rightarrow \quad \vec{B} = \nabla \wedge \vec{A}$$
$$\phi(x) = \frac{1}{4\pi\epsilon_0} \iiint \frac{\rho(x')}{|x - x'|} d^3x' \quad \Rightarrow \quad \vec{E} = \nabla \phi$$

   - uses 3d Integrated Retarded Green functions
   - computes $\vec{E}$ and $\vec{B}$ fields directly from ϕ and $\vec{A}$ (this ensures $\nabla \cdot \vec{B} = 0$)
   - can save $\vec{E}$ and $\vec{B}$ field maps on disk, and use them for fast tracking
   - implements continuous beams
   - fully parallel

- No approximations such as "small velocities", or $\vec{B} \ll \vec{E}$, or gaussian bunch, are made.
- Can simulate beam-beam forces

Physics: Electron cooling

"Hybrid" model: fluid electrons, kinetic ions

- The **ion beam** is represented as an ensemble of macro particles
  - full 6d phase space, e.g.
  $$(x, x', y, y', t, P)^T$$
  for accurate tracking and for capturing non linearities
  - integrate the effect of cooling force + solenoidal magnetic field, in Δz

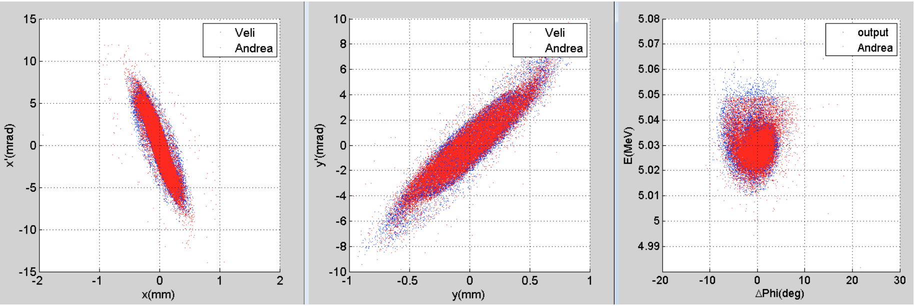
- The **electron beam** is represented as a fluid on a 3d cartesian mesh
  - it allows arbitrary electron density / velocity distributions
  - each cell (i, j, k) is characterised by

$n_{e,\ ijk}$	electron density [#/m^3]
$\vec{v_{ijk}}$	average electron velocity [c]
$\Delta_{e\perp,\ ijk}$	electron transverse temperature
$\Delta_{e\parallel,\ ijk}$	electron longitudinal temperature

  - automatic tricubic interpolation of each quantities, to work at any arbitrary location (e.g. ion positions)
  - integrate the Euler equation of an incompressible fluid, in Δt [in progress]

Example: RFQ

(Thanks to Alessandra Lombrardi, Veliko Dimov)

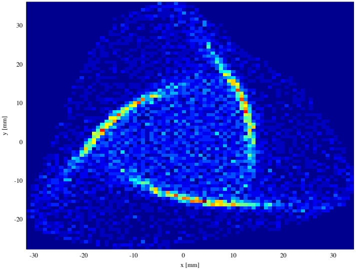


Simulated the tracking in the RFQ. Veliko kindly provided the field maps and the input distribution.

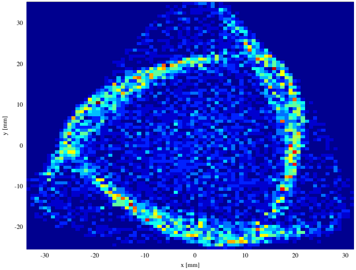
- Accelerates proton from $E_{\mathrm{kinetic}} = 40$ keV to $E_{\mathrm{kinetic}} = 5$ MeV
- Expected transmission confirmed = ~25%
- Field map: $27 \times 27 \times 10'000$
- Tracking 100'000 particles takes less than 1 minute on a modern PC
- 3D tracking of losses

Example: Lead Ion Source for Linac 3

(Thanks to Alessandra Lombrardi, Marc Maintrot, Ville Toivanen)

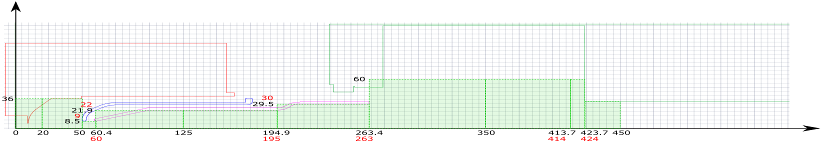


(no space charge)



(with space charge)

- IBSimu-generated input distribution, contains oxygen ions from O^{1+} to O^{8+} , and lead ions from Pb^{21+} to Pb^{36+}
- Lead ions are accelerated from $P = 146$ MeV/c to $P = 450$ MeV/c
- IBSimu-generated field maps (embed the effects of space-charge)



Example: Beam-beam force (1/2)

(Many thanks to Elias Métral for reviewing these results and for answering all my questions)
As RF-Track solves the full set of Maxwell equations for $\vec{E}$ and $\vec{B}$. With two bunches going in opposite directions, it is possible to simulate beam-beam effects.

Benchmarks against analytical model

- Beam-beam force for a gaussian beam:

$$F_r(r) = \frac{N q_1 q_2}{2\pi\varepsilon_0 l_b}\left(1+\beta_{\text{rel}}^2\right)\frac{1-\exp\left(-\frac{r^2}{2\sigma^2}\right)}{r}$$

- Simulation setup
  - bunch with 20 million particles
  - profile: gaussian transversely / uniform longitudinally
  - mesh 256×256×128
    $\sigma_z \gg \sigma_x, \sigma_y$
  - transverse velocities = 0 (to match the analytical assumption)

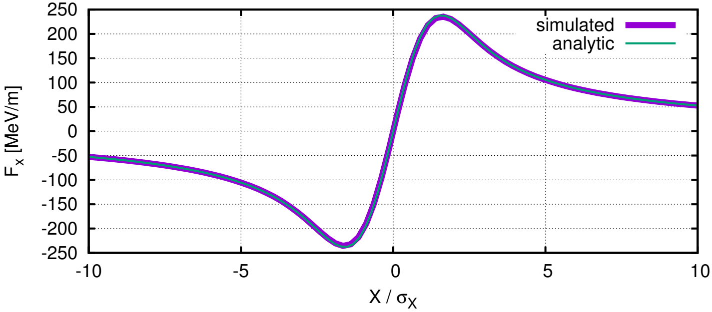
  computational time : $t_{\text{cpu}} \approx 30$ s on my laptop

Example: Beam-beam force (2/2)

LHC-like parameters:

- head-on collision
- $P_z = 7$ TeV / c;
- $P_x = P_y = 0$
- $\sigma_z = 75.5$ mm
- $\beta^\star = 0.55$ m
- normalised emittance = 3.75 mm·mrad;

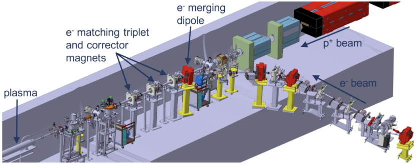
Force is calculated in the range $[-10\sigma, +10\sigma]$



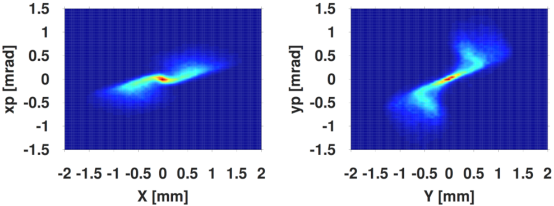
Example: AWAKE primary beam lines

(J. S. Schmidt, A. Latina, THPAB050, IPAC 2017)

The AWAKE project at CERN will use a high-energy proton beam at 400 GeV/c to drive wakefields in a plasma.



The amplitude of these wakefields will be probed by injecting into the plasma a low-energy electron beam (10-20 MeV/c), which will be accelerated to several GeV.



Upstream of the plasma cell the two beams will either be transported coaxially or with an offset of few millimetres for about 6 m. Figure shows the electron phase space at the focal point after propagation on axis with the $3 \cdot 10^{11}$ proton bunch.

Last example: Electron cooling at LEIR

[VIDEO 1]

[VIDEO 2]

[VIDEO 3]

Summary and outlook

RF-Track:

- A new code has been developed: minimalistic, parallel, fast
  - powerful synergy of: C++, numerical algorithms, particle tracking
- It is flexible: can track any particles at any energy, all together, using a simple user interface
- It implements direct space-charge (and beam-beam)
- It is being documented, but it's simple enough to be already usable
  - you are welcome to use it!

Possible improvements:

- Implement indirect space-charge
- Implement intrabeam scattering
- Embed the electron cooler in realistic fieldmap
- ?

Availability: a pre-compiled version is on lxplus, with simple clarifying examples