Virtual-IPM

A modular framework for IPM
(and other related) simulations

Outline

- Motivation
- Structure of the program
- Use cases
- Available models
- Benchmarking + Testing

Motivation

- Although many different solutions were available they could not be easily combined
- A clear, separate way of configuring is important
- Cover many different usage scenarios without diving into the source code
- The goal is to have a tested, documented, maintained code which is easy to use and easy to extend

Built with ...

Python 3.5 (+ Python 2.7 compatibility)

PyQt5 (+ PyQt4 compatibility)

matplotlib

numpy + scipy

+ anna, injector, ionics, pandas, pyhocon, reactivex, six

Why Python?

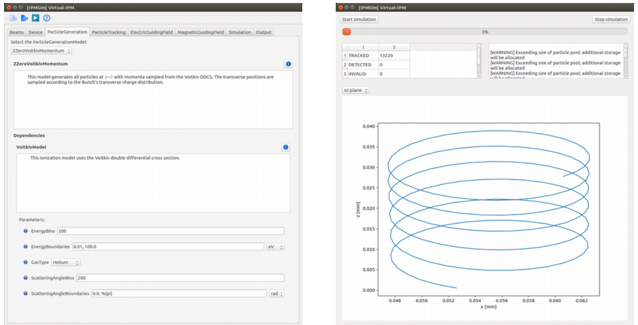
- Concise and intuitive syntax → clean and well understandable code
- No code "overhead" (e.g. resource allocation is done by the compiler) → focus on the logic / algorithm → move faster from code to results
- "Batteries included" → Python ships with a huge standard library + tons of third-party packages are available
- Native code inspection allows for integration with a GUI

What about performance?

- Python merely serves as an interface to the "computational libraries" and only does the job of "gluing together"
- Those components who do the heavy lifting are compiled in C for example (e.g. numpy)
- Different options are possible as for example tensorflow in order to harness GPU power

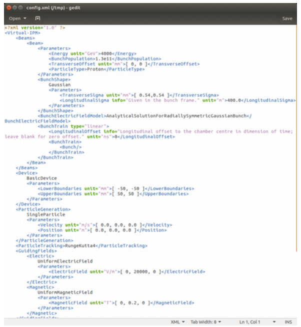
How does it work?

Graphical User Interface



generate

XML Configuration File



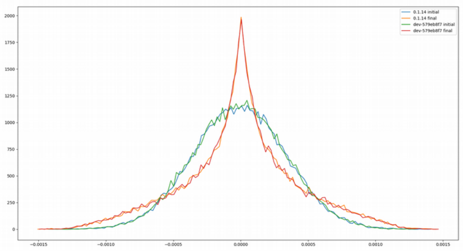
Input

Application Core / Simulation

generate

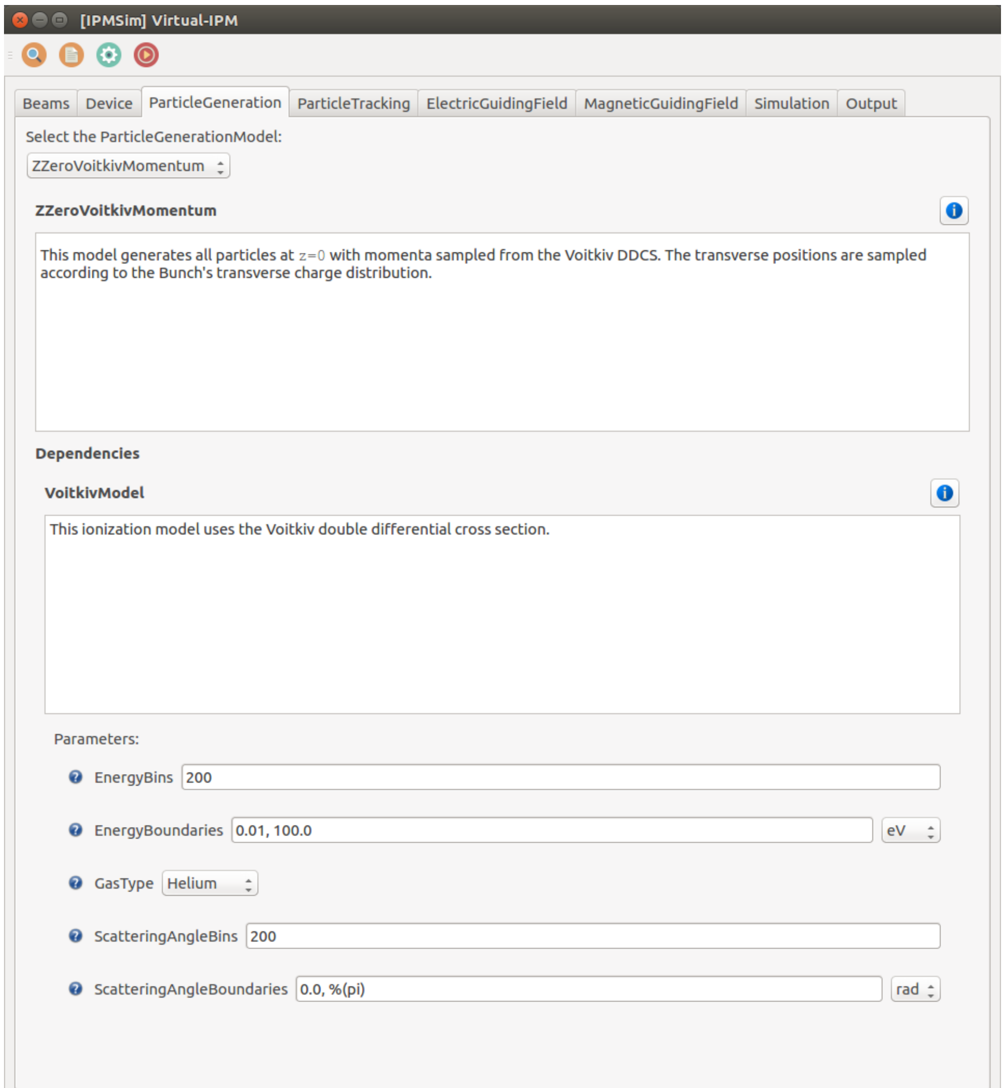
Feedback

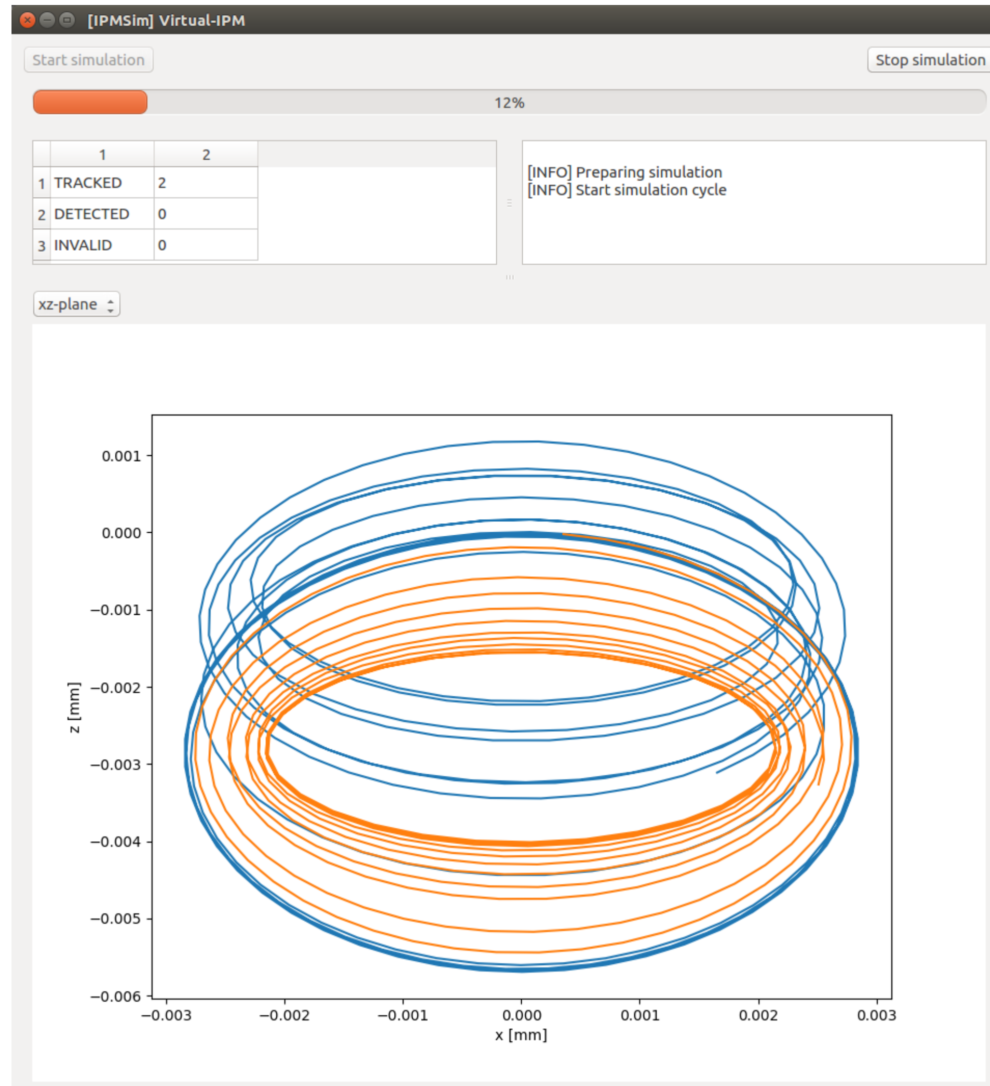
Output / Results



- GUI can be used for specifying the parameter values
- Simulation expects a configuration file as input
- Output can be controlled via configuration parameters

Graphical User Interface





Use cases

- ✓ Beam space charge
- ✓ Guiding field non-uniformities
- ✓ Correlation between electron and ion detection
- ✓ Multiple beams (e.g. electron lens)
- ✓ Particle trajectories

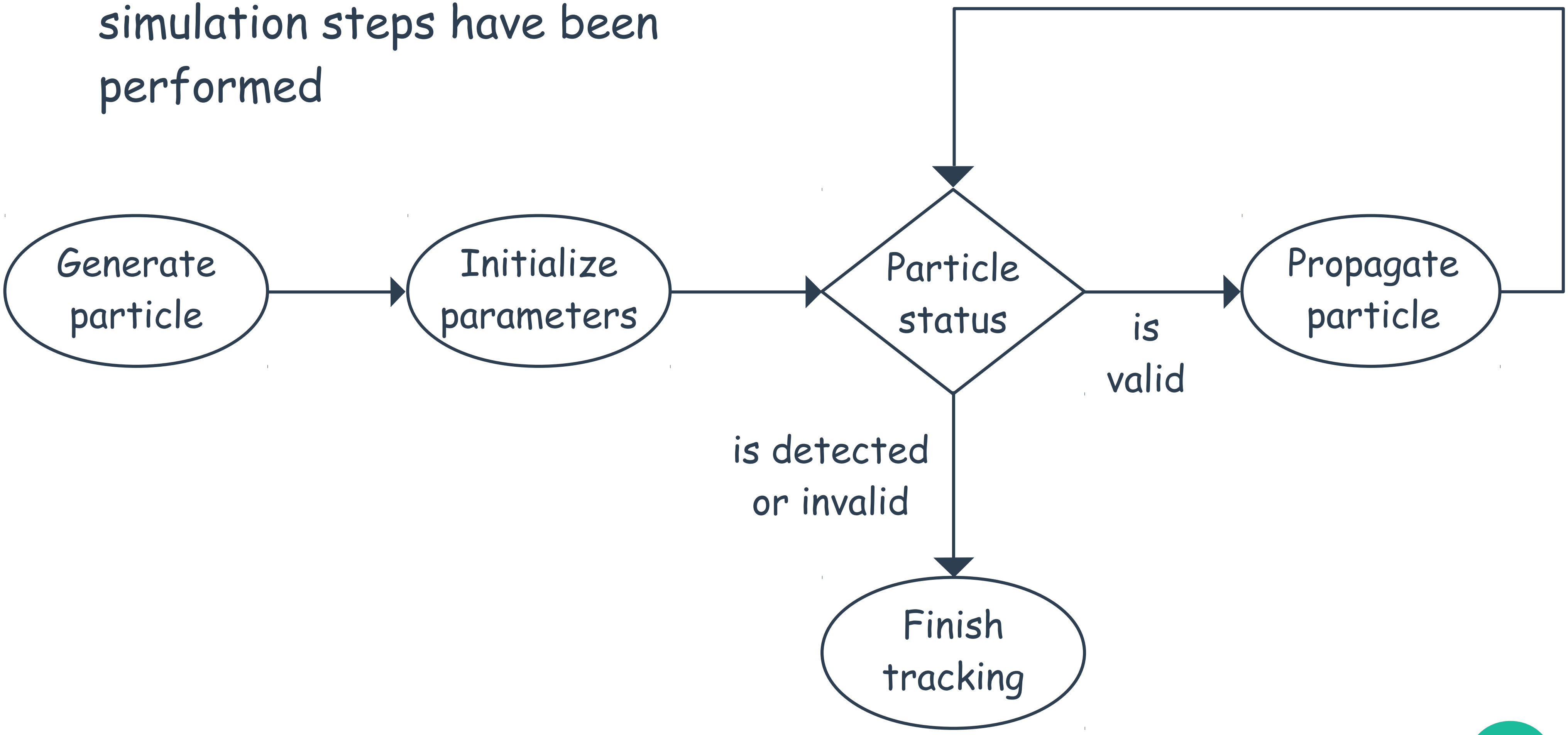
- ⏳ Electron background
- ⏳ Electron wire scanner
- ⏳ Gas-jet for IPM and BIF
- ⏳ Secondary electrons
- ⏳ Meta-stable excited states for BIF

Modules

- Particle generation
- Particle tracking
- Particle detection
- Guiding fields
- Bunch shapes
- Beam fields
- Output
- Several other auxiliary components

Particle life cycle

- Loop until the specified number of simulation steps have been performed



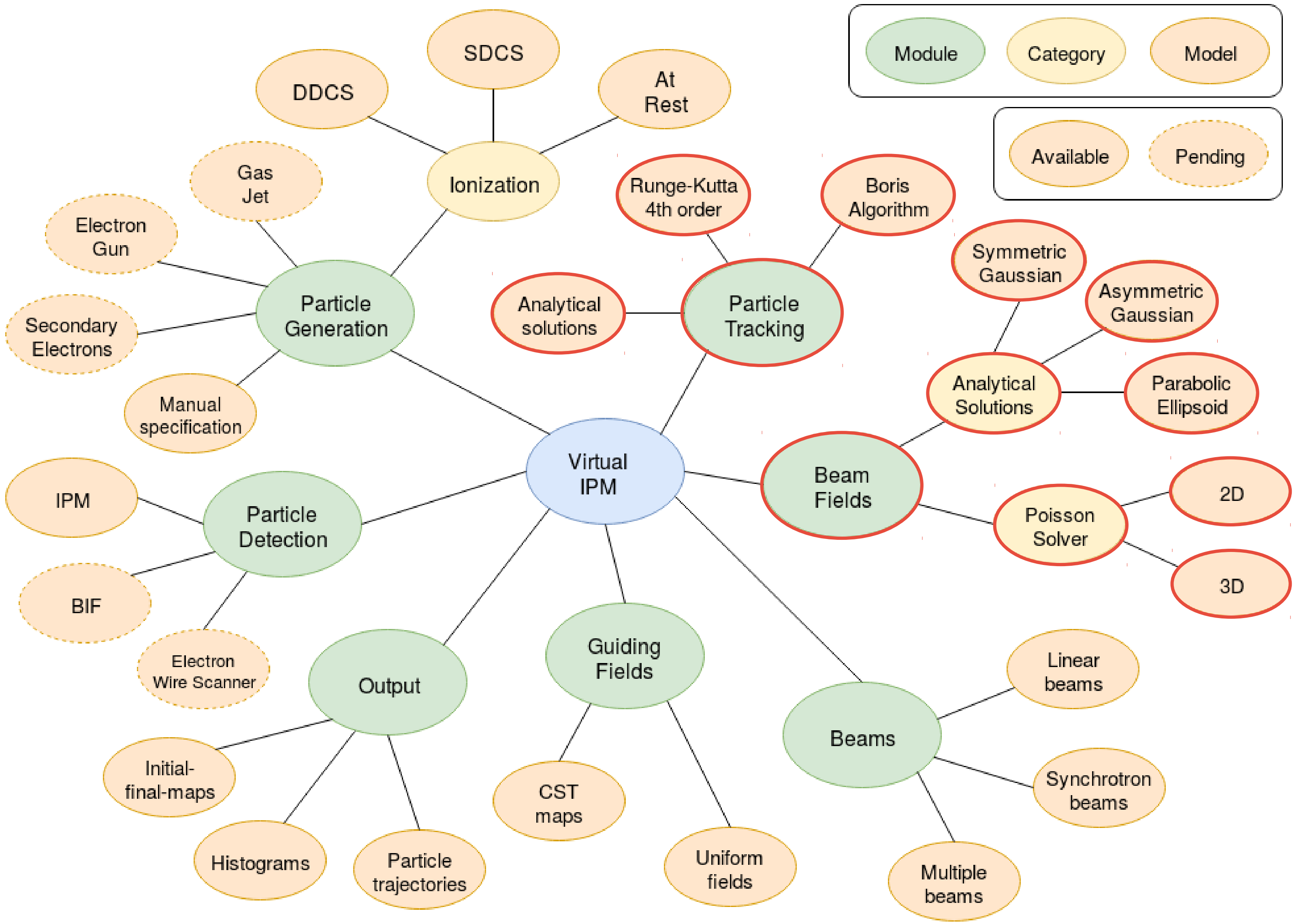
Inspired by ...

... PyECLOUD-BGI - analytical formula for particle tracking + Bassetti & Erskine bunch field model (thanks to G. Iadarola)

... GSI-code – analytical formula for the electric field of ellipsoids (thanks to P. Forck, S. Udrea)

... JPARC-code – Runge-Kutta-4th order particle tracking + 2D Poisson solver (thanks to K. Satou)

What components are available?



Bunch field models

Symmetric Gaussian

- Solution is obtained from solving Poisson's equation in 2D
- Field is scaled with the fraction of the long. density

Asymmetric Gaussian [1]

- Uses the complex error function to solve Poisson's equation in 2D
- Field is scaled with the fraction of the long. density

1) M.Bassetti, G.A.Erskine: "Closed expression for the electrical field of a two-dimensional Gaussian charge", CERN-ISR-TH/80-06, 1980

Parabolic Ellipsoid [2]

- Charge density ~ $1 / ab^2 * (1 - r^2/b^2 - z^2/a^2)$
- Uses elliptical coordinates to solve Poisson's equation in 3D

Poisson Solver

- Solve Poisson's equation numerically in either 2D or 3D
- For 2D the field is scaled with the fraction of the long. density

2) M.Dolinska, R.W.Mueller, P.Strehl: "The Electric Field of Bunches", 2000

Particle tracking models

Analytical Solution

For the special case of

- uniform electric and magnetic fields and $B_z = E_z = 0$

Runge-Kutta 4th order

- Solve differential equation of the form **d/dt y = f(t, y)** by turning it into a linear equation with four intermediate evaluations of **f**

Boris algorithm

- Position and momentum are shifted by half a time step against each other (momentum is "behind")
- Uses a transformation to separate electric and magnetic field terms
- Widely used in plasma simulations

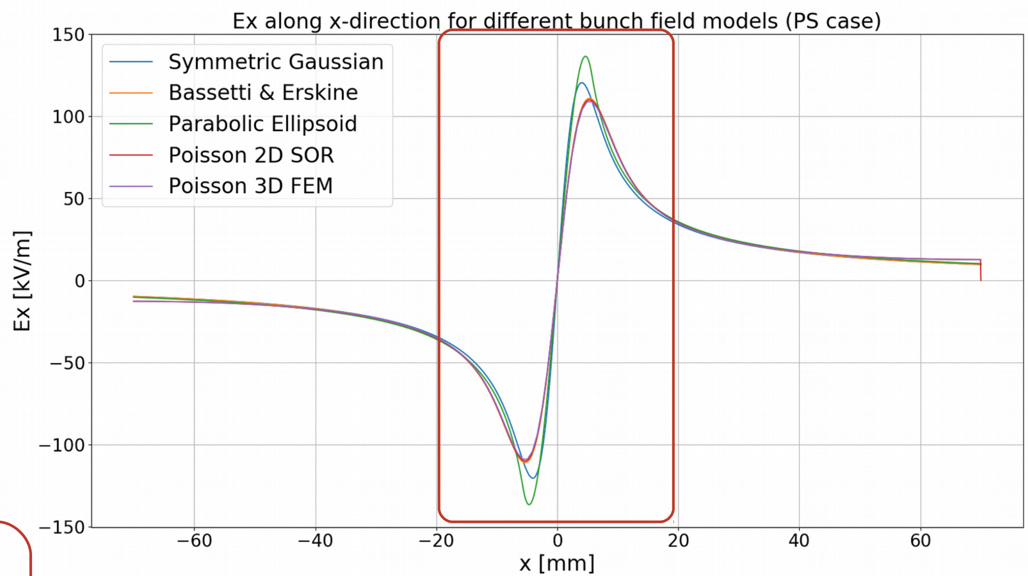
Benchmark cases

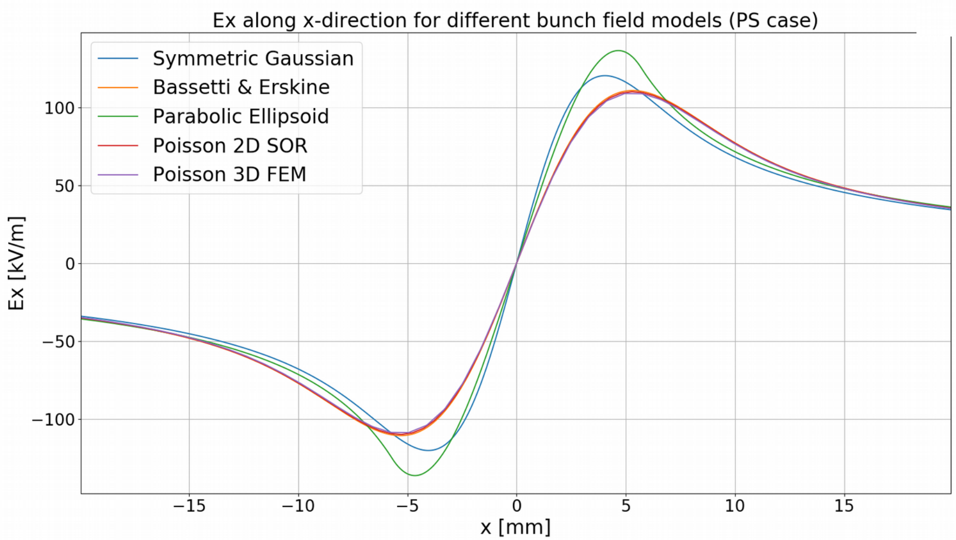
	LHC case	PS case
Energy	6.5 TeV	25 GeV
Bunch pop.	1.3e11	1.33e11
Length (4σ)	1.25 ns	3.0 ns
Width, Height	229, 257 µm	3.7, 1.4 mm
Electrode dist.	85 mm	70 mm
Applied voltage	4 kV	3, 20 kV
Magnetic field	0.2 T	0 T

Comparison of bunch field models

PS case

- For the symmetric Gaussian: $\sigma = (\sigma_x + \sigma_y)/2 = 2.55$ mm
- For the parabolic ellipsoid: $a = \sqrt{5} \cdot \sigma_z$, $b = \sqrt{5} \cdot (\sigma_x + \sigma_y)/2$



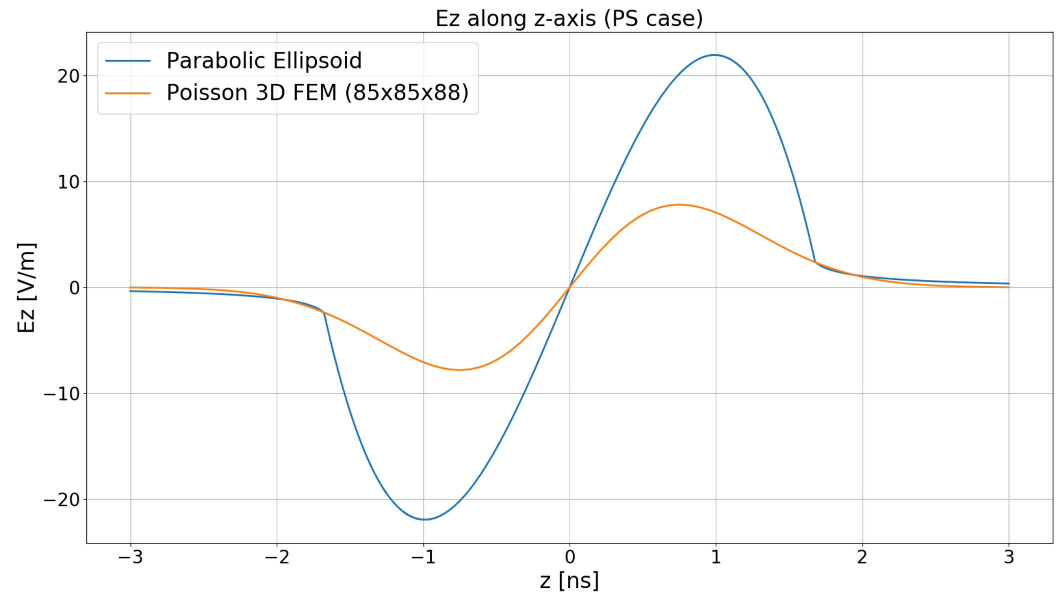


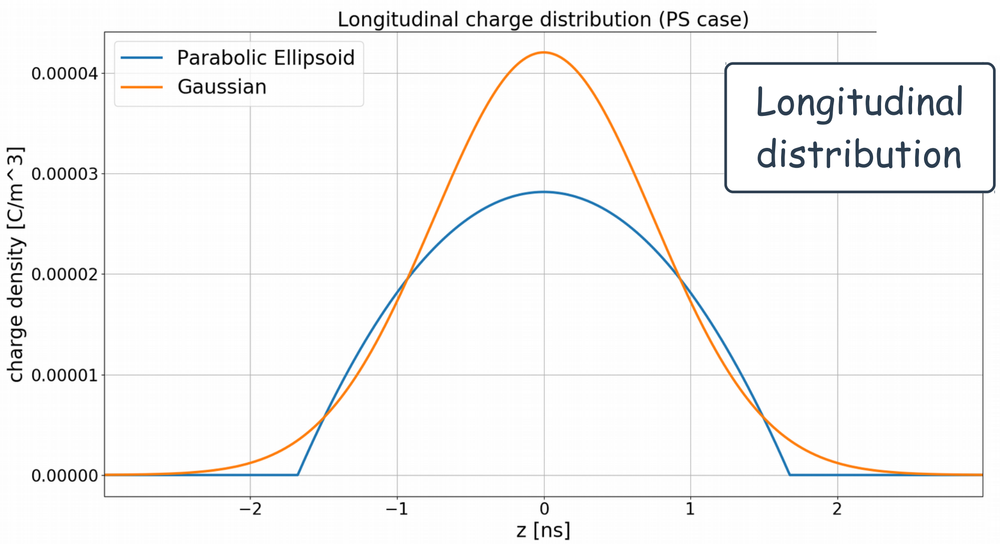
- Poisson 2D: grid spacing 0.5mm → 280x280 grid; 2818 iterations, 13 min.
- Poisson 3D: 170x170x22 grid → transverse grid spacing 0.82 mm, long. grid spacing 0.27 ns; 6 GB memory, 5 min.

Comparison of bunch field models

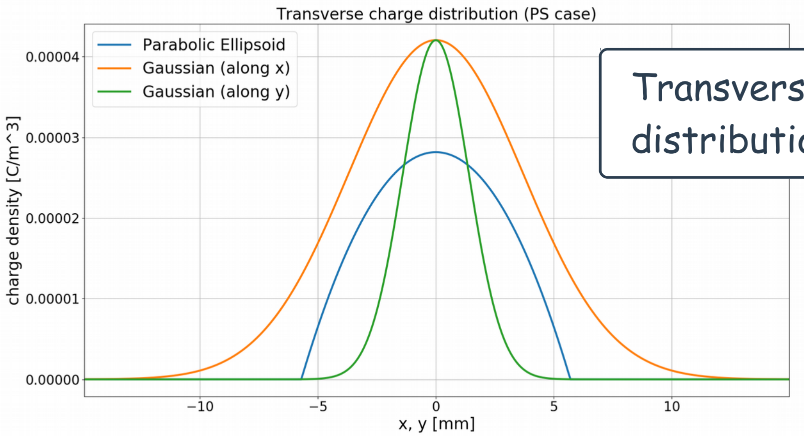
<u>PS case – longitudinal field</u>

- Long bunch ($\sigma_z/\sigma_x \approx 1.6e3$) → small longitudinal field is expected
- Field is in the order of magnitude ≈ 10 V/m
- For the parabolic ellipsoid the charges are closer to the z-axis, especially for $z \neq 0$





Longitudinal distribution

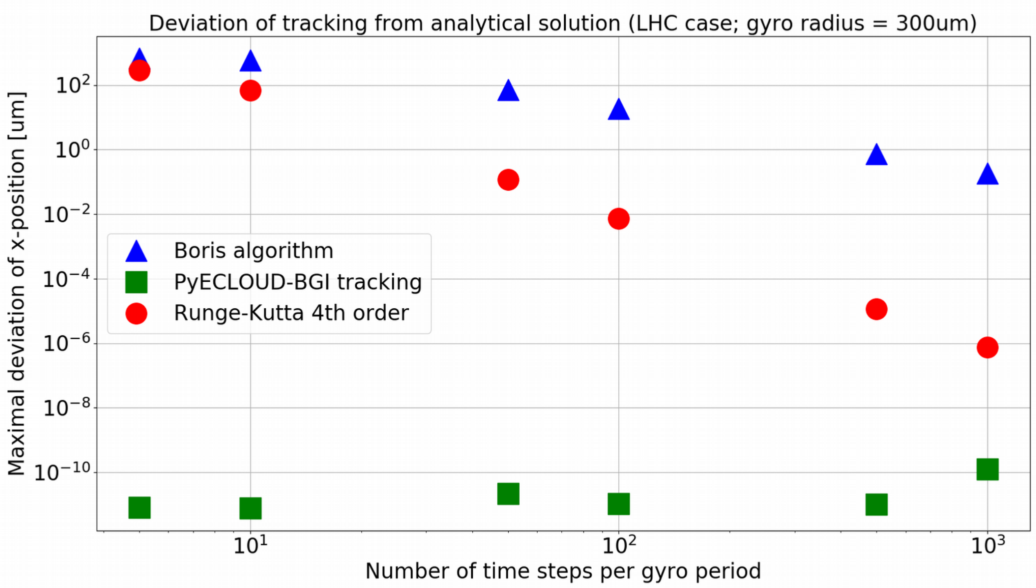


Transverse distribution

Comparison of tracking algorithms

<u>LHC case – gyro motion</u>

- Simulate gyration with 300μm radius
- No beam fields → compare with analytical solution
- Cyclotron period ≈ 0.178ns, extraction time ≈ 4.47ns → simulate 30 gyrations



- PyECLOUD-BGI is analytical solution of e.q.m. → good accuracy
- Runge-Kutta 4th order peforms better than Boris algorithm
- Reasonable results can be obtained with RK4 for 50 steps per period

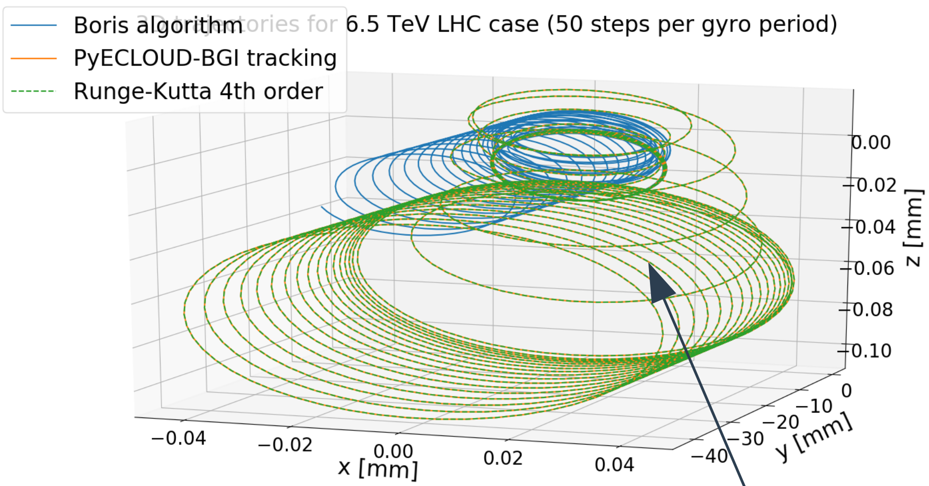
Deviation in y-direction (only electric field acceleration) is found to be negligible

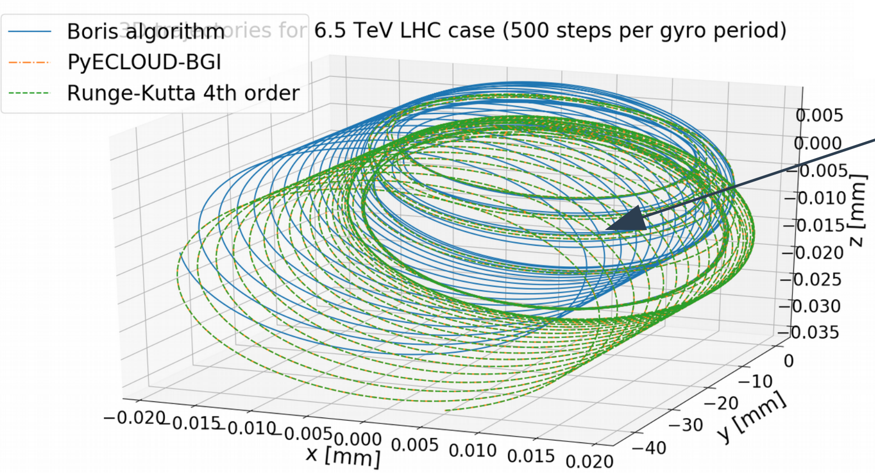
Similar behavior for ExB-drift in uniform E-field

Comparison of tracking algorithms

LHC case – Trajectories in beam field

- Initial energy: 1 eV (→ from DDCS)
- Particle generated at t=0, z=0; beam has offset $z = 4\sigma_z$
- Magnetic field in y-direction: 0.2 T





Running for 500 steps per gyro period shows less increase in gyro momentum and a smaller ExB-drift

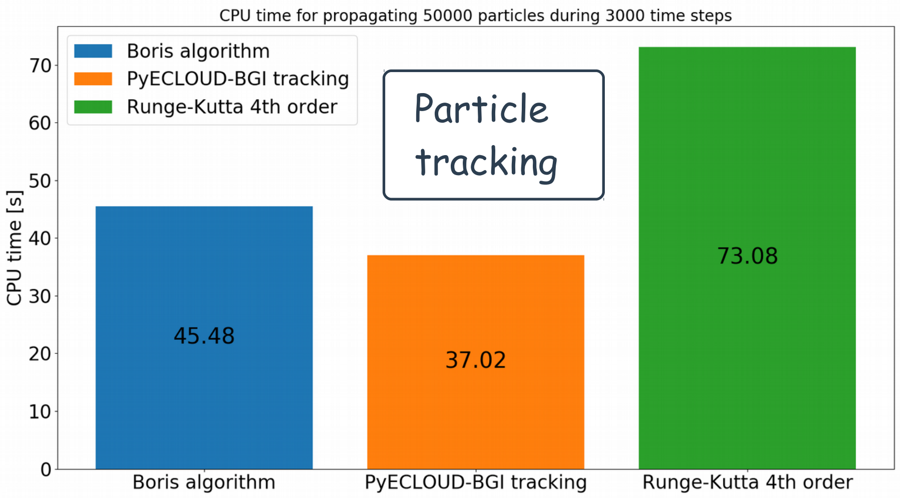
No <u>net</u> ExB-drift expected because field is symmetric around x=0 and contributions from either side should cancel

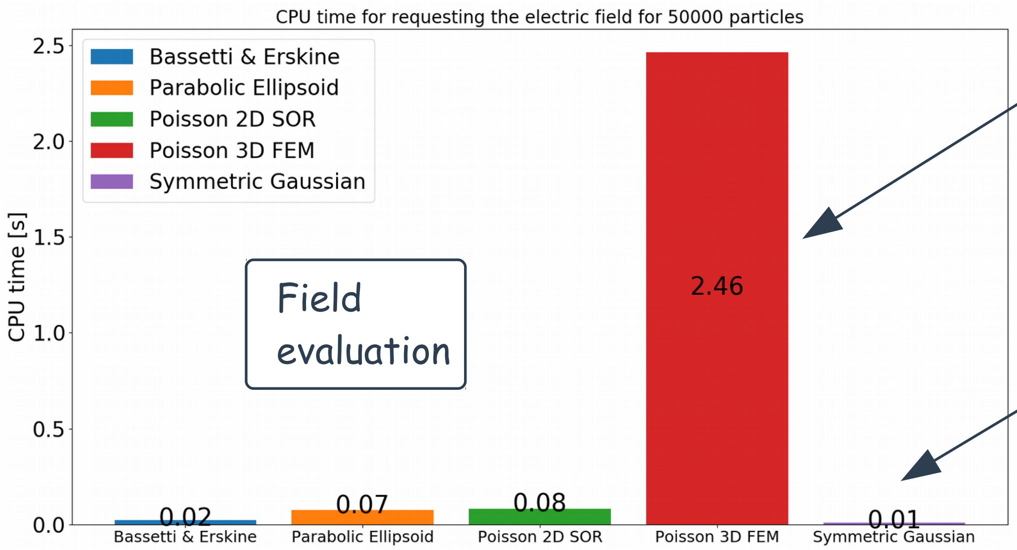
→ For large beam fields the time step must be chosen a lot smaller in order to obtain similar accuracy

Comparison of bunch field models

Efficiency/performance – CPU benchmarking

- CPU: Intel Core i7-5500U @ 2.40GHz x 4
- Memory: SO-DIMM DDR3 1600MHz



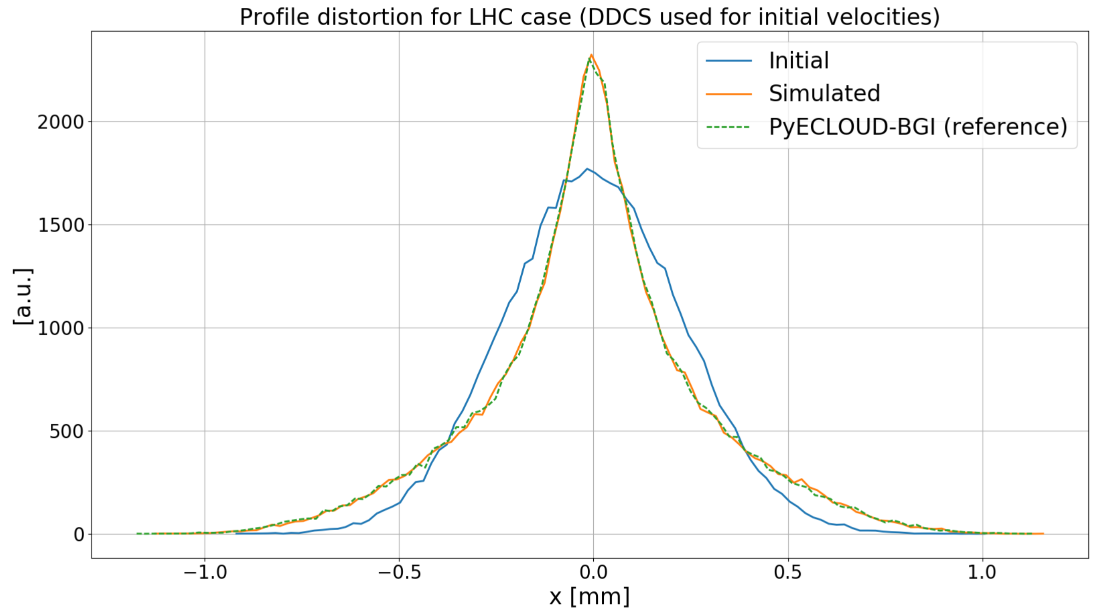


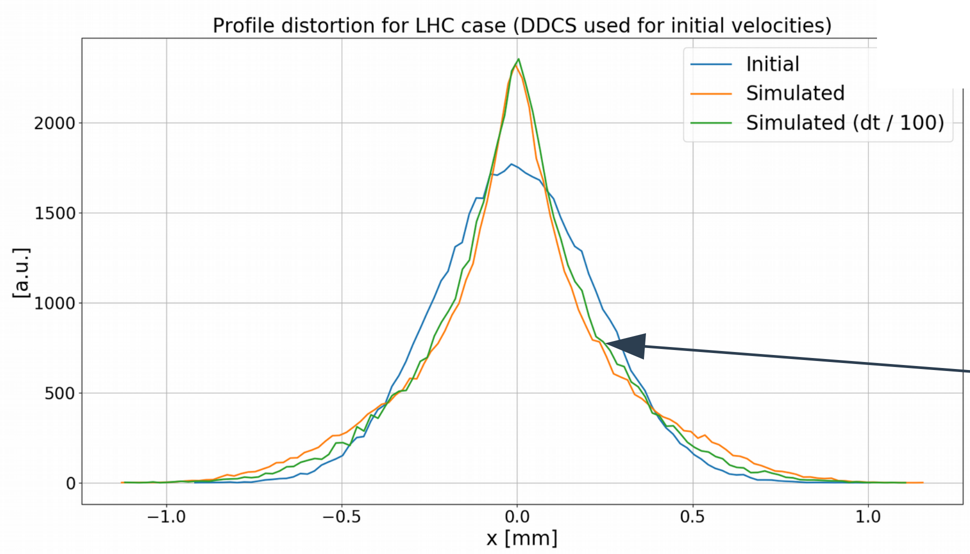
Poisson3D model evaluates the field for each position in a Python for-loop → requires more CPU time

Other models evaluate the fields in external C-for-loops (via numpy or scipy) → fast computation

LHC Case – Profile distortion

- Good agreement between original PyECLOUD-BGI code and corresponding setting → successful migration



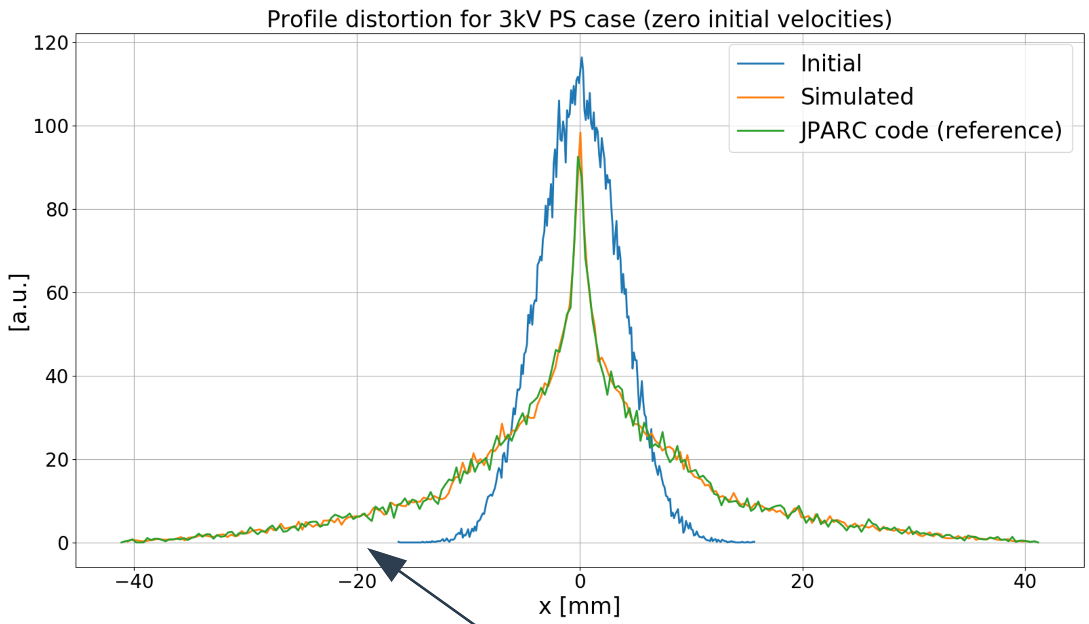


- Simulating for Δt = 10ns / 3200 = 3.125ps gives already reasonable results compared with Δt = 0.03125ps

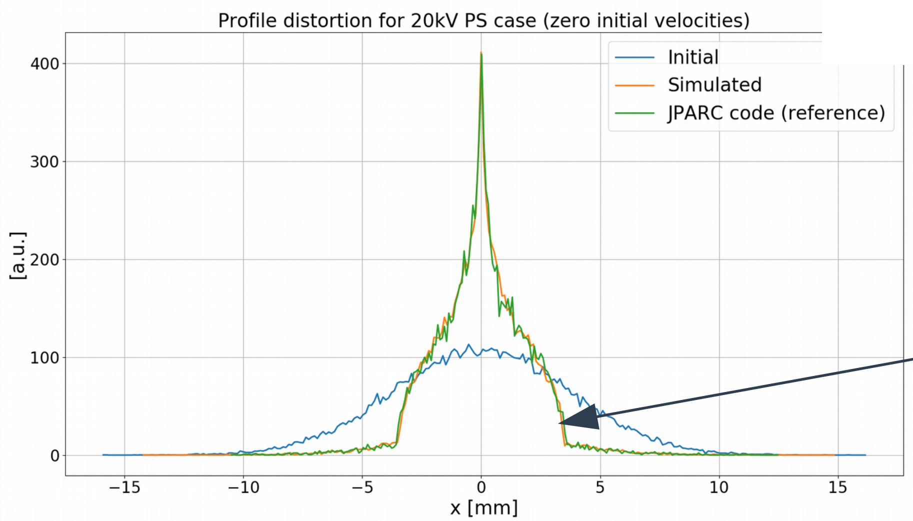
Slightly more signal near x=0 because the gyroradius of those electrons is not increased as much

PS Case - Profile distortion

- Good agreement between JPARC-code and the corresponding models → successful migration



Electrons even cross x=0 (i.e. to the other side of the profile)



Electrons are pulled towards the center of the profile

Summary

- Various models have been successfully migrated + new models have been added
- Benchmarking with existing codes shows very good agreement
- The application's modular structure allows for easy implementation of new use cases
- A graphical user interface and code documentation allows for a convenient usage

Available on ...

... the Python package index:
https://pypi.python.org/pypi/virtual-ipm

... GitLab:
https://gitlab.com/IPMsim/Virtual-IPM
(git repository + issue tracker)

... GitLab pages:
https://ipmsim.gitlab.io/Virtual-IPM/
(documentation)

Extra slides

Configuration

- Configuration is handled by a separate framework (https://pypi.python.org/pypi/anna) → focus on the solution
- Various different parameter types are available
- Parameters are declared in the code and specified by the user
- Physical quantities can be specified in various units, the conversion is handled internally

Particle Generation

- Particle generation models define a way for particles to enter the simulation
- This is a very general requirement and thus many different implementations are possible:
  - ionization involving the beams
  - secondary electron emission
  - ...

<u>Available models:</u>

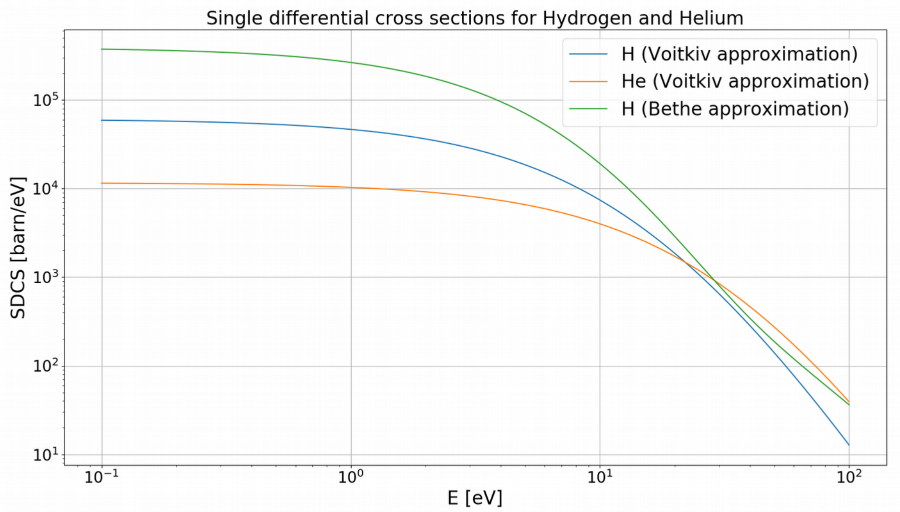
- Ionization
- At Rest
- Manual specification

- Each simulation cycle involves exactly one such way

Ionization

- Ionization involves two aspects: position and momentum generation

- Bunch shape models are responsible for the generation of positions (e.g. Gaussian)



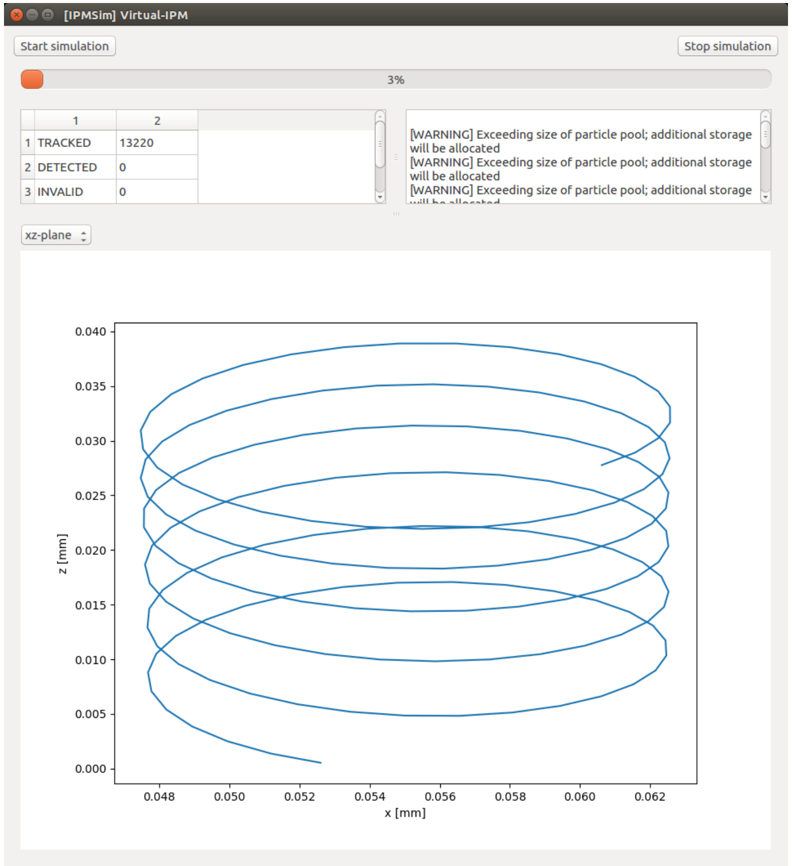
- Ionization cross sections build the basis for momentum generation

- Ionization cross sections are bundled in a separate package which is connected to the simulation

Particle tracking

- Particle tracking models are responsible for propagating particles during the simulation
- Particle tracking is an operation that takes place per time step and per particle
  → high computational demand
- Based on either analytical or numerical solutions of the equations of motion

Important aspects: accuracy and efficiency



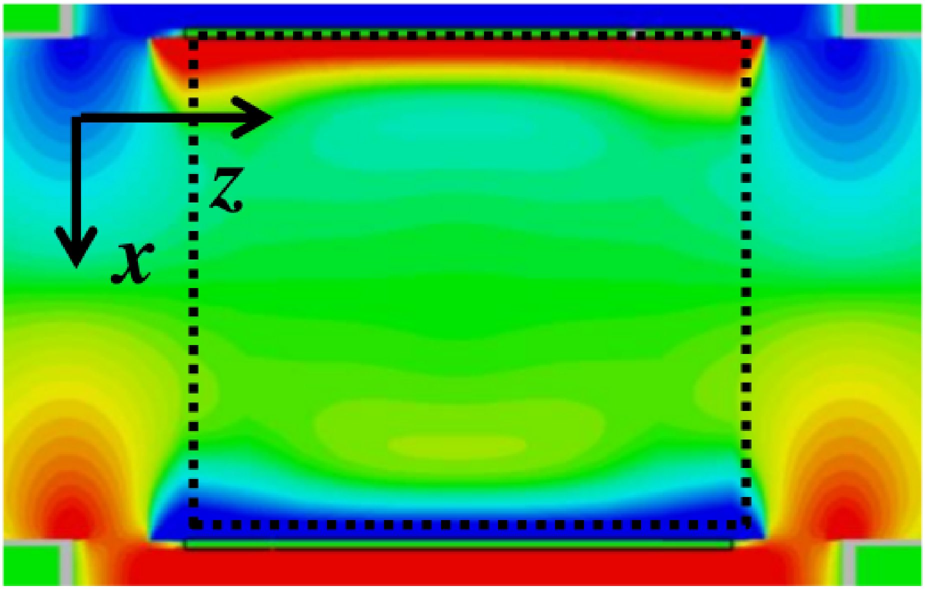
Particle detection

- Particle detection models ("Devices") define when particles are considered "detected" or "invalid"
- This is a very general requirement which applies to all use cases; for example such a model could compute the decay probability per particle and use it to decide when the particle is detected (BIF)
- Once a particle is detected or invalidated it is excluded from tracking and its parameters can be stored

Guiding fields

- Guiding field models define either the electric or magnetic component of the guiding fields

- Available models include:
  - uniform fields
  - 2D field maps
  - 3D field maps

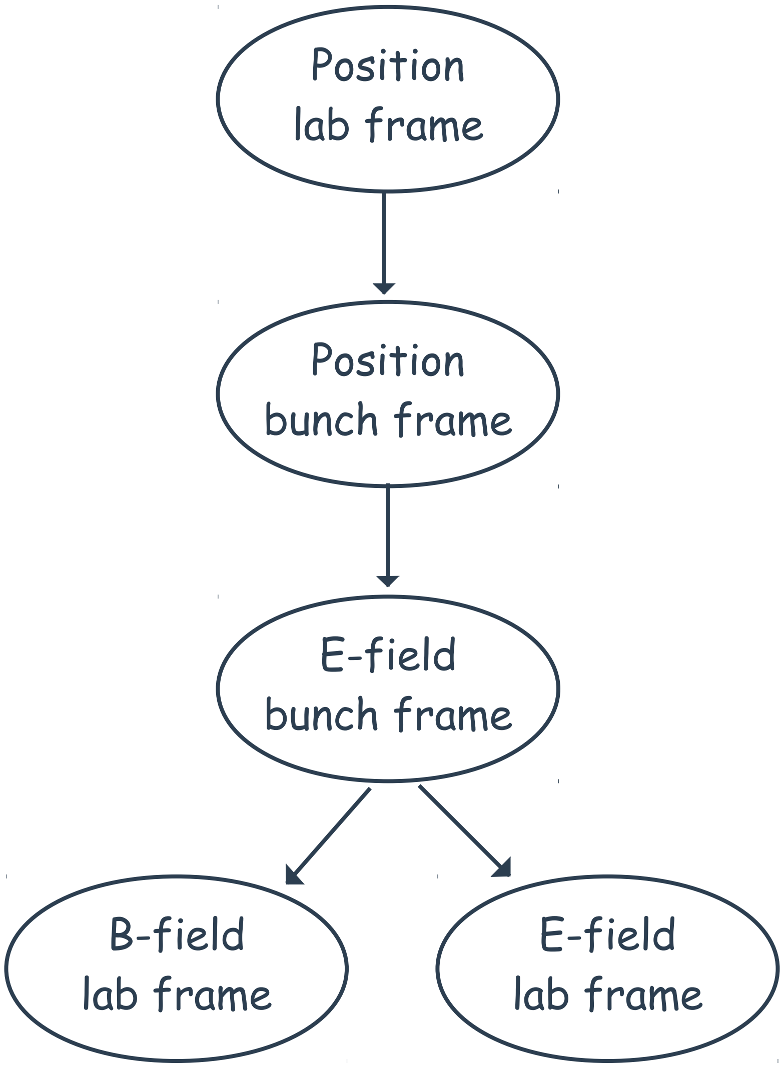


Study of electric guiding field for PS IPM, E_x at y=0 (K. Satou)

Guiding fields are evaluated per time step and per particle → efficiency plays an important role

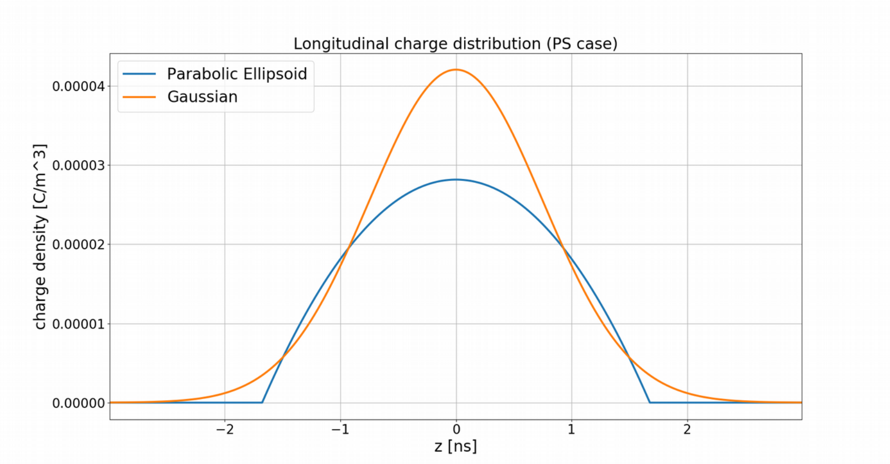
Beam fields

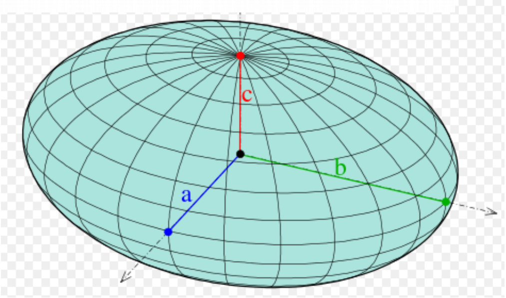
- The bunch electric field is defined and evaluated in the rest frame of the bunch
- Particle positions are transformed from the lab frame to the bunch frame (Lorentz transformation)
- Each bunch in the bunch train uses a separate Lorentz transformation (as they have different longitudinal positions)
- Electric and magnetic fields in the lab frame are computed via Lorentz transformation from the electric field in the bunch frame



Bunch shapes

- A bunch shape is involved in two processes:
  - Particle generation (position distribution)
  - Bunch electric field computation
- Different bunch electric field models might require different bunch shapes; for Poisson solvers the charge distribution is important
- Available shapes include:
  - Gaussian
  - Parabolically charged ellipsoid





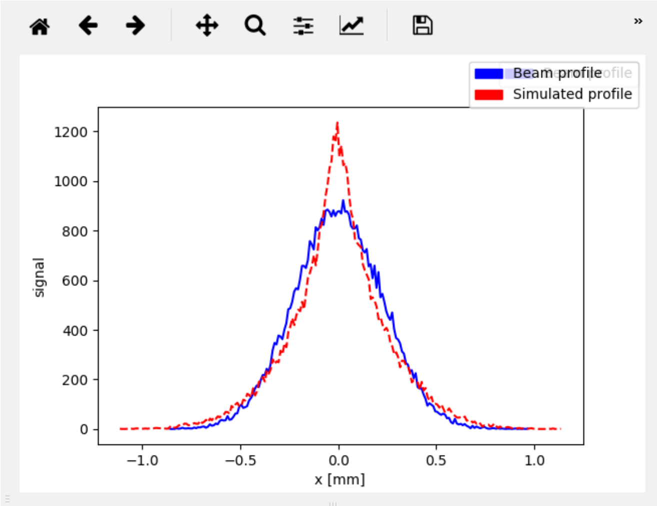
→ Different bunch shapes can be easily realized; e.g. based on measurement data

Simulation output

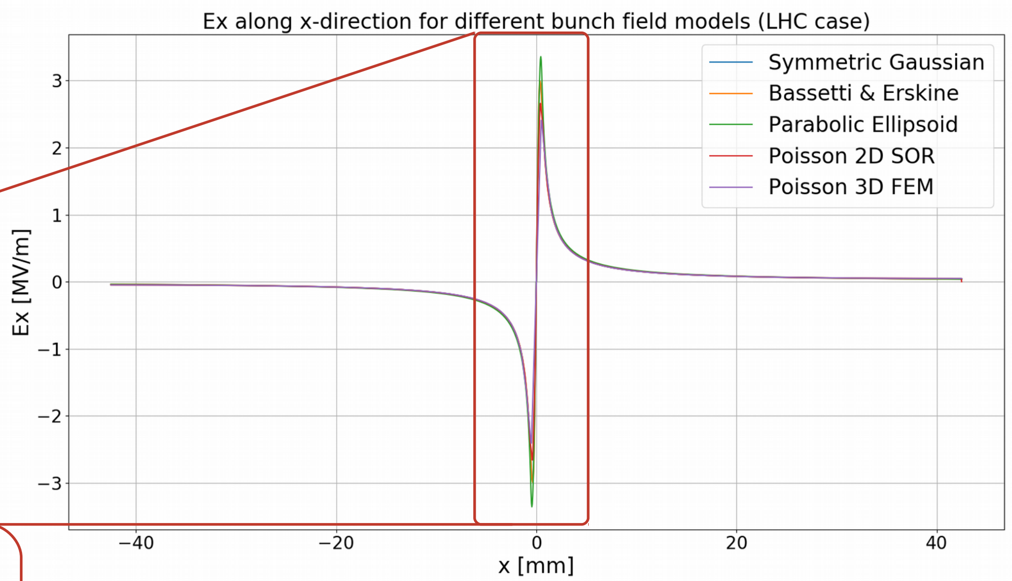
- Output recorders serve as an "information sink" for particle data; they are responsible for extracting this information and propagating it to external resources
- Two kinds of particle data information are considered:
  - Event based information such as initial and final positions of particles
  - Continuous information which is queried periodically such as particle trajectories

Available recorders:

- Initial → final maps (csv)
- Particle trajectories (csv)
- Profiles / Histograms (xml)

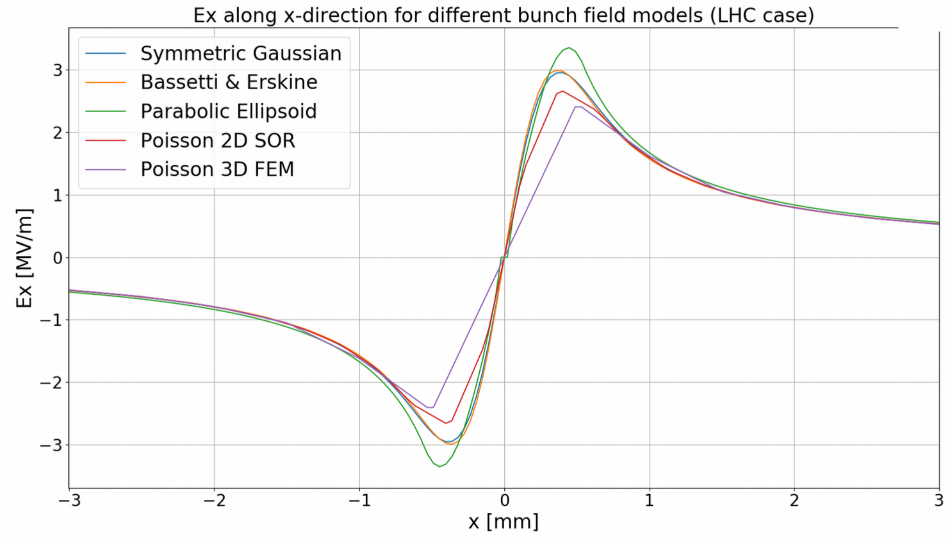


Comparison of bunch field models



LHC case

- For the symmetric Gaussian: $\sigma = (\sigma_x + \sigma_y)/2 = 243\ \mu m$
- For the parabolic ellipsoid: $a = \sqrt{5} \cdot \sigma_z$, $b = \sqrt{5} \cdot (\sigma_x + \sigma_y)/2$

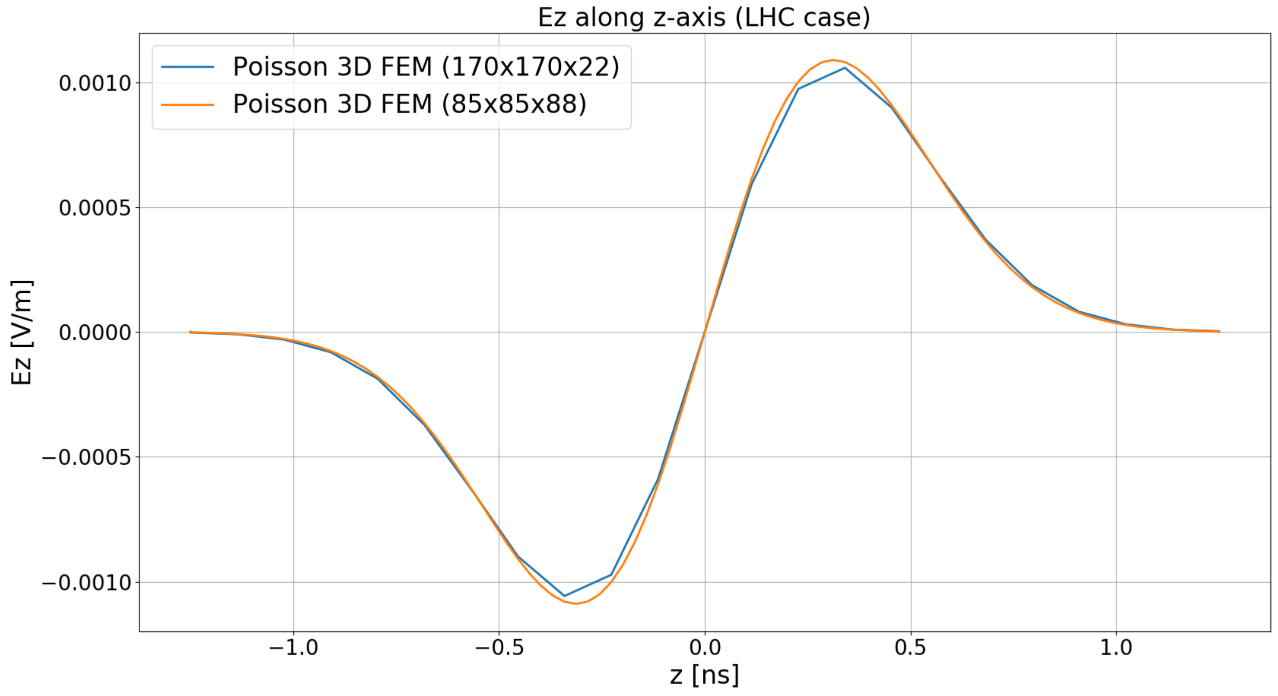


- Poisson 2D: grid spacing 0.25mm → 340x340 grid; 3669 iterations, 22 min.
- Poisson 3D: 170x170x22 grid → transverse grid spacing 0.5 mm, long. grid spacing 0.11 ns; 7 GB memory, 11 min.

Comparison of bunch field models

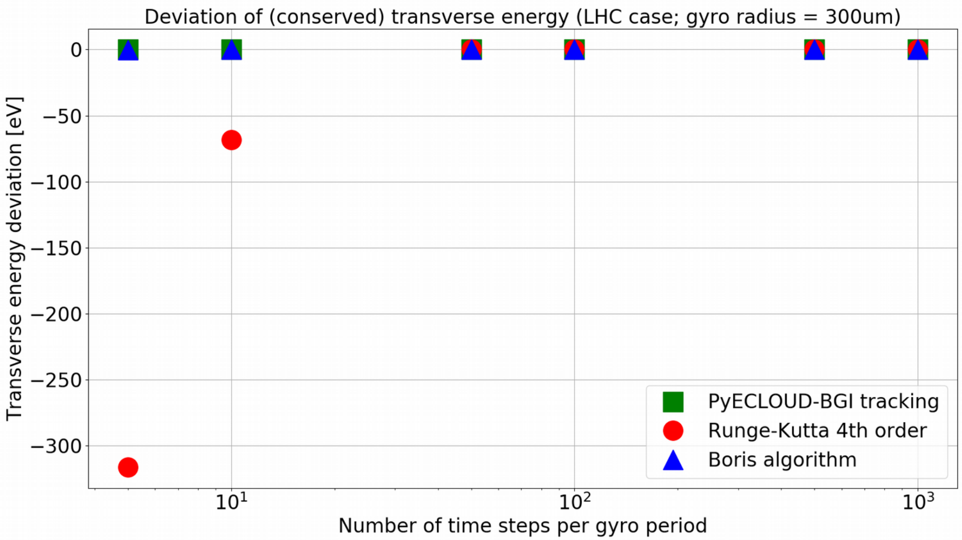
LHC case – longitudinal field

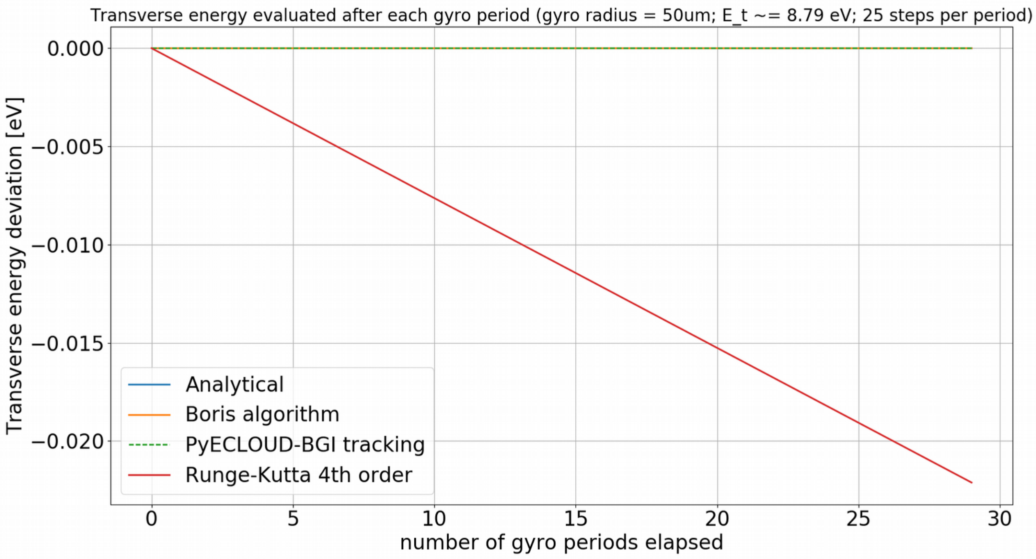
- Very long bunch → longitudinal field should be negligible
- $\sigma_z / \sigma_x \approx 2.67e6$
  (in the bunch frame)



Comparison of tracking algorithms

- Investigate (transverse) energy conservation for pure gyro motion
- No beam fields → gyro momentum is conserved (ideally)



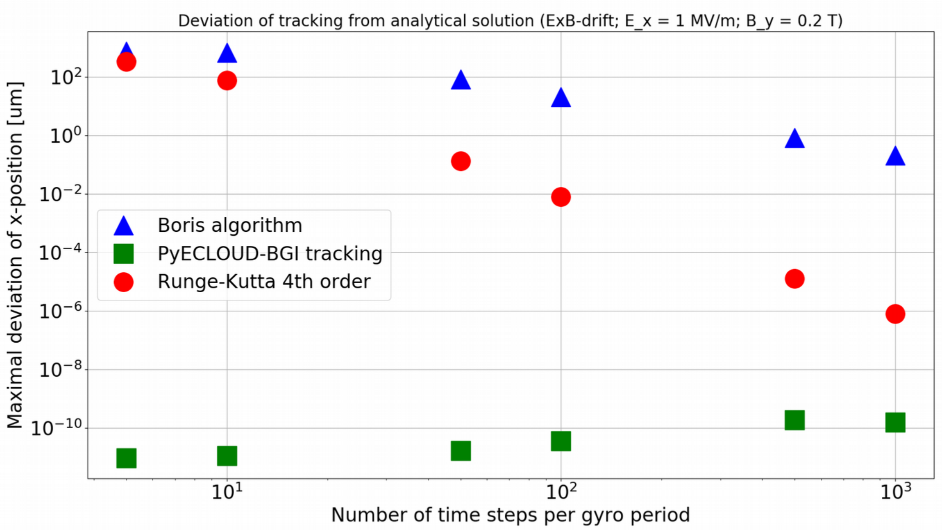


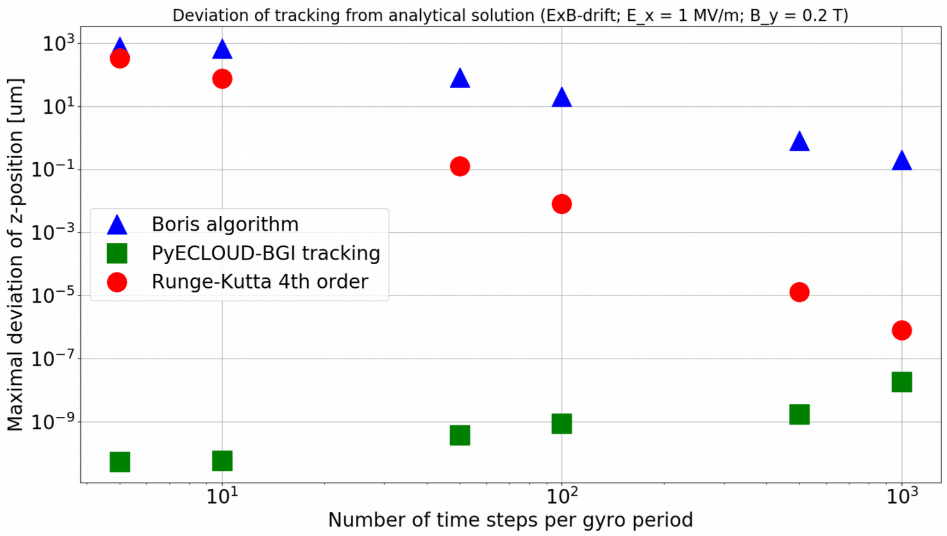
- Energy is very well preserved for PyECLOUD-BGI tracking and Boris algorithm
- Slight deviation for the Runge-Kutta 4th order method (→ no symplectic integrator) however deviation is negligible for the presented case

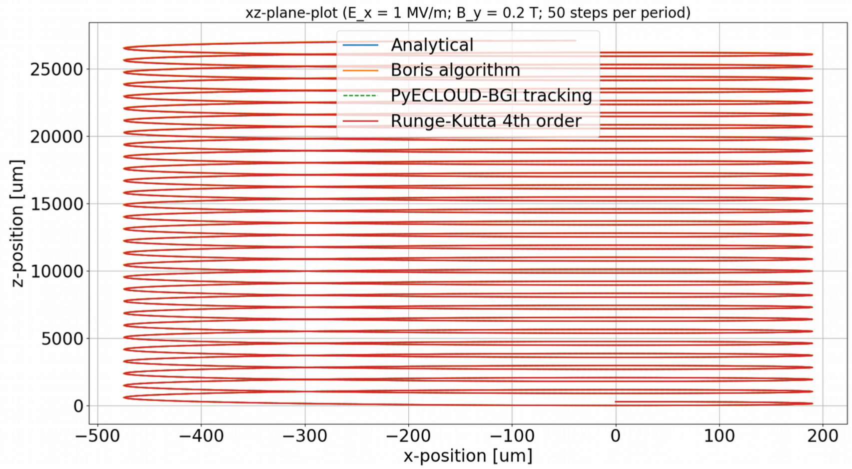
Comparison of tracking algorithms

LHC case - ExB-Drift

- Simulate gyration with 300µm radius
- (Constant) beam electric field in x-direction: 1 MV/m (6.5 TeV beam)
- Magnetic field in y-direction: 0.2 T
- Simulate 30 gyrations



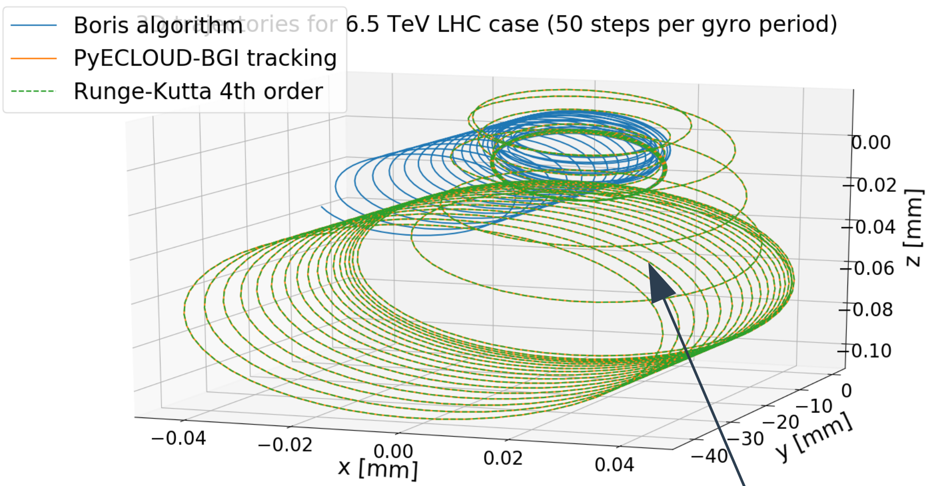


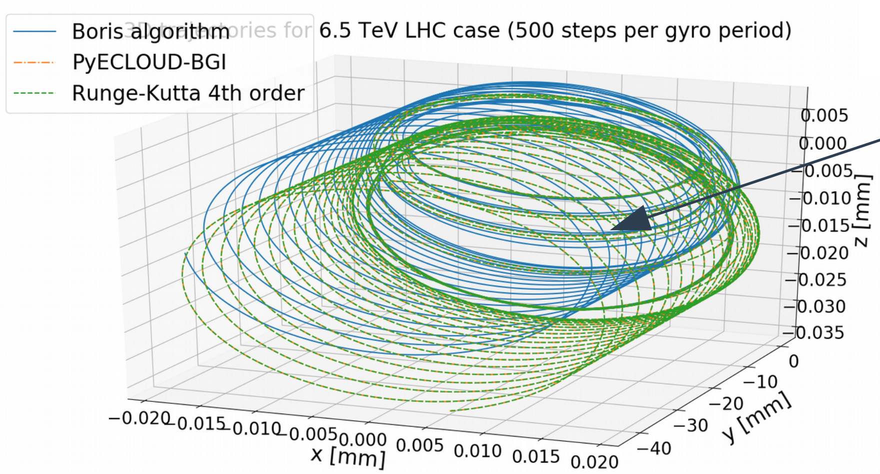


Comparison of tracking algorithms

LHC case – Trajectories in beam field

- Initial energy: 1 eV (→ from DDCS)
- Particle generated at t=0, z=0; beam has offset $z = 4\sigma_z$
- Magnetic field in y-direction: 0.2 T





Running for 500 steps per gyro period shows less increase in gyro momentum and a smaller ExB-drift

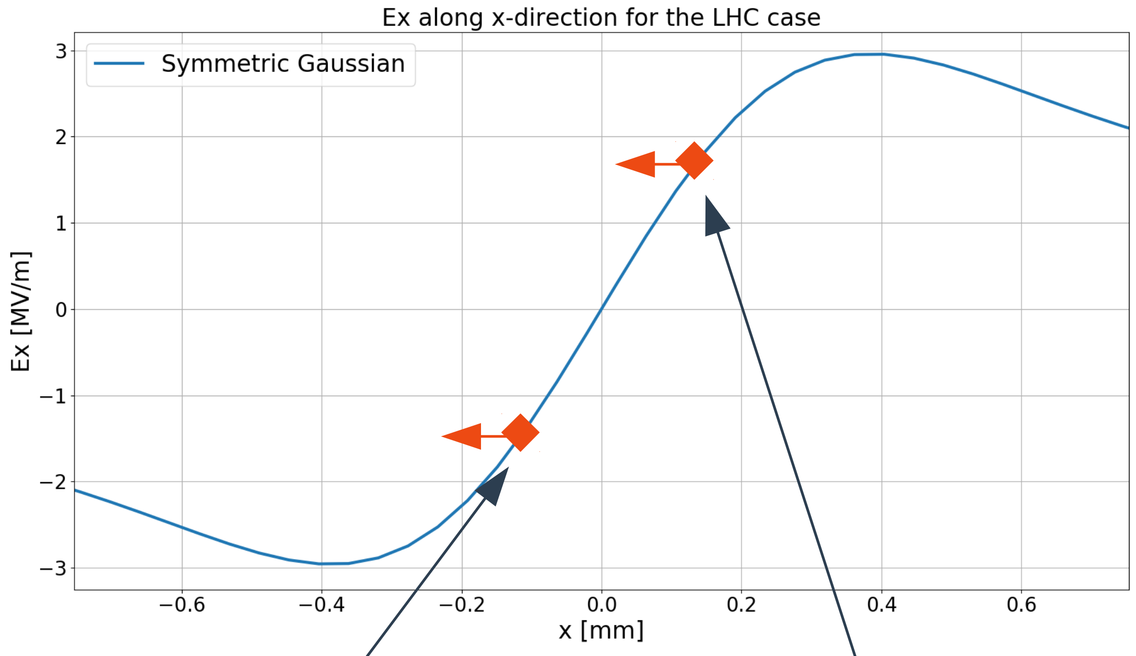
No net ExB-drift expected because field is symmetric around x=0 and contributions from either side should cancel

→ For large beam fields the time step must be chosen a lot smaller in order to obtain similar accuracy

Comparison of tracking algorithms

What causes the significant increase in gyro momentum and the net ExB drift?

- Algorithms "push" the particle during an update assuming that the electric field is constant during that push
- The electric field is evaluated at the beginning of the push



The repeated over- and underestimation of the accelerating and decelerating electric field leads to an increase in gyro momentum; the same holds for the ExB-drift as the contributions do not exactly cancel

Electron is decelerated towards x=0 → field is underestimated → deceleration is too small

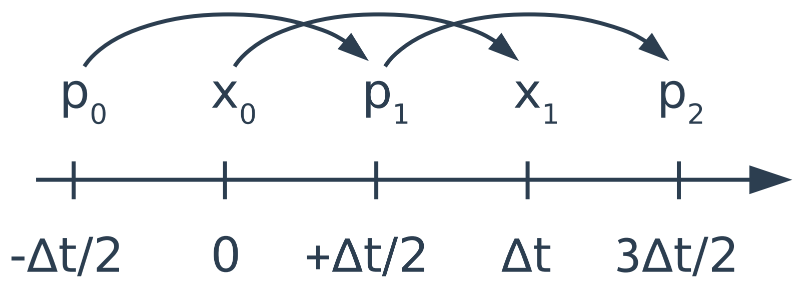
Electron is accelerated towards x=0 → field is overestimated → acceleration is too large

Comparison of tracking algorithms

- For 5000 steps per gyro period (Δt = 0.035 ps) the effect becomes negligible
- For the PS case the electric field is smaller and the effect is negligible also for larger time step sizes (Δt = 0.35 ps)

Why does the Boris algorithm perform better than the others?

- Position and momentum is shifted by Δt/2 for the Boris pusher (momentum is "behind")
- That is for each momentum update the electric field for the corresponding step + Δt/2 is used
- Because the field is linear close to x=0 using this average field at +Δt/2 is a good approximation



This shift could be used for other algorithms as well

Comparison of tracking algorithms

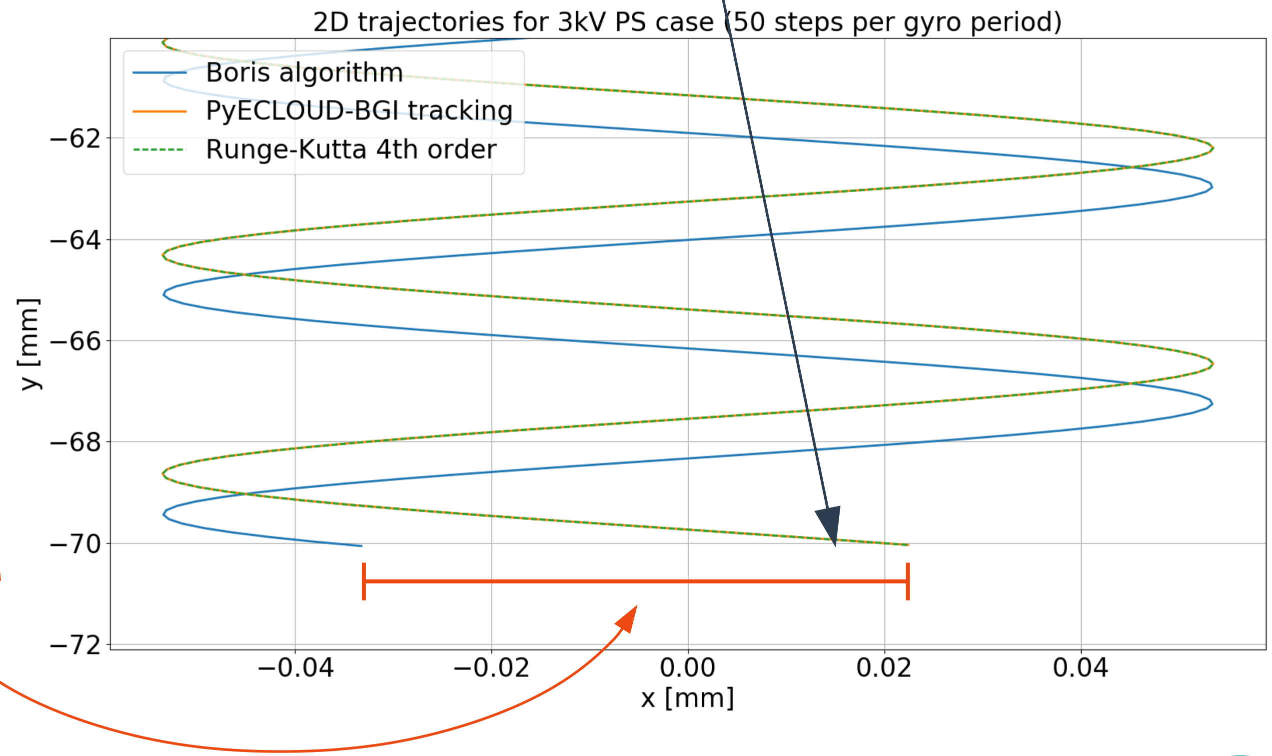
PS case – Trajectories in beam field

- Initial energy: 10 eV (→ from DDCS)
- Particle generated at t=0, z=0; beam has offset $z = 4\sigma_z$
- Magnetic field: 0.2 T
- Time step Δt = 3.57 ps

→ For Δt = 0.357 ps the deviation is found to be negligible

Detector at y = -70 mm

50 steps per gyro period

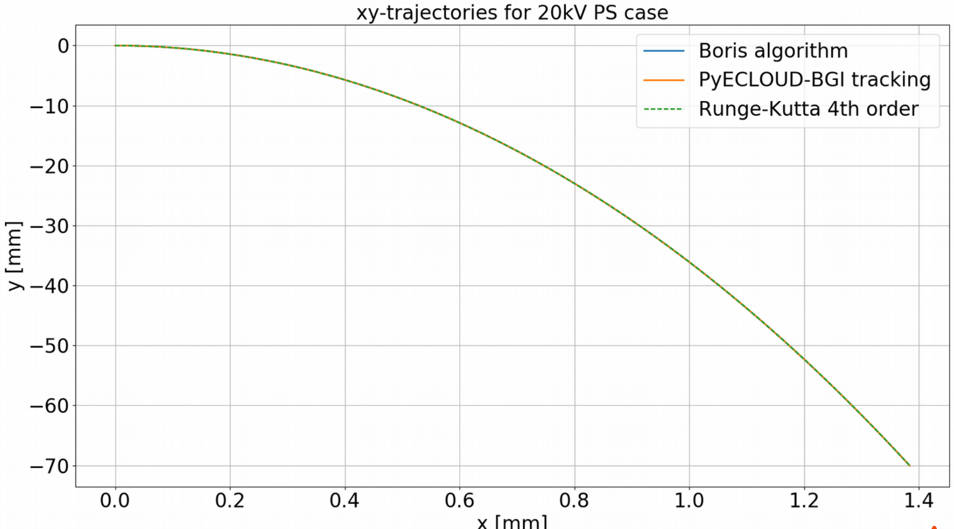


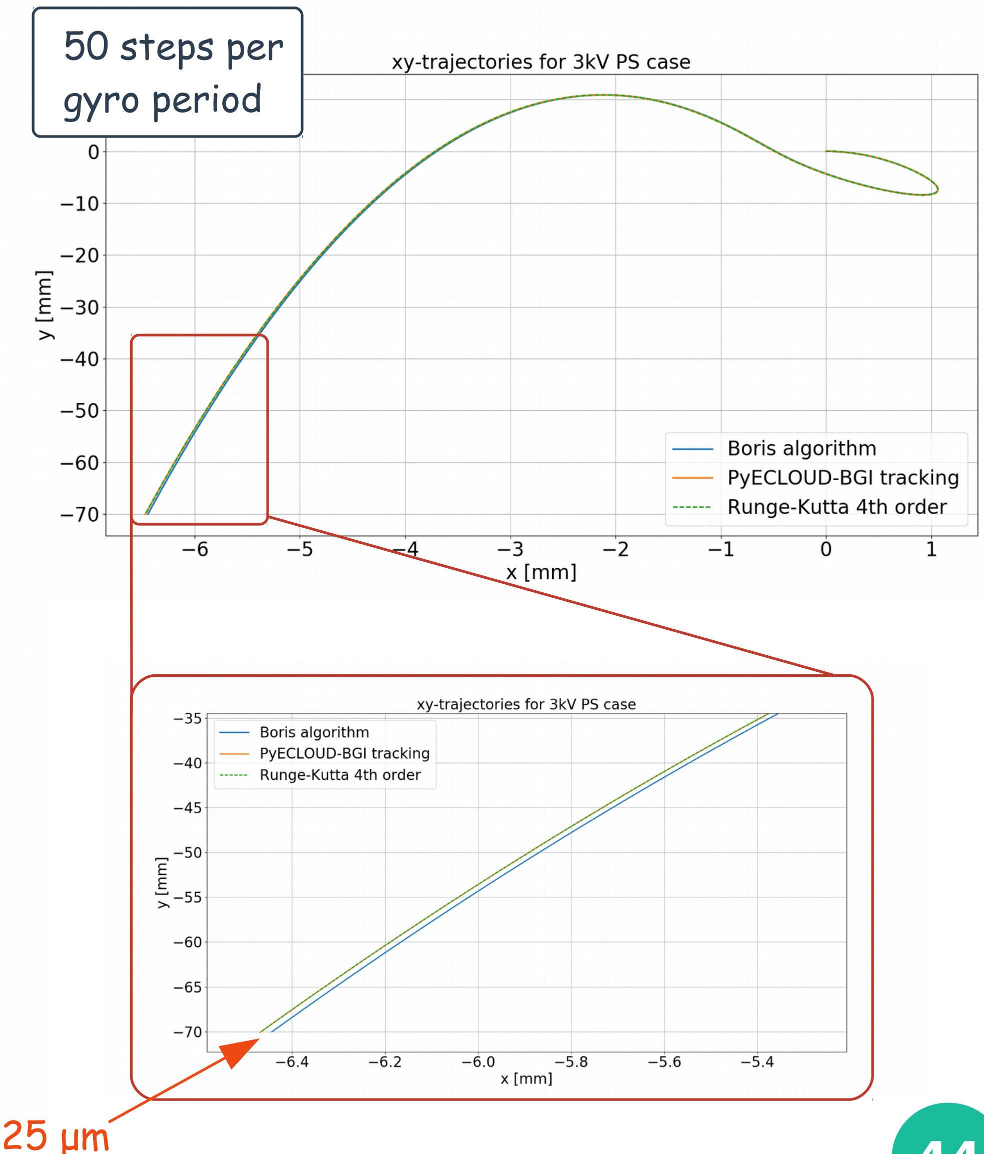
Deviation of 55 μm

Comparison of tracking algorithms

PS case – Trajectories in beam field

- Initial energy: 1 eV (→ from DDCS)
- Particle generated at t=0, z=0; beam has offset $z = 4\sigma_z$
- Magnetic field: 0 T
- Time step $\Delta t \approx 3.57$ ps

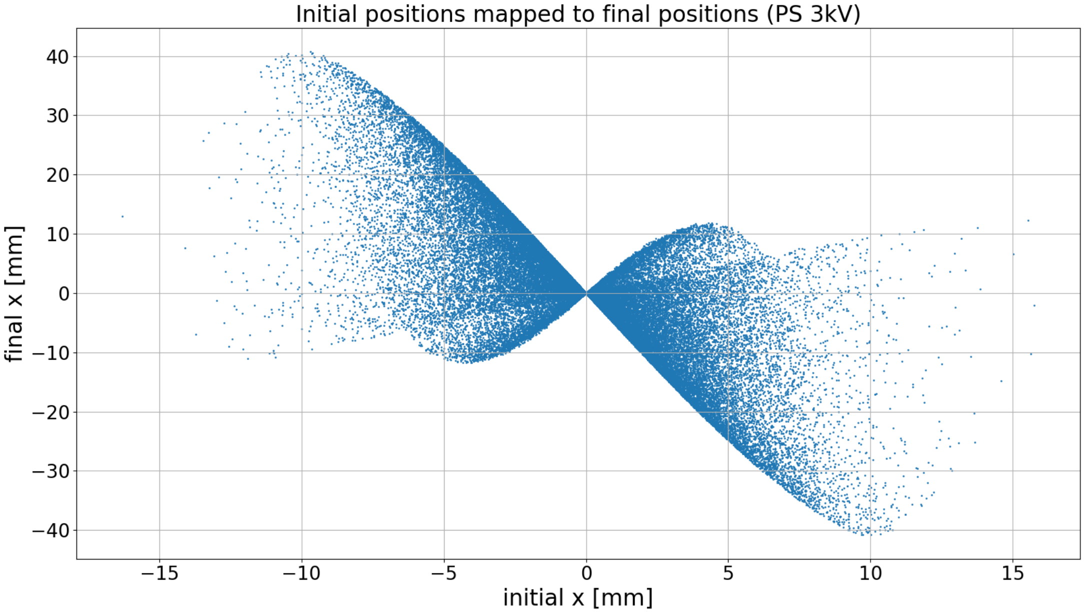


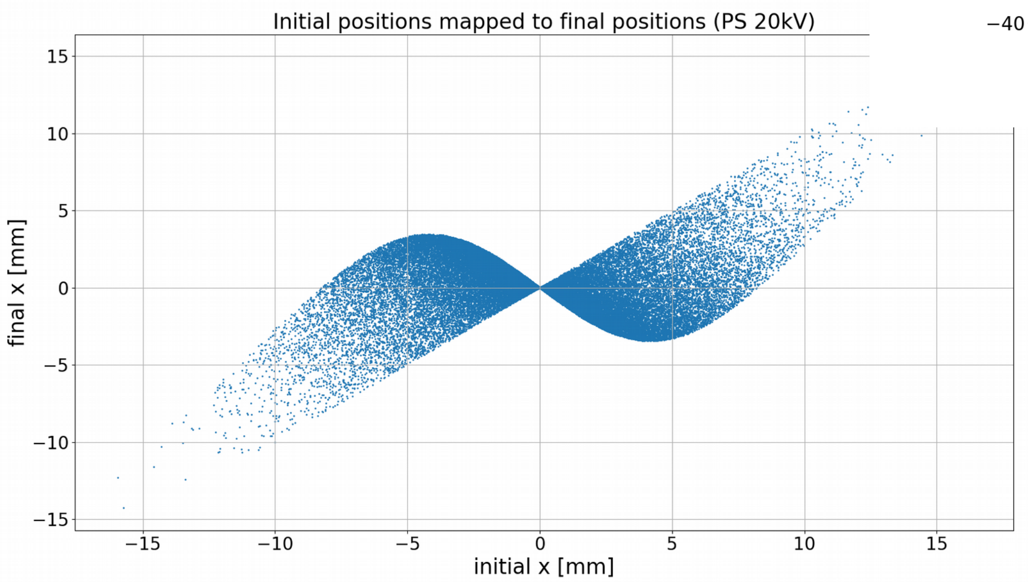


PS Case – Profile distortion

<u>Mapping of initial to final x-positions</u>

- The plots show that for low extraction fields (3kV) electrons actually move to the other "side" of the distribution





- For larger extraction fields (20kV) the electrons are still attracted towards the center of the distribution and thus accumulate in this region