Numerical modeling of fast beam ion instabilities

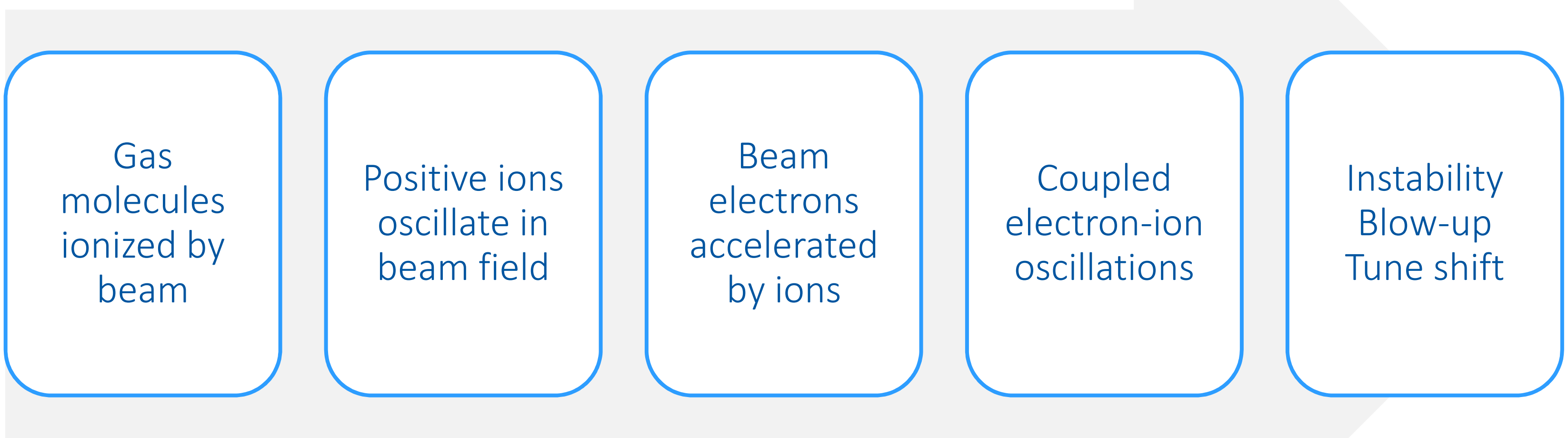
L. Mether, G. Iadarola, G. Rumolo

HB2016, 3-8 July, Malmö

Outline

- Introduction
- Simulation outline and tool development
- Application to CLIC main damping ring
- Simulation challenges and prospects
- Summary & outlook

Fast beam ion instability

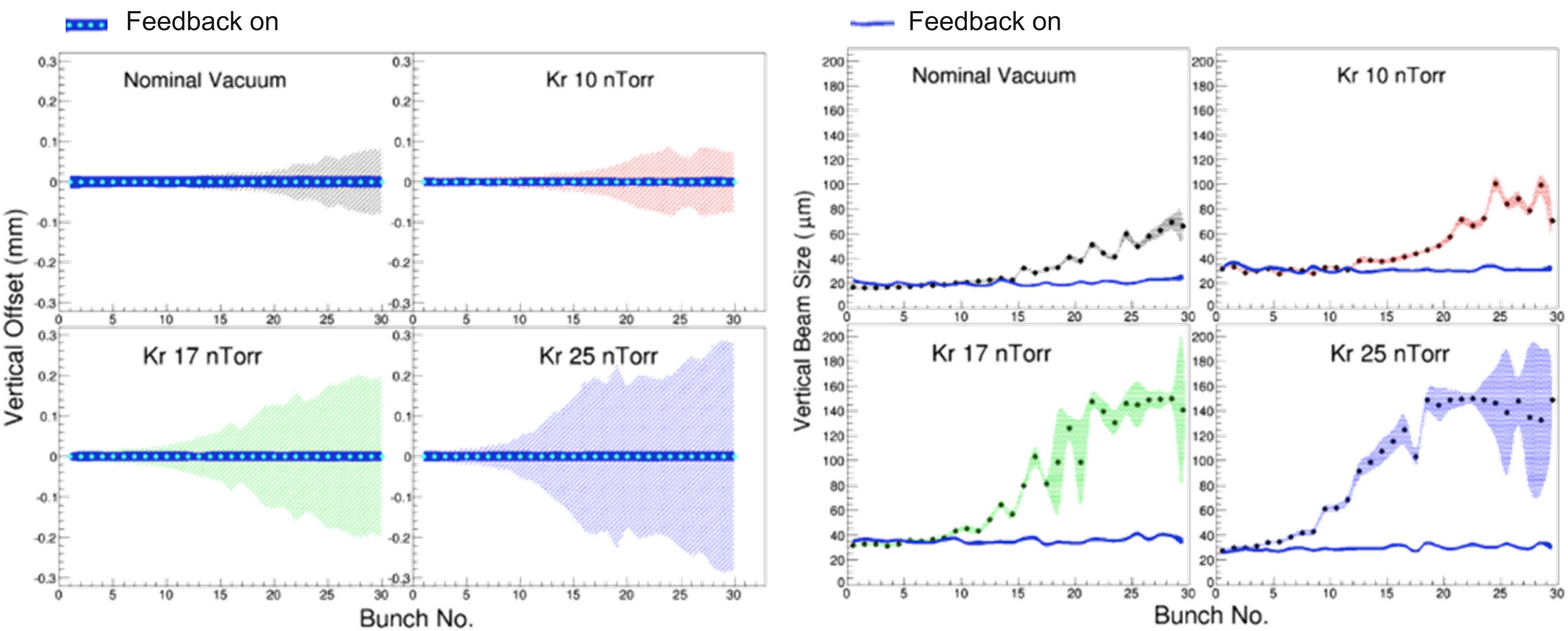


- For bunch train followed by large gap, ions build up during one train passage
  - Fast beam ion instability (FBII)

- E-cloud-like instability
  - Ion density increases for every bunch passage → effect strongest at tail of train

Observations

- Observed in several machines under vacuum degradation
- Measurements at CESR-TA (Apr 2014)
  - Varying ion species, pressure, bunch charge, train structure, feedback etc.



A. Chatterjee *et al*. Phys. Rev. ST Accel. Beams 18 (2015) 6, 064402

Theory

- Based on linear approximation of Bassetti-Erskine formula
- Dynamics essentially depend on beam brightness
  - Velocity kick for ion with mass number A

$$k_{x,y}(x,y) = \frac{2N_b r_p c}{A} \frac{x,y}{(\sigma_x+\sigma_y)\sigma_{x,y}}$$

    - A = ion mass number, N_b= bunch intensity, r_p = classical proton radius

  - Trapping condition

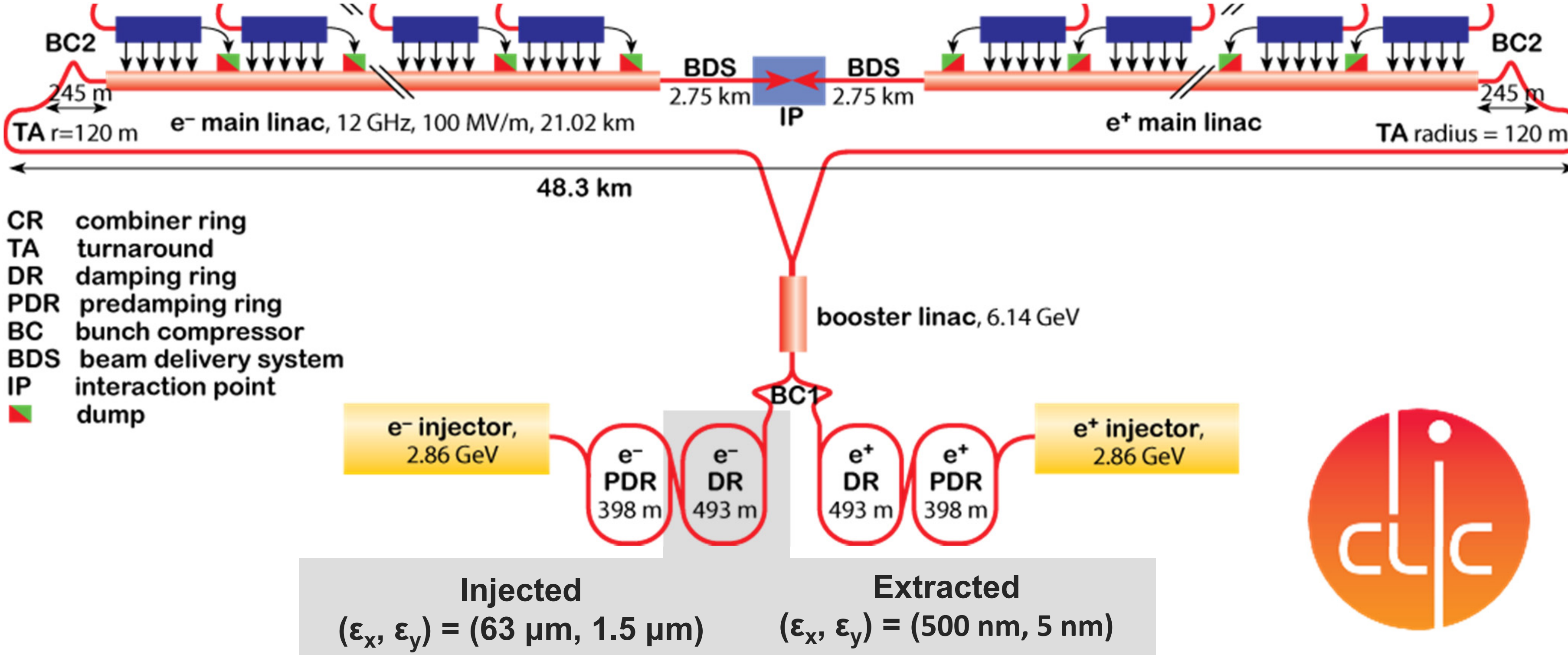
$$A > A_{\text{trap}} = \frac{N_b r_p T_b c}{2\left(\sigma_x + \sigma_y\right)\sigma_{x,y}}$$

    - T_b = bunch spacing

Fast beam-ion instability. I. Linear theory and simulations
Raubenheimer *et al.* Phys. Rev. E 52, 5, 5487

CLIC accelerator complex

Parameters	Value
Bunch population	4 x 10^9
Bunches per train	312
Bunch spacing [ns]	0.5
Bunch length (rms) [mm]	1.6



Simulation studies

- Aim to estimate vacuum (and/or feedback) requirements imposed by FBII
- Simulated with strong-strong 2D macroparticle multi-bunch tracking code

- FASTION
  - Developed and used for studies of FBII in linear CLIC structures
  - Based on HEADTAIL for e-cloud
  - Development required to adapt to damping rings

- Several CERN beam dynamics codes re-designed recently
  - Electron cloud build-up: ECLOUD → PyECLOUD
  - Collective effects: HEADTAIL → PyHEADTAIL

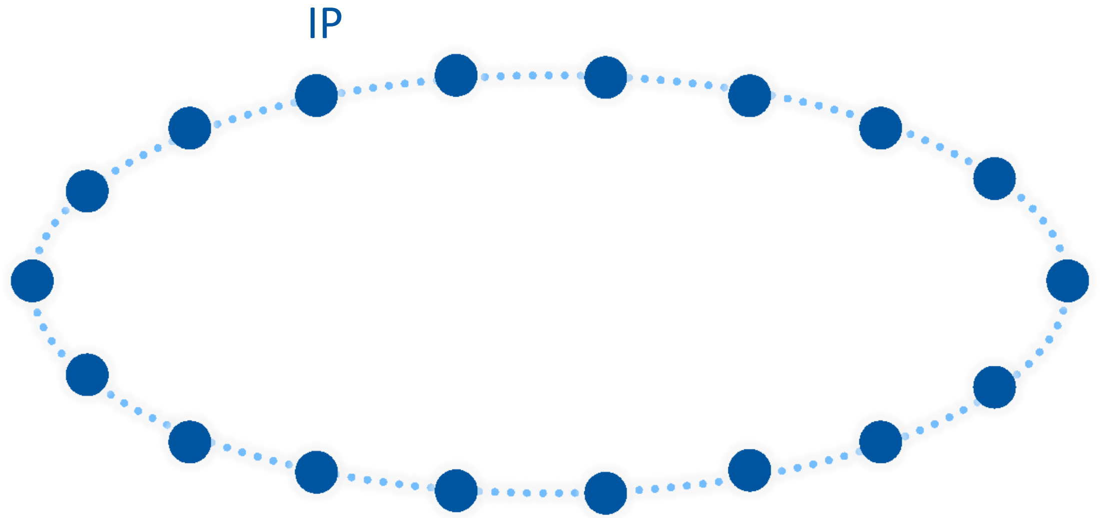
  (ECLOUD → PyECLOUD and HEADTAIL → PyHEADTAIL: coupled)

  - Aim to make codes more maintainable, flexible and user friendly

- Decision to incorporate FASTION functionality into PyECLOUD – PyHEADTAIL

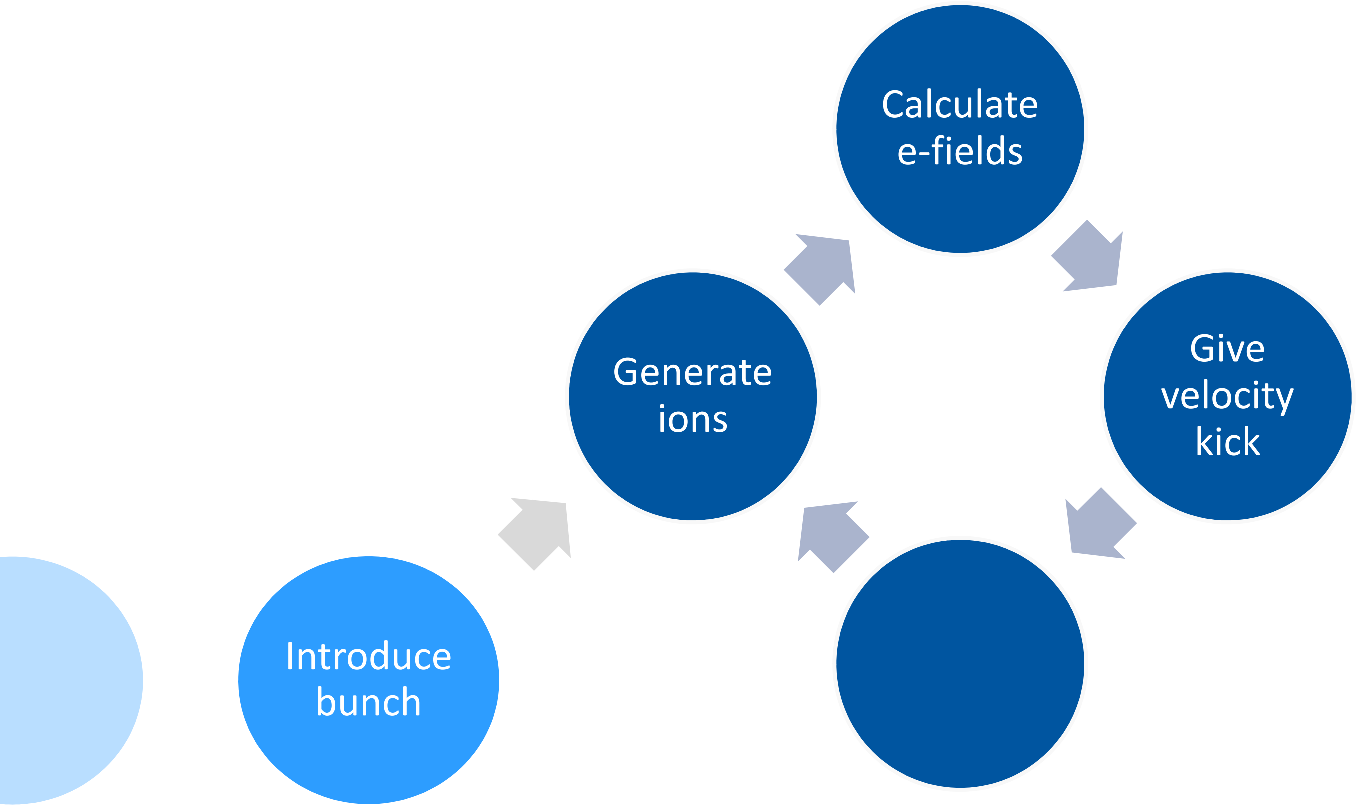
Simulation outline

- The machine lattice is divided into a number of interaction points (IP)
  - An electron bunch train is tracked through the lattice
  - In every IP, the beam-ion interaction is simulated



Simulation outline

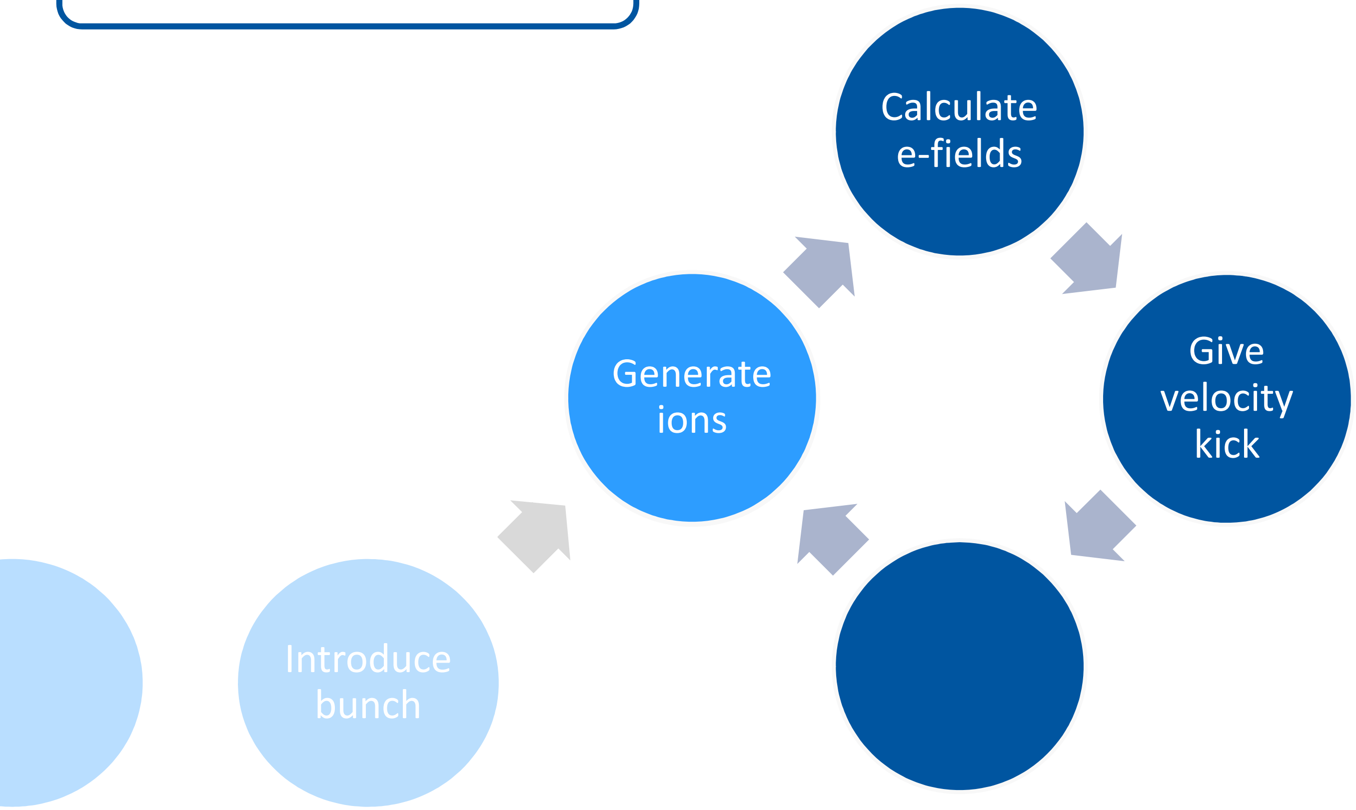
- In every interaction point, the beam is passed bunch by bunch



Simulation outline

➢ In every interaction point, the beam is passed bunch by bunch

Represent ions generated between two IP's

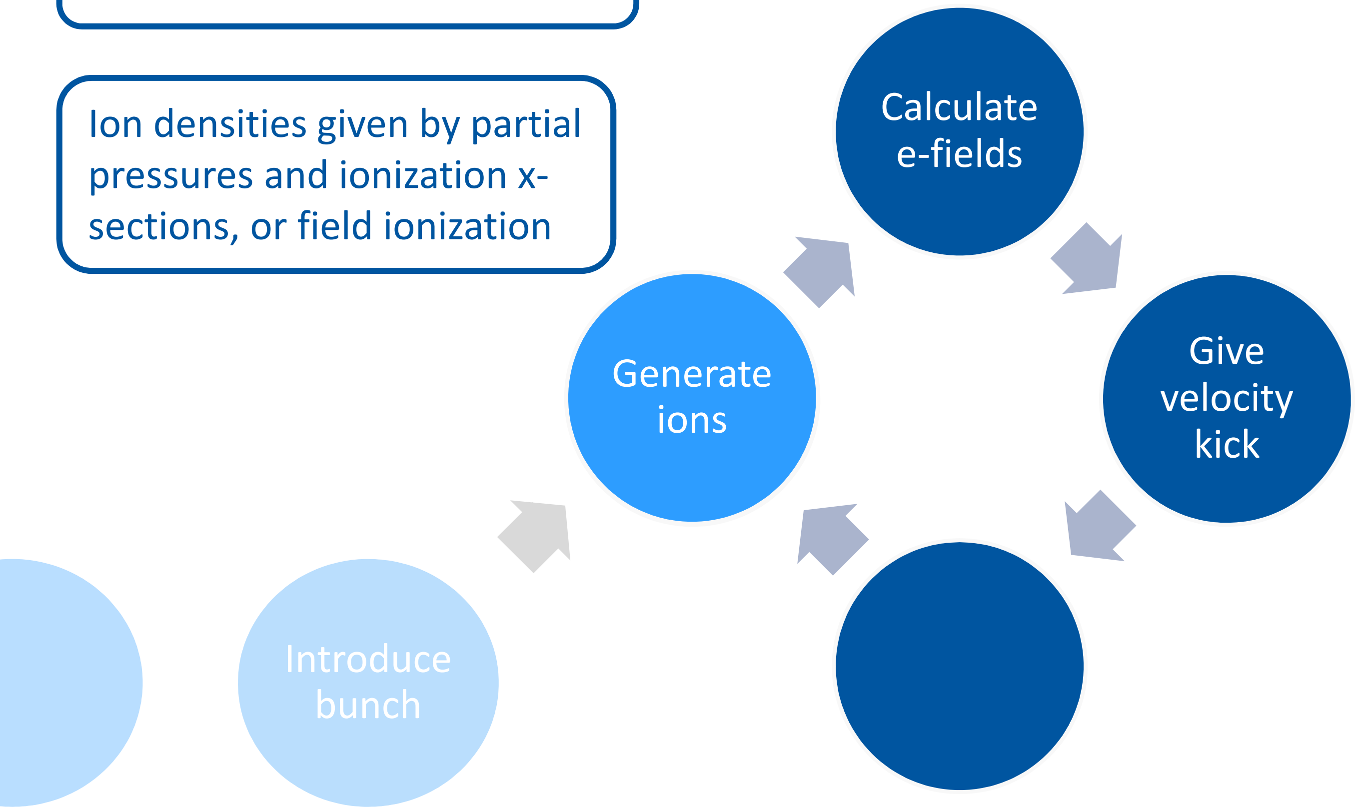


Simulation outline

- In every interaction point, the beam is passed bunch by bunch

Represent ions generated between two IP's

Ion densities given by partial pressures and ionization x-sections, or field ionization



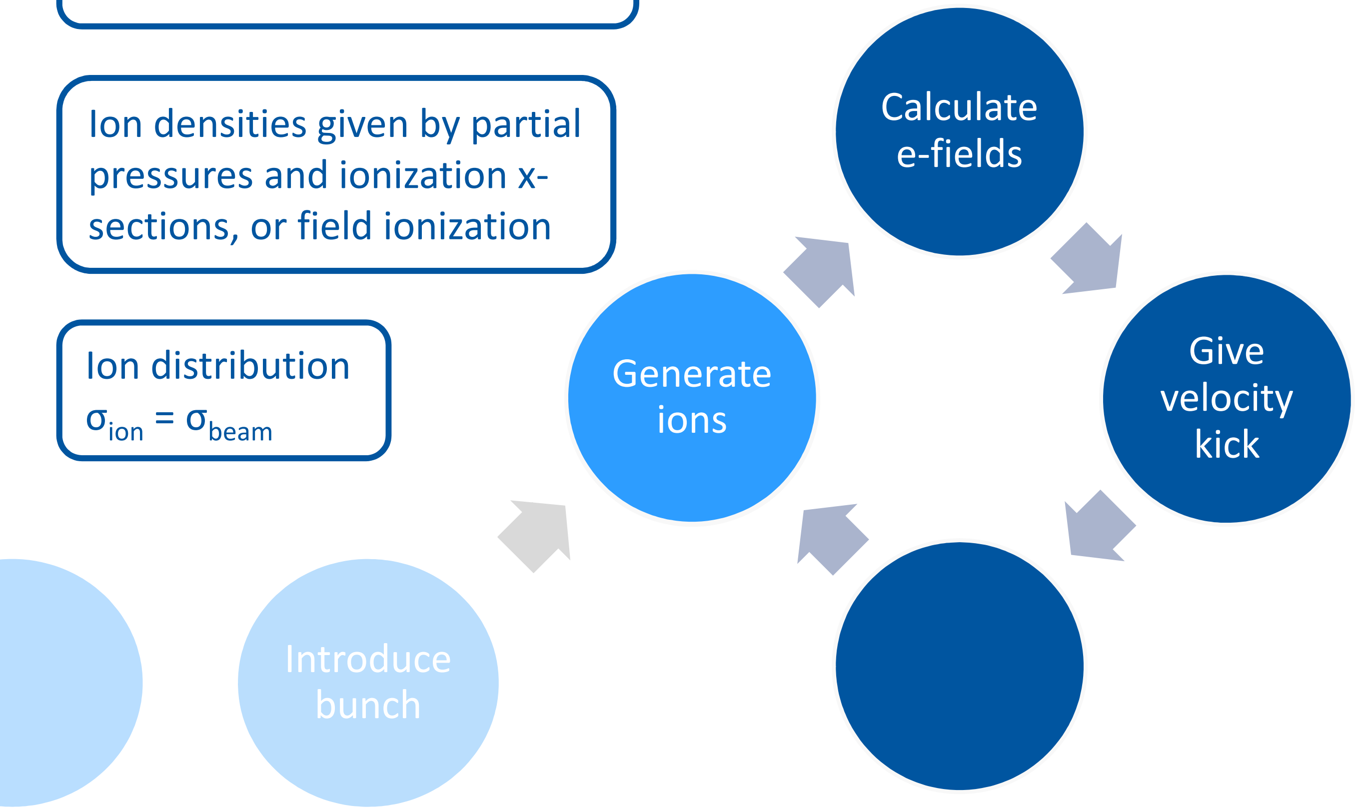
Simulation outline

- In every interaction point, the beam is passed bunch by bunch

Represent ions generated between two IP's

Ion densities given by partial pressures and ionization x-sections, or field ionization

Ion distribution
$\sigma_{ion} = \sigma_{beam}$



Simulation outline

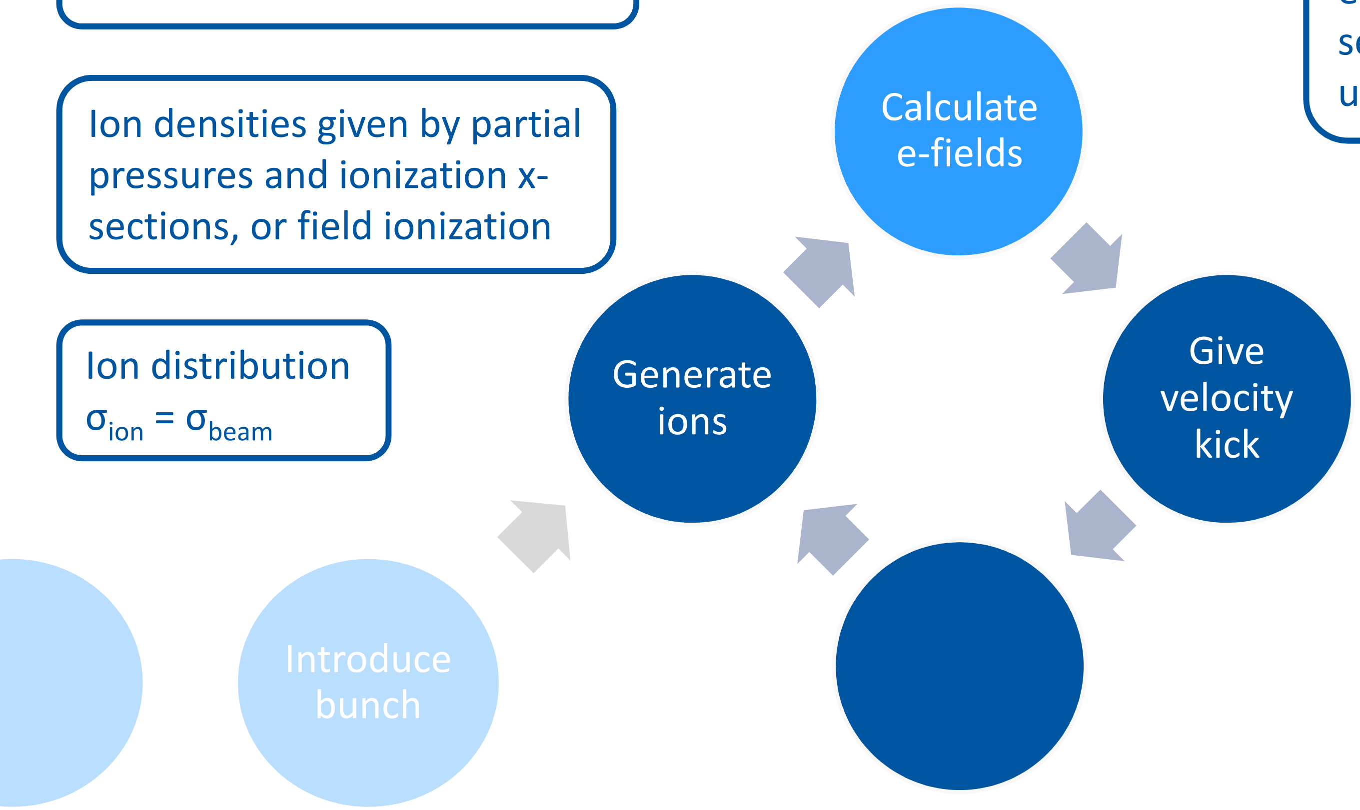
- In every interaction point, the beam is passed bunch by bunch

Represent ions generated between two IP's

Ion densities given by partial pressures and ionization x-sections, or field ionization

Ion distribution
$\sigma_{ion} = \sigma_{beam}$

Electric fields of ions and electrons calculated separately on same grid using FFT open boundary



Simulation outline

- In every interaction point, the beam is passed bunch by bunch

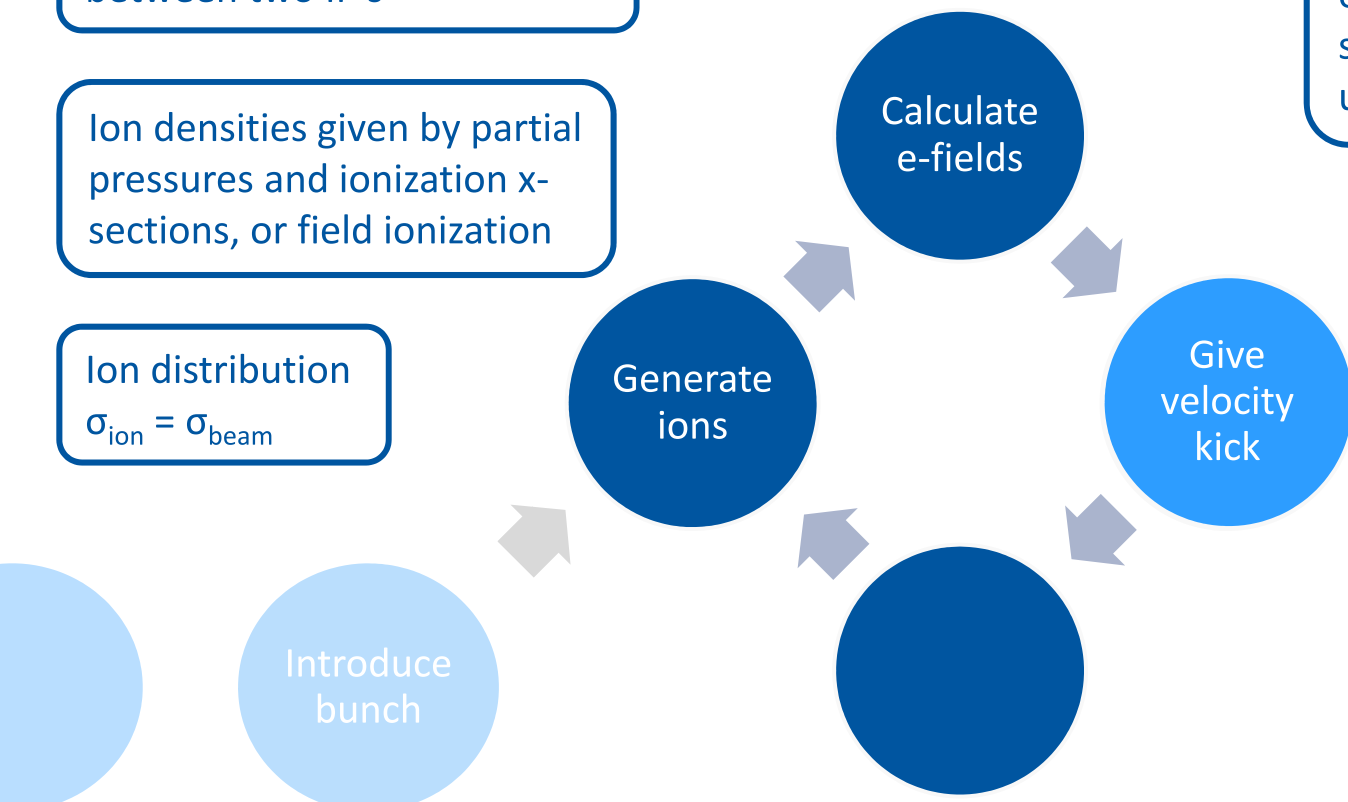
Represent ions generated between two IP's

Ion densities given by partial pressures and ionization x-sections, or field ionization

Ion distribution
$\sigma_{ion} = \sigma_{beam}$

Electric fields of ions and electrons calculated separately on same grid using FFT open boundary

Velocity kicks applied according to e-field of opposite particles



Simulation outline

- In every interaction point, the beam is passed bunch by bunch

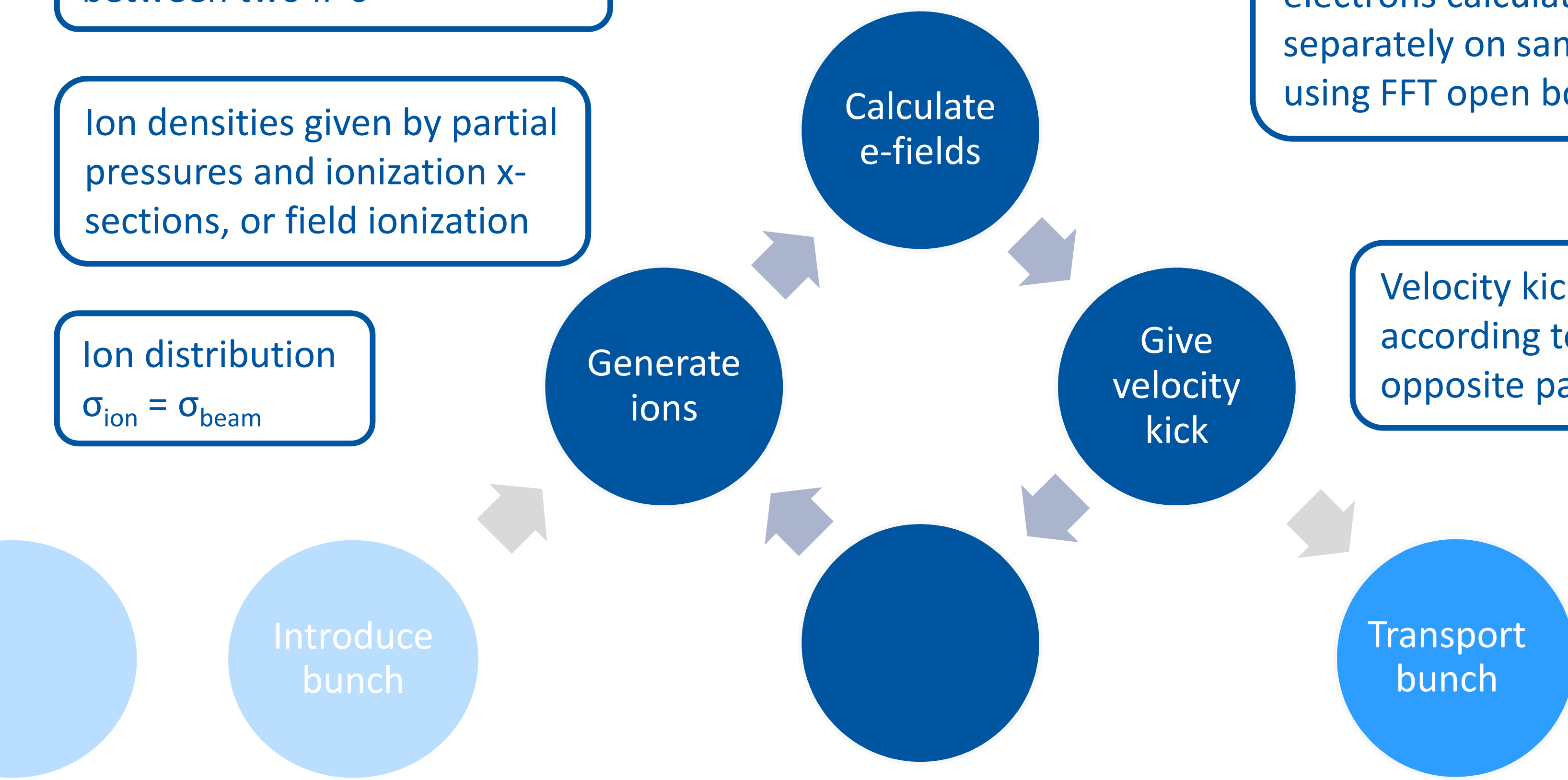
Represent ions generated between two IP's

Ion densities given by partial pressures and ionization x-sections, or field ionization

Ion distribution
$\sigma_{ion} = \sigma_{beam}$

Electric fields of ions and electrons calculated separately on same grid using FFT open boundary

Velocity kicks applied according to e-field of opposite particles



Simulation outline

- In every interaction point, the beam is passed bunch by bunch

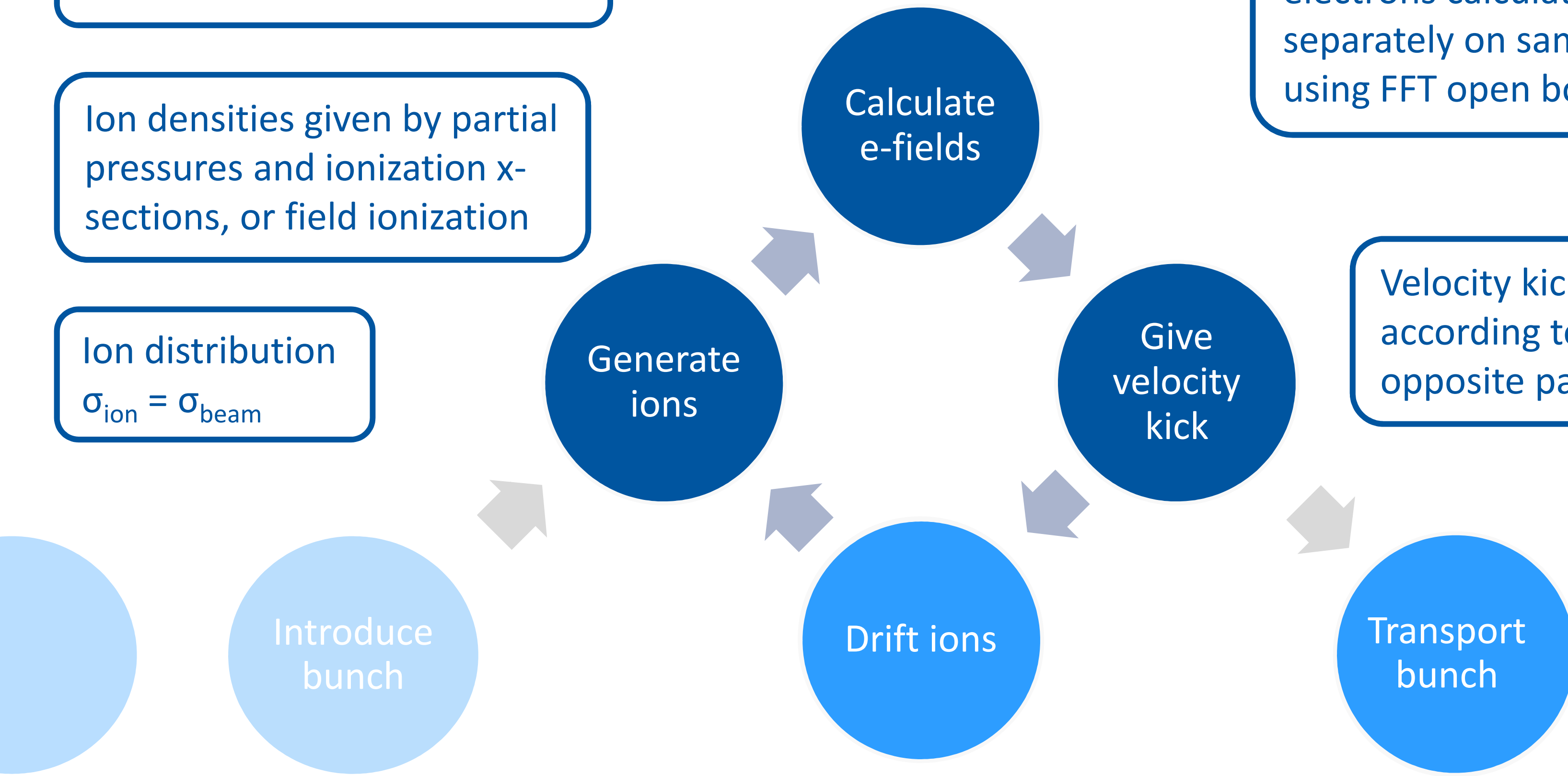
Represent ions generated between two IP's

Ion densities given by partial pressures and ionization x-sections, or field ionization

Ion distribution
$\sigma_{ion} = \sigma_{beam}$

Electric fields of ions and electrons calculated separately on same grid using FFT open boundary

Velocity kicks applied according to e-field of opposite particles



Simulation outline

➢ In every interaction point, the beam is passed bunch by bunch

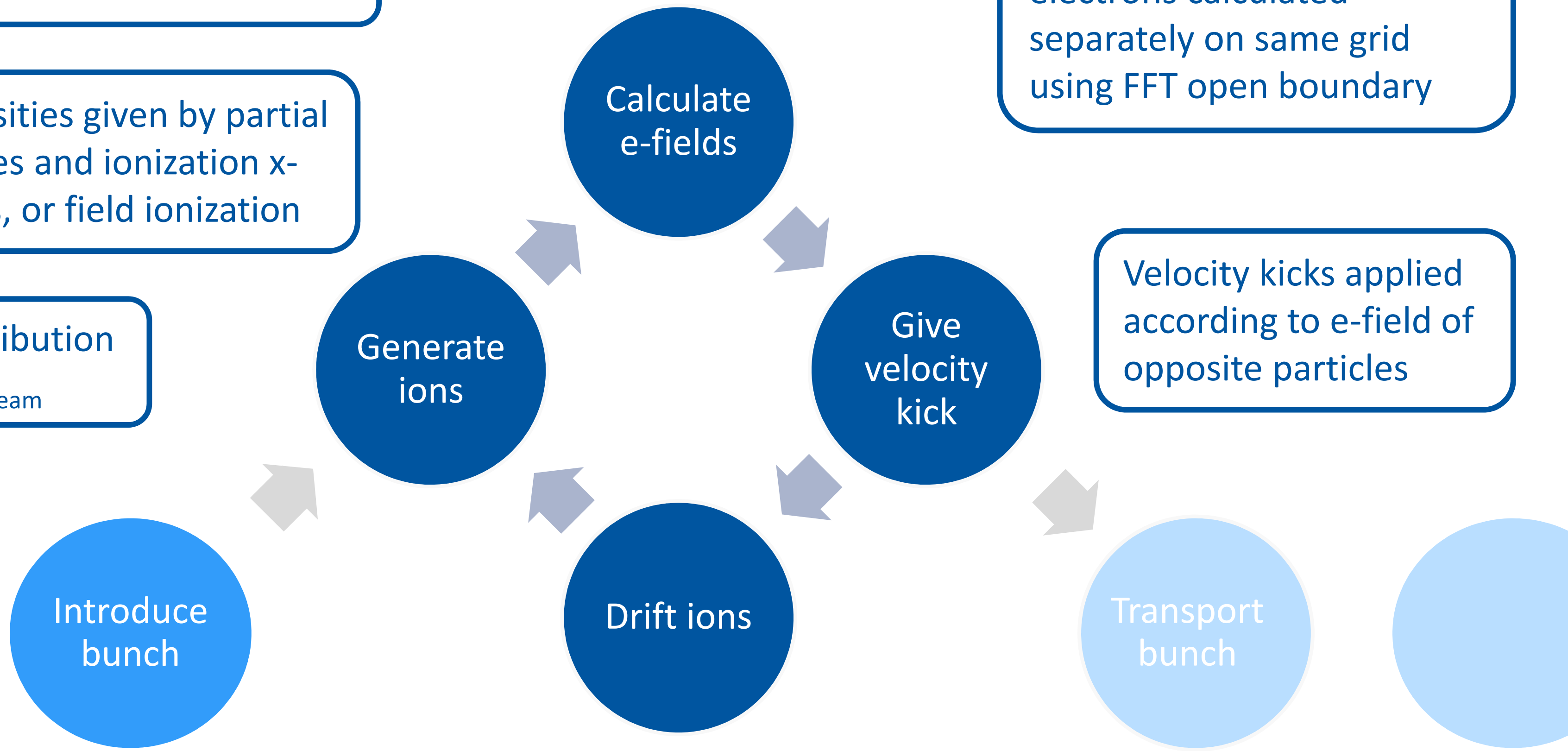
Represent ions generated between two IP's

Ion densities given by partial pressures and ionization x-sections, or field ionization

Ion distribution
$\sigma_{ion} = \sigma_{beam}$

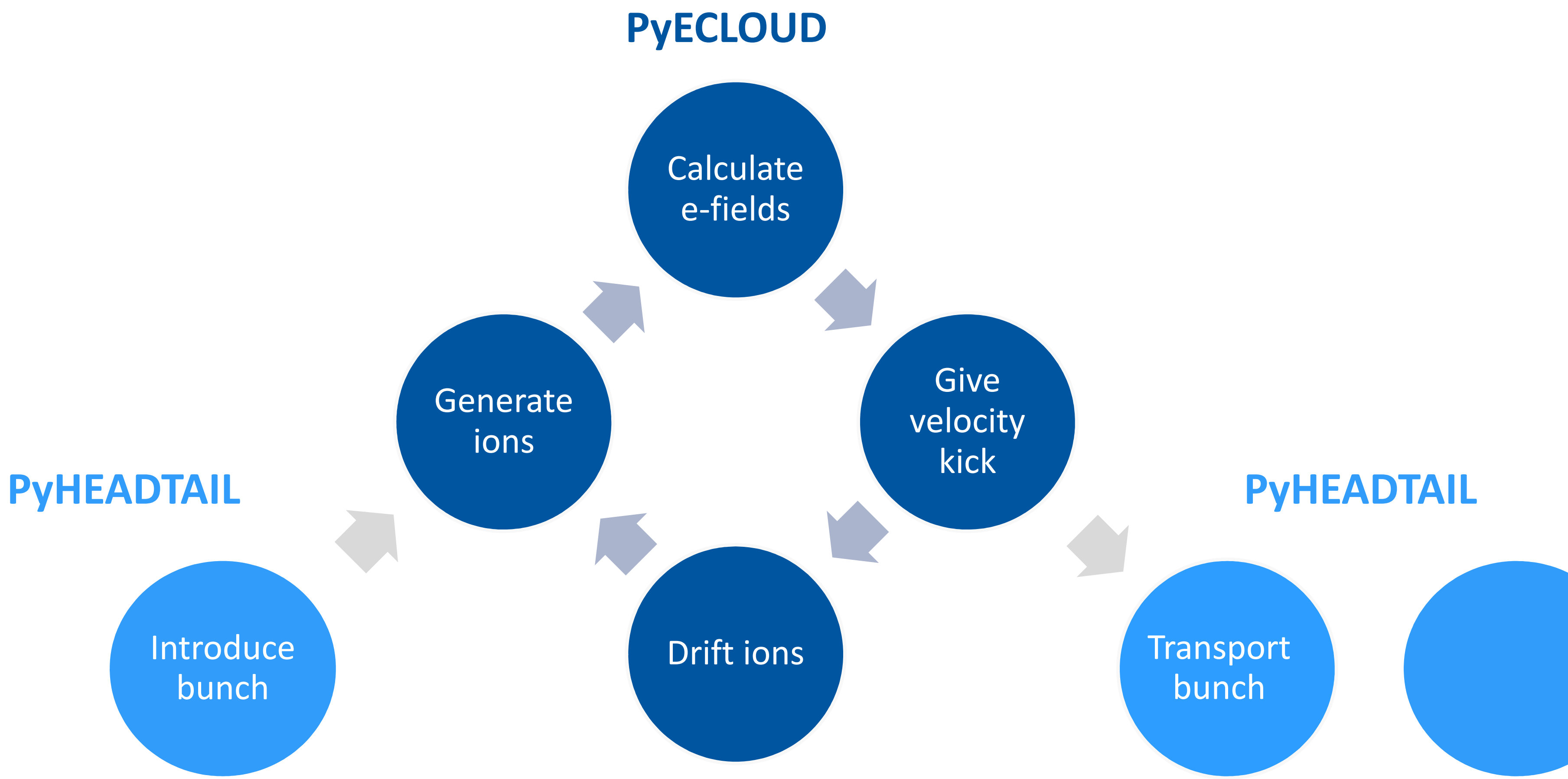
Electric fields of ions and electrons calculated separately on same grid using FFT open boundary

Velocity kicks applied according to e-field of opposite particles



Simulation outline

- Implemetation in PyECLOUD and PyHEADTAIL

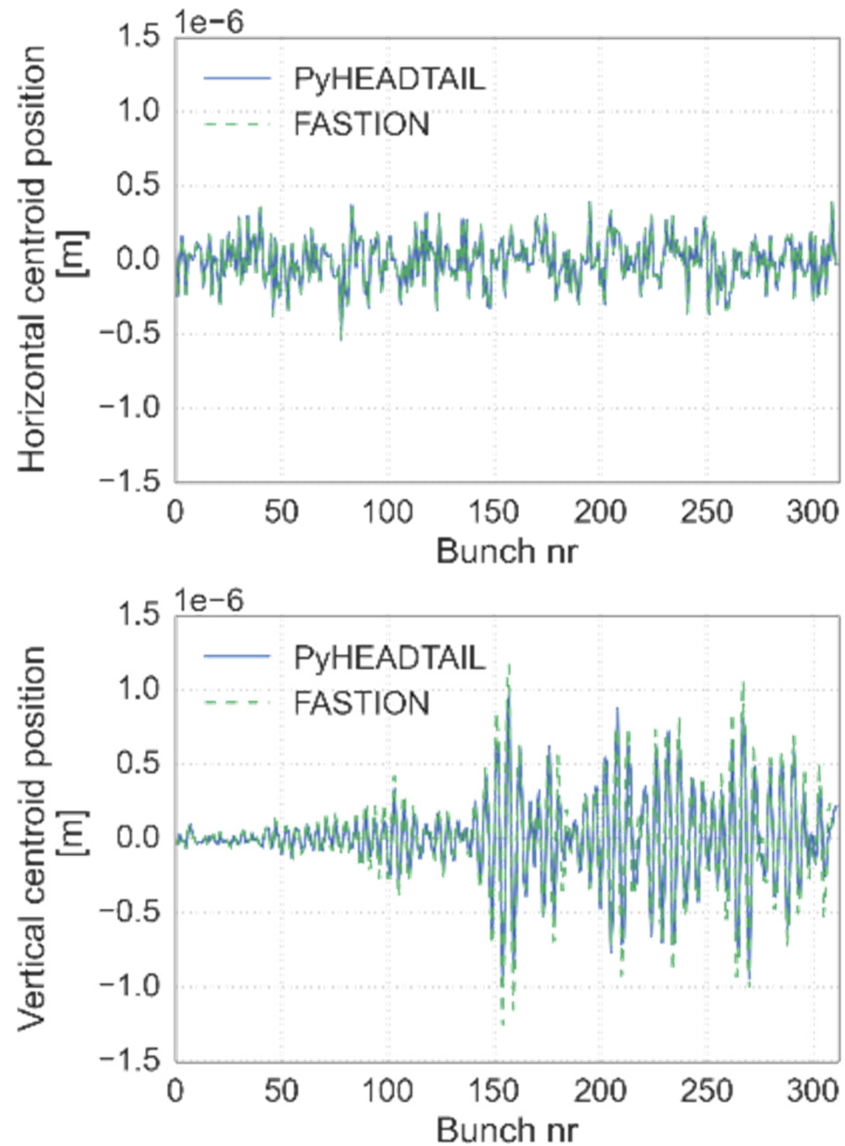


Implementation

- Generalization to arbitrary charge and mass in PyECLOUD and PyHEADTAIL
- Extension of PyECLOUD gas ionization routines
  - Multiple ion species, field ionization
- Ion boundary conditions (perfect absorber)
- Modification of PyPIC FFT solver method
  - Rectangular (non-square) grid cells, useful due to the very flat beams
- Single kick interaction
- Implementation of multi-bunch in PyHEADTAIL
  - Create and track multi-bunch beam, “slice” into bunches for interaction

Application to CLIC damping ring

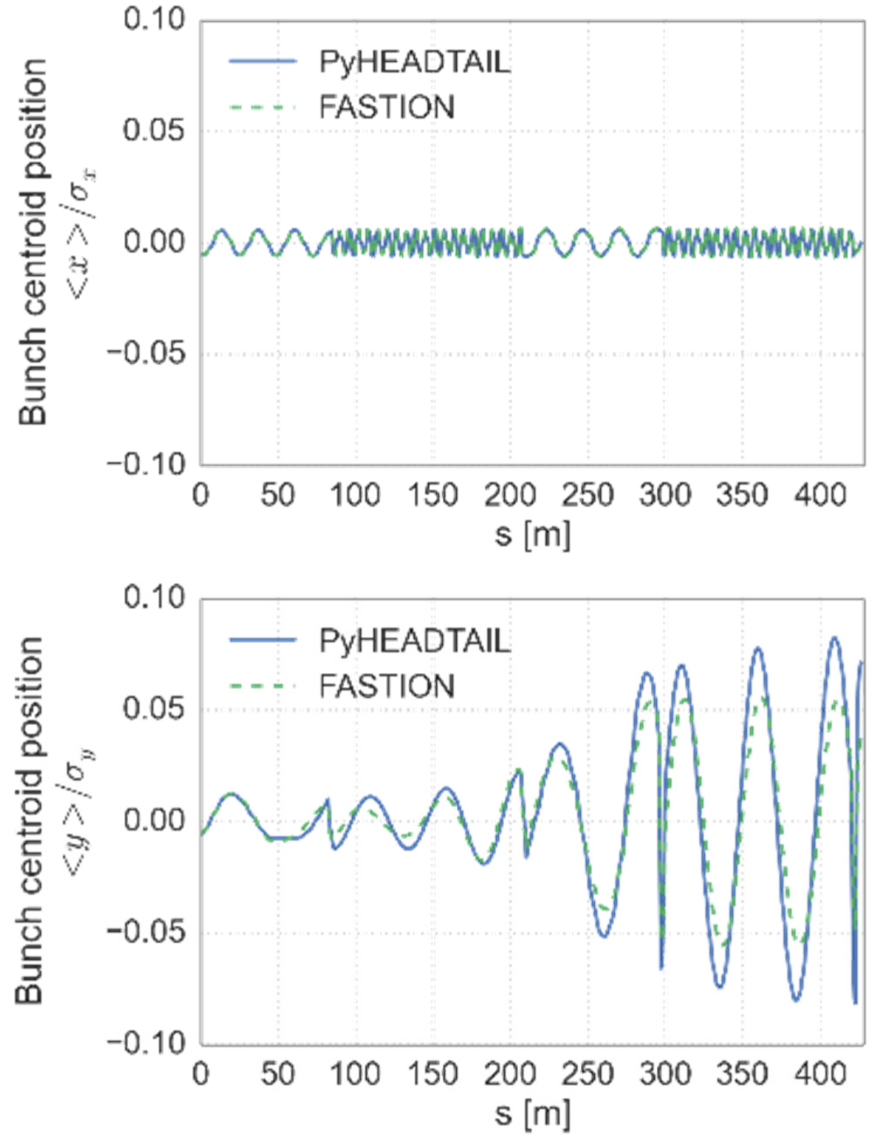
- Benchmark study I
  - Bunch train initialized identically in FASTION and PyEC-PyHT
  - Machine lattice divided in 677 interaction points ~ 60 cm long
  - Residual gas: water, A = 18
  - Pressure 20 nTorr
- Track over 1 turn



- Bunch train centroids after 1 turn
- Unstable motion in vertical plane, as expected

Application to CLIC damping ring

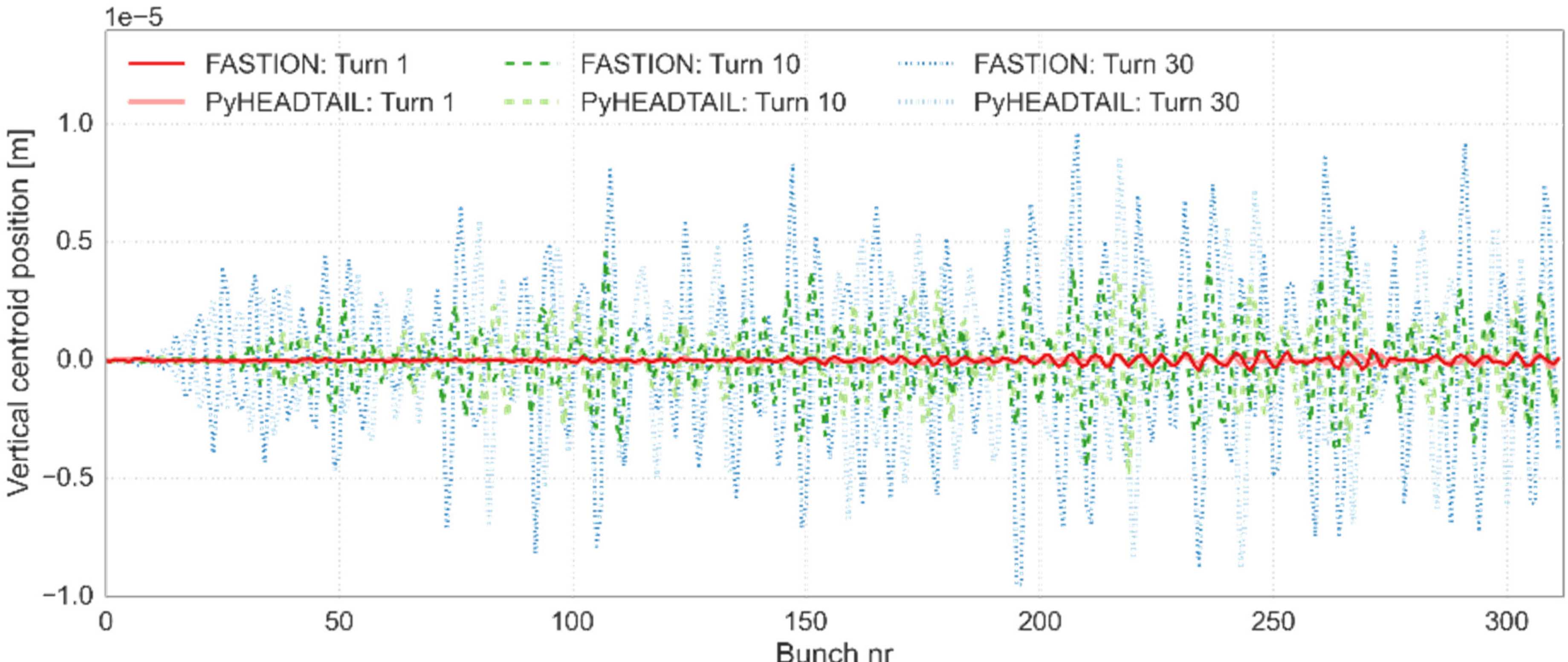
- Benchmark study I
  - Bunch train initialized identically in FASTION and PyEC-PyHT
  - Machine lattice divided in 677 interaction points ~ 60 cm long
  - Residual gas: water, A = 18
  - Pressure 20 nTorr
- Track over 1 turn



- Centroid of last bunch along turn
- Good agreement between FASTION and PyECLOUD-PyHEADTAIL

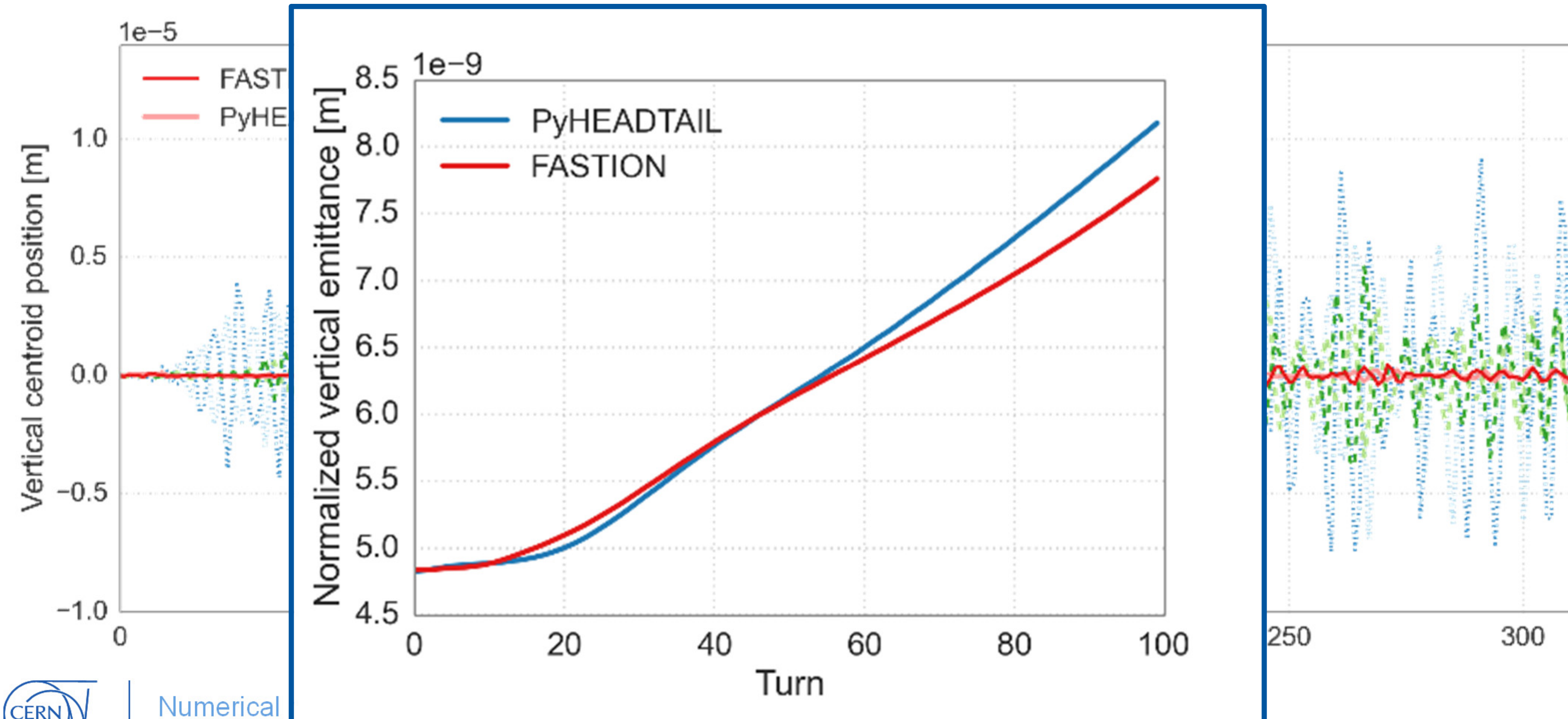
Application to CLIC damping ring

- Benchmark study II
  - Bunch train initialized with different random seeds in FASTION and PyEC-PyHT
  - Residual gas: water, A = 18, P = 10 nTorr
- Track over 100 turn, snapshots of train after 1, 10 and 30 turns



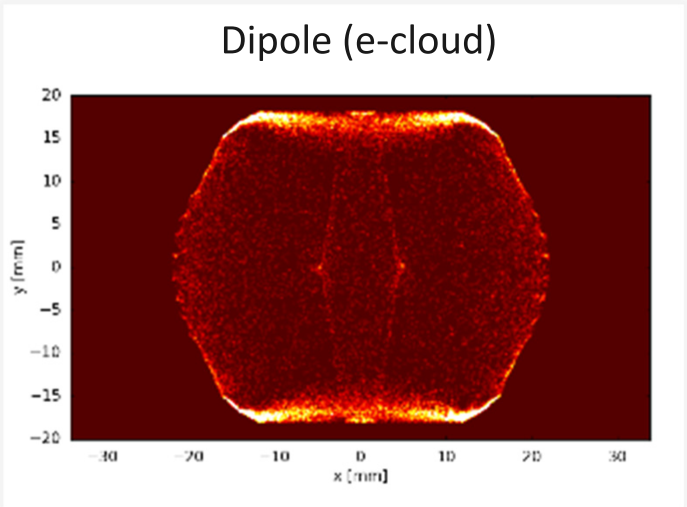
Application to CLIC damping ring

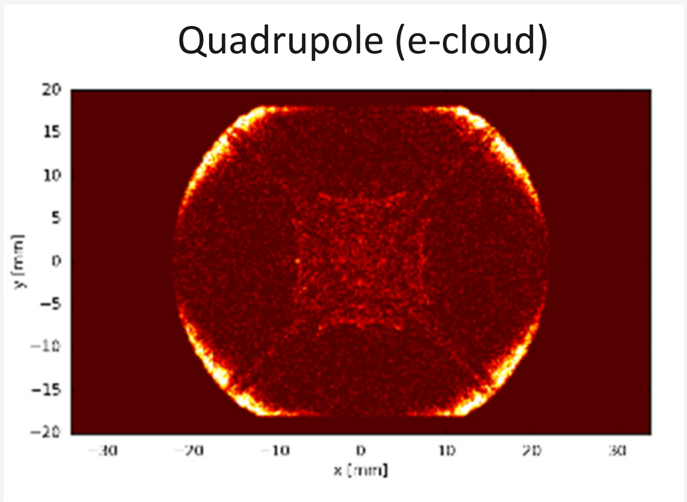
- Benchmark study II
  - Bunch train initialized with different random seeds in FASTION and PyEC-PyHT
  - Residual gas: water, A = 18, P = 10 nTorr
- Track over 100 turn, vertical emittance growth of last bunch



Simulation tool status

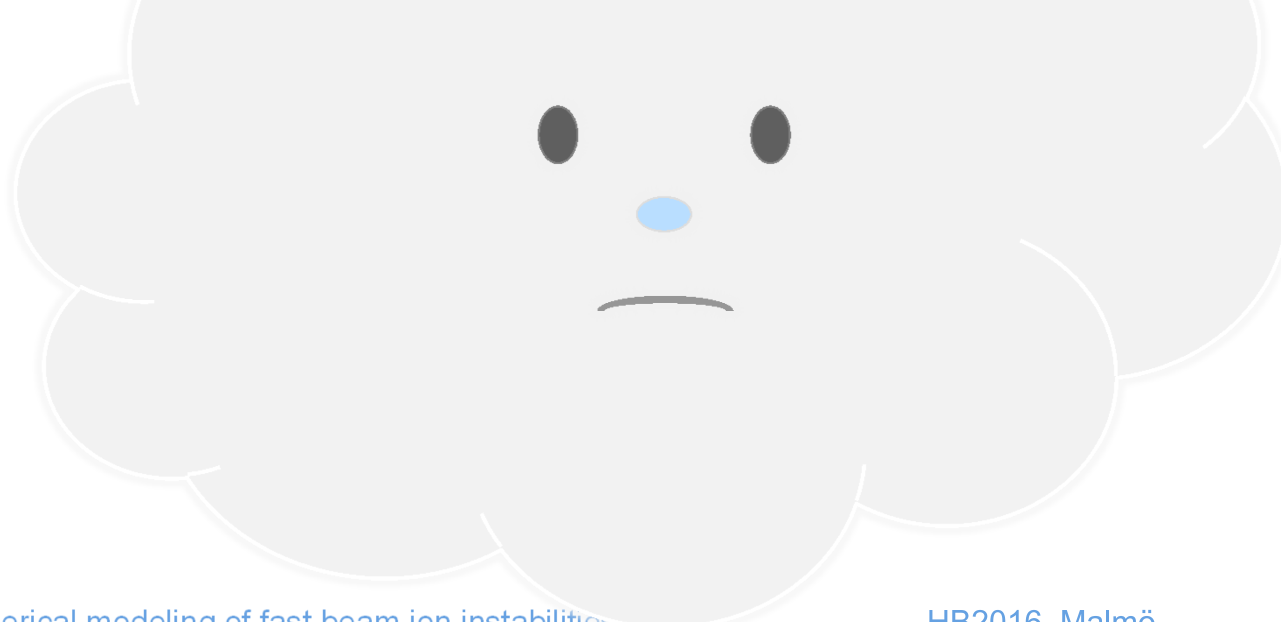
- Basic simulation scenario agrees with FASTION
- Ready to test new features available in PyECLOUD and PyHEADTAIL
  - Ion self space charge
  - PIC solvers with boundary for complex beam chamber profiles
  - Dipole and quadrupole magnetic fields on ion motion
  - Bunch slices
  - Synchrotron motion, chromaticity, transverse feedback





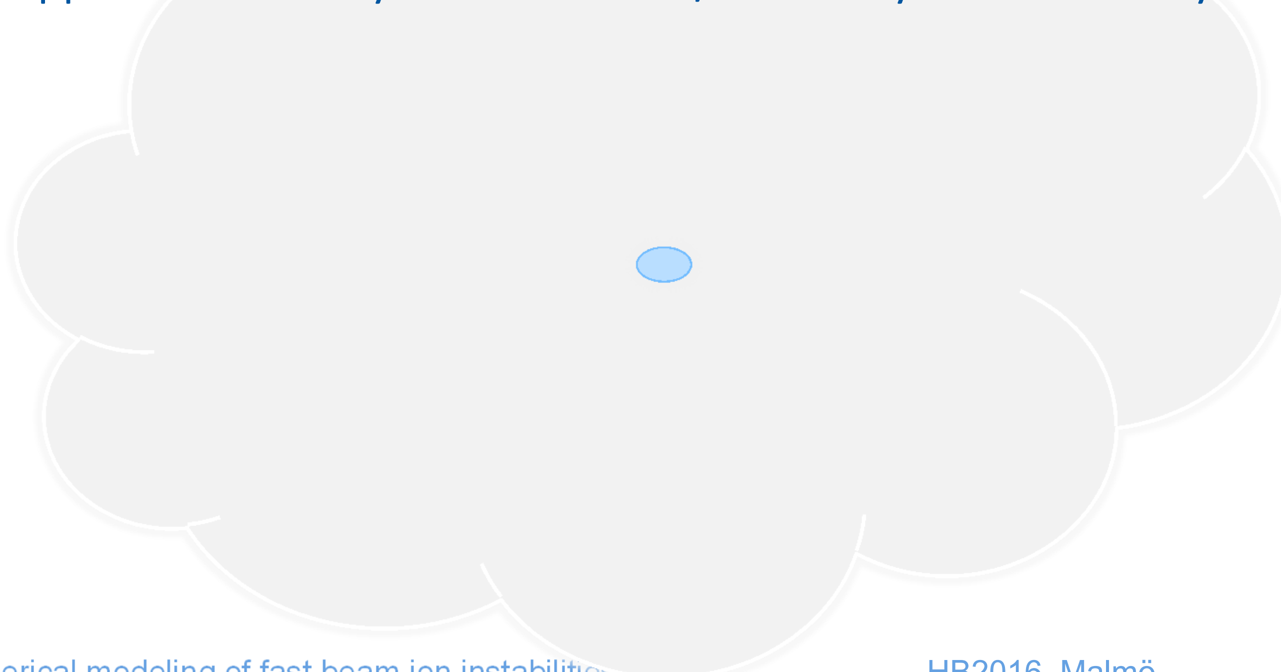
Challenges

- Resolution
  - Simulating two-stream instabilities generally challenging: big cloud – small beam
  - Especially for lepton machines, with tiny beams
  - For FBII, variations in electric field at slightly different locations inside beam are an important ingredient in exciting the instability
  - Simply increasing the number of PIC grid cells quickly leads to unacceptably long execution times, and eventually memory issues



Challenges

- Resolution
  - Solution: multiple nested grids
  - Dual-grid method with fine grid around beam, coarse grid for cloud in FASTION
  - Multigrid method, using modular structure, under development in PyPIC
    - Input number target grid size in beam & coarsest grid size → N grids
    - Applicable with any solver method / boundary condition in PyPIC

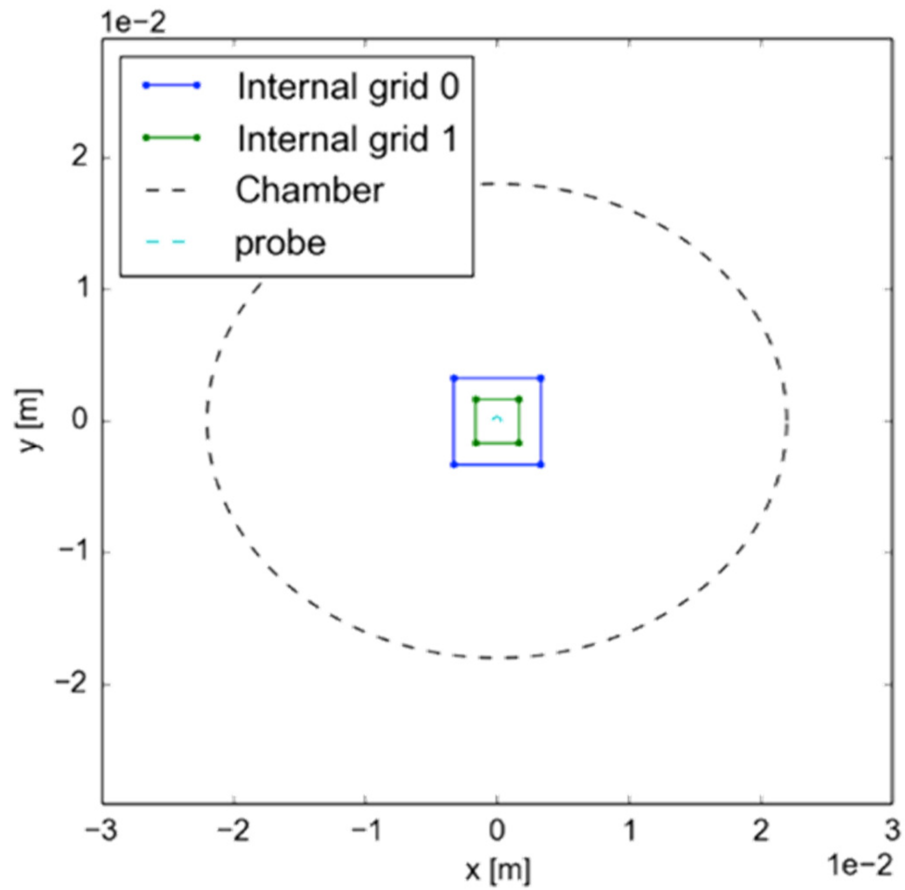


Multigrid method

with E. Belli

- Example
  - Compare single grid vs. multigrid with 3 grids
  - Reference from Bassetti-Erskine
  - Similar execution times
  - Circular beam

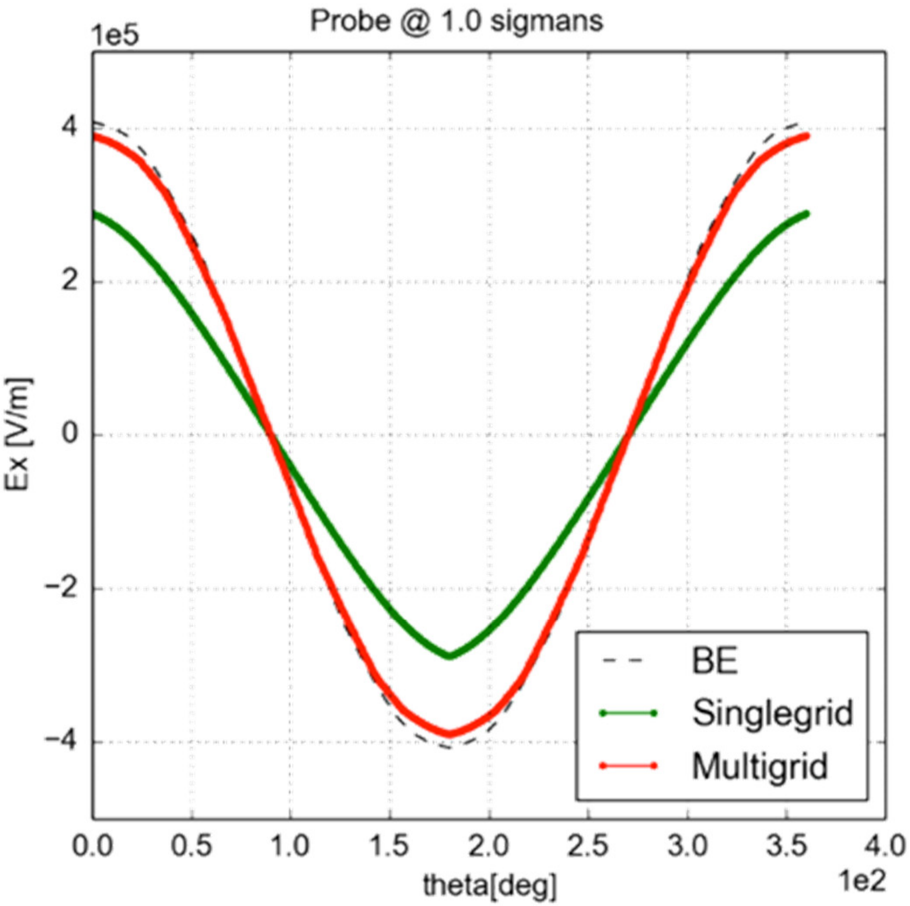
$$\sigma_x = 3.30\text{e-}04 \text{ [m]}$$
$$\sigma_y = 3.30\text{e-}04 \text{ [m]}$$
$$\Delta h_{single} = 3.95\text{e-}04 \text{ [m]}$$
$$\Delta h_{multi} = 1.32\text{e-}04 \text{ [m]}$$
$$\Delta h_{BE} = 9.89\text{e-}05 \text{ [m]}$$

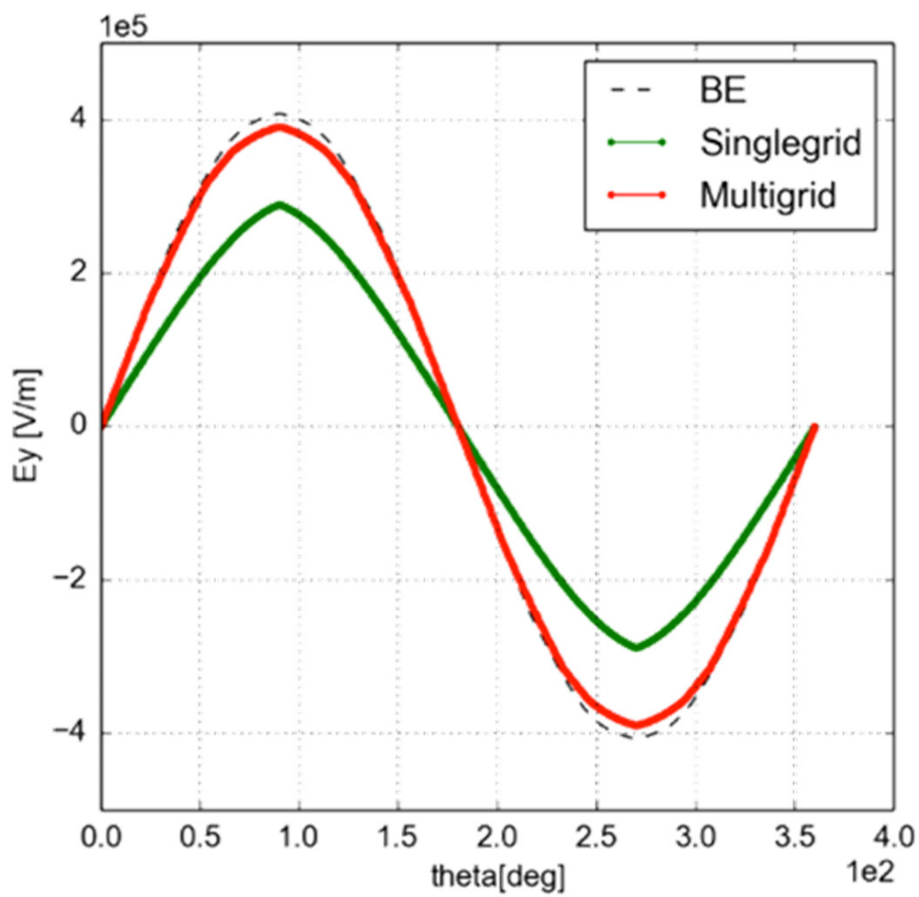


Multigrid method

with E. Belli

- Example
  - Electric fields at 1 sigma

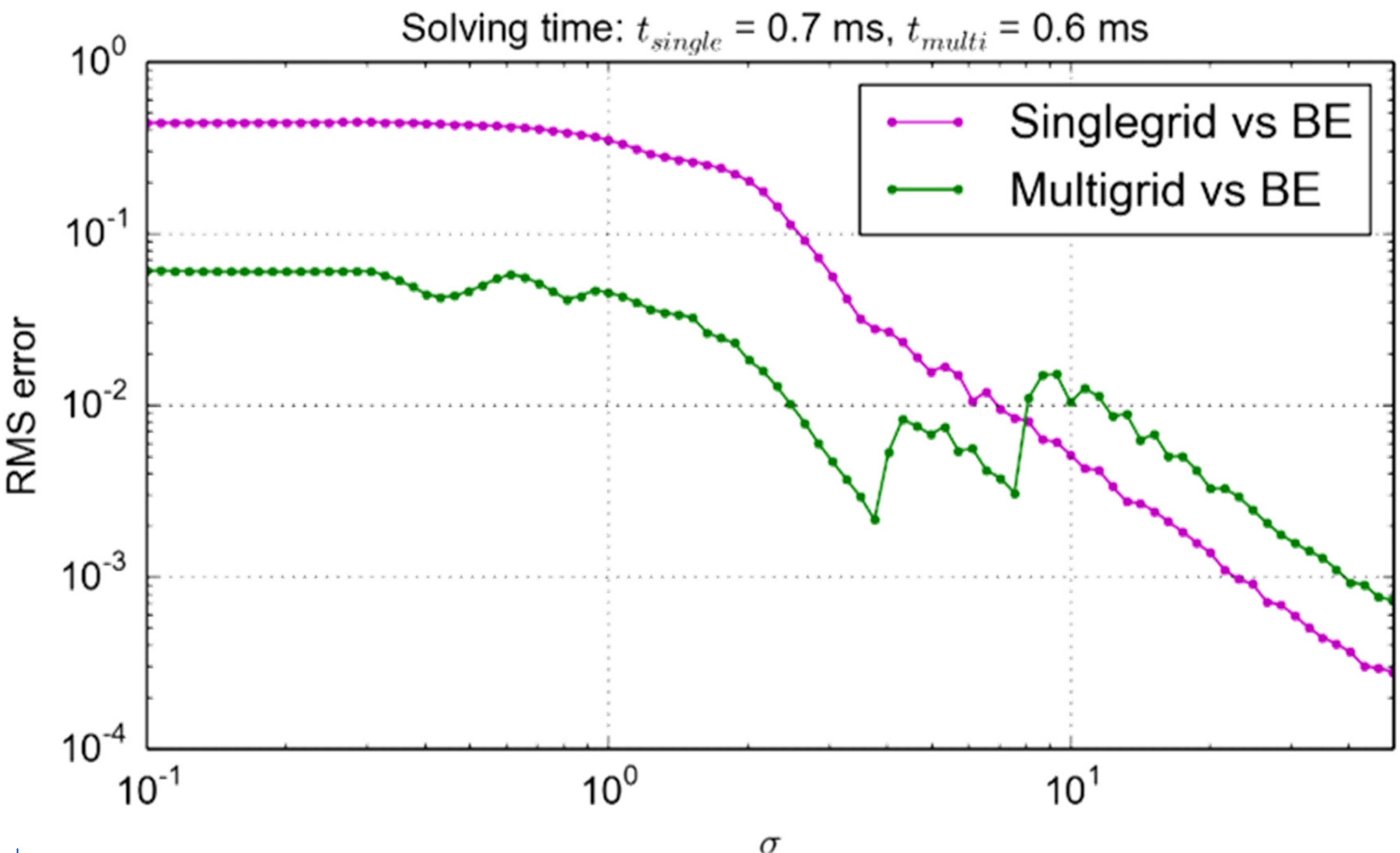




Multigrid method

with E. Belli

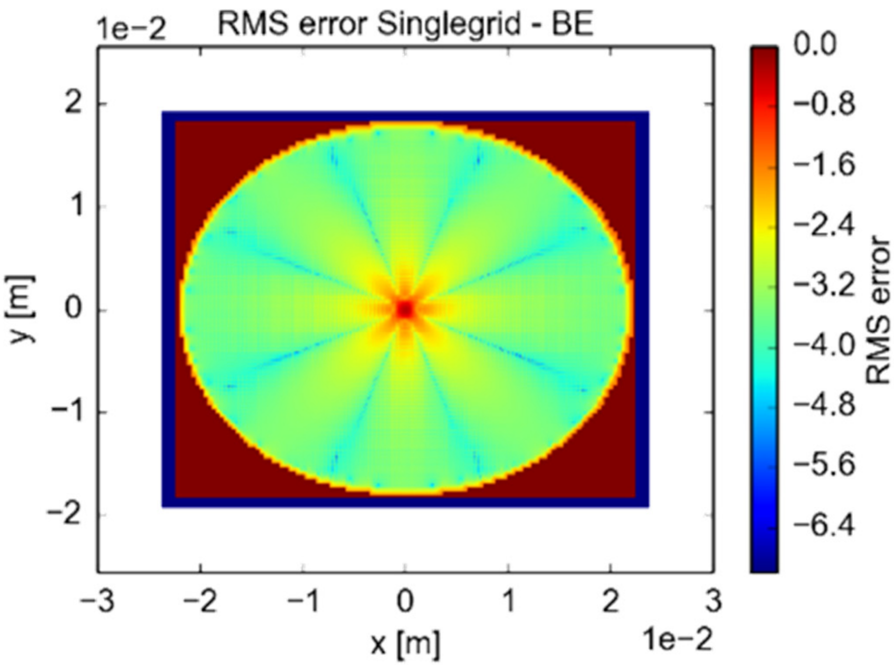
- Example
  - RMS error compared to Bassetti-Erskine

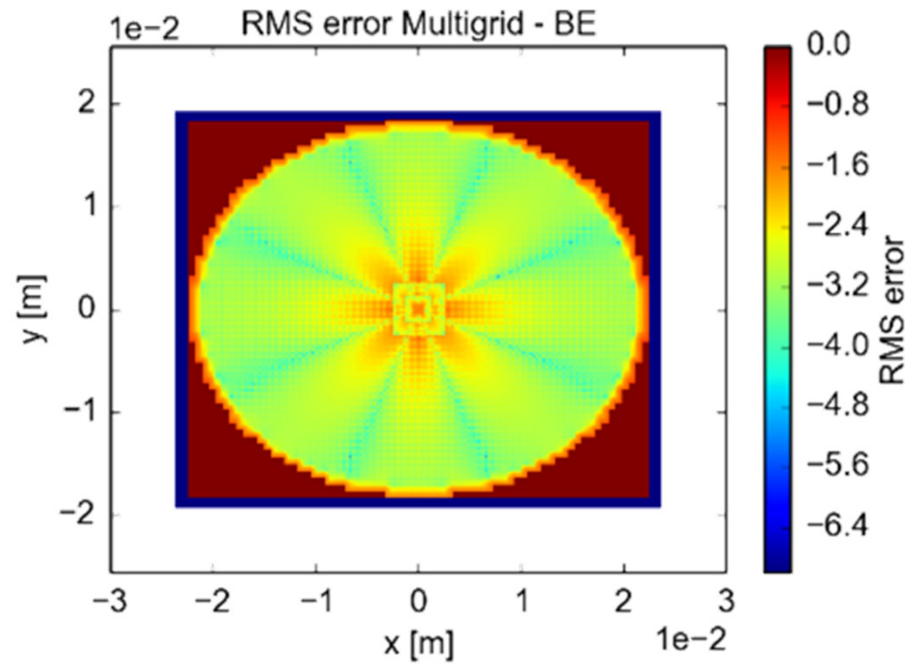


Multigrid method

with E. Belli

- Example
  - RMS error map (logarithmic scale)
  - Better resolution around beam in multigrid
  - At the expense of slightly lower resolution outside (for similar run times)





Challenges

- Run-time performance
  - Dynamics of instability proportional to beam brightness
  - In damping ring, brightness increases by large factor during damping period
  - To capture full dynamics, ideally simulate full damping period
    - CLIC main damping ring, damping time around 2 ms ~ 1400 turns
  - FASTION: 1 turn ~ 20 min → 20 days for full damping cycle
  - PyEC-PyHT: currently ~ 50 % slower
    - Profiling shows is largely due to FFT solver, room for optimization
    - Multigrid may also help
  - Too long in both cases!
- Effort ongoing to create parallelization layer applicable to ion, e-cloud & other studies

Summary & outlook

- Fast beam ion instability modeling implemented in PyECLOUD – PyHEADTAIL
  - First application to CLIC main damping ring
  - Benchmarked against FASTION
  - Ready for systematic studies
- Many new features available
  - Future studies to estimate effect on instability
- Multigrid solver methods have been implemented
  - Essential for good resolution without compromising on performance
  - First full multigrid simulations with PyEC-PyHT are being run
- Long run times still a problem
  - Parallelization effort ongoing

Thank you!

Thanks to PyPIC, PyECLOUD and PyHEADTAIL developers:
H. Bartosik, E. Belli, S. Hegglin, K.Li, A. Oeftiger,
A. Passarelli, A. Romano, M. Schenk