Universität Rostock Traditio et Innovatio

SPONSORED BY THE
Federal Ministry of Education and Research

First Steps Towards a New Finite Element Solver for MOEVE PIC Tracking

U. van Rienen, J. Heller, D. Zheng and C.R. Bahls[a]

ICAP 2018, 23.10.2018

[a]Work supported by the German Federal Ministry for Research and Education BMBF under contract 015K16HRA

Outline

- Introduction
- Replacing MOEVE's FD Solver with FEM from FEniCS
- First Results
- Summary and Future Directions

MOEVE

- In-house Particle-in-Cell (PIC) code, written mostly in C [1]
- Particle mesh method
- Underlying mathematical approach: Finite Difference (FD) method
- Poisson's equation solved by conjugate gradient method with geometric multigrid as preconditioner
- Implemented in ASTRA[2] and GPT[3]

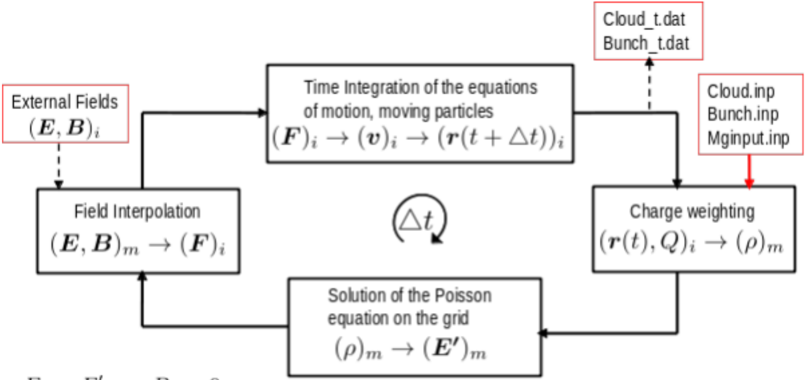
[1] G. Pöplau. *MOEVE: Multigrid Poisson Solver for Non-Equidistant Tensor Product Meshes* (2003)
[2] K. Flöttmann. *ASTRA: A space charge tracking algorithm*. Manual, Version 3 (2014)
[3] S. van der Geer, M. de Loos. *The general particle tracer code. Design implementation and application* (2001)

Ion clouds and Ion Clearing

- Residual gas can be ionized rapidly by electron beam $\rightarrow$ ion cloud
- Current-limiting factor for many synchrotron radiation sources
- Source of beam instabilities and beam loss
- Hinders continuous filling of electron bunches
- Main strategies to ensure a minimum stability in standard operational regimes:
  - Clearing gaps
  - Clearing electrodes
- High-current operation at ERL facilities requires precise analysis and development of appropriate measures to suppress ion-induced beam instabilities

Investigation of Ion Dynamics over Longer Distances

- Longitudinal transport of ions through the whole accelerator plays a key role for the establishment of the ion concentration in the machine
- This aspect of the dynamics has implications on both the beam dynamics and the ion clearing efficiency but it has not been deeply studied up to now
- Extent to which resonators contribute to the transport is largely unclear
- Thus, we are targeting a fast, systematic investigation of ion dynamics of the machines involving the impact on the beam
- Shall be applied to reduce the effects related to ionized residual gas in high-current electron machines
- This study follows our previous investigations on ion trapping in high-current storage rings and linear accelerators e.g. for bERLinPro



$$E_{||} = E'_{||} \qquad B_{||} = 0$$

$$E_{\perp} = \gamma E'_{\perp} \qquad B_{\perp} = \frac{\gamma}{c^2}(\boldsymbol{v} \times \boldsymbol{E}')_{\perp}$$

MOEVE [4]

[4] A. Markovic. *Simulation of the interaction of positively charged Beams and electron clouds*. PhD Thesis. Rostock University, 2013.

SPONSORED BY THE
Federal Ministry of Education and Research

Limitations due to Finite Differences and an Alternative

- MOEVE's prior limitations due to the underlying FD discretization:
  - Comparably large number of degrees of freedom (DOFs) required for accurate solution e.g. since tensor product grid of FD poorly approximates general boundary geometries
  - PIC scales with number of macro-particles and required mesh cells of discretization - number of macro-particles can not be further reduced but one can reduce the number of mesh cells by using a different discretization technique, e.g. the Finite Element Method (FEM)
- Using appropriate ansatz functions in FEM, e.g. *Crouzeix-Raviart* elements allows improving convergence by at least one order[5] $\rightarrow$ quadratic (or better) convergence in the force with FEM compared to linear convergence using FD $\rightarrow$ reduction in the number of mesh cells gets possible

[5] C.R. Bahls. *Space charged calculations using refinements on structured and unstructured grids*. PhD Thesis. Rostock University, 2015.

Replacing the FD Solver with FEM from FEniCS

- Electric field of charge density $\rho(x)$ results from scalar potential $u(x)$ obtained from Poisson's equation on domain Ω:

$$-\Delta u(x) = \frac{\rho(x)}{\varepsilon} \quad \forall x \in \Omega, \tag{1}$$

with vacuum permittivity ε_0 and following boundary conditions on $\partial\Omega$

$$u(x) = g_D(x) \quad \forall x \in \partial\Omega_D, \tag{2}$$

$$\frac{\partial\, u(x)}{\partial\, n(x)} = g_N(x) \quad \forall x \in \partial\Omega_N. \tag{3}$$

Weak Formulation of Poisson´s Equation

- We use FEniCS[6] to solve this boundary value problem after FEM discretization
- FEniCS allows to directly write down the weak formulation of Poisson's equation:

$$\int_\Omega \nabla u(x) \cdot \nabla v(x)\ dx = \int_\Omega \frac{\rho(x)}{\varepsilon_0}\ v(x)\ dx \quad \forall v \in V \tag{4}$$

as a pair of a bilinear form $a(u, v)$ and linear form $L(v)$::

$$a(u, v) = \int_\Omega \nabla u(x) \cdot \nabla v(x)\ dx; \qquad L(v) = \int_\Omega \frac{\rho(x)}{\varepsilon_0}\ v(x)\ dx \tag{5}$$

[6] A. Logg, K.-A. Mardal, G. N. Wells et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012

Some Aspects of FEniCS Implementation

- Meshing, definition of function spaces and bilinear form, etc. very straightforward
- Assembly of system matrix and solution by PETSc (Krylov subspace solvers) calls
- Use of charge weighting `PointSource` from DOLFIN[7]
- Available function spaces for field interpolation
    - Raviart-Thomas FE space
    - Brezzi-Douglas-Marini FE space
    - Discontinous-Galerkin vector function space
    - Continuous Lagrange (Courant) vector function space

[7] A. Logg, G. N. Wells and J. Hake. *DOLFIN: a C++/Python Finite Element Library. Automated Solution of Differential Equations by the Finite Element Method*. 2012.

First Results

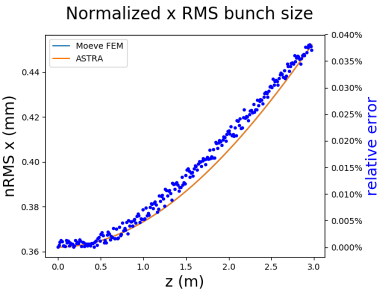
- Simple model problem
  - Tracking of an electron bunch of Gaussian distribution in all directions for a short drift space of 3.0 m
  - Initial bunch is generated by ASTRA[8]
  - Bunch profile is listed in table on next slide
  - No external electromagnetic field
  - No ion cloud
- Comparison with ASTRA: rms bunch size and emittance growth were compared with ASTRA for transverse directions

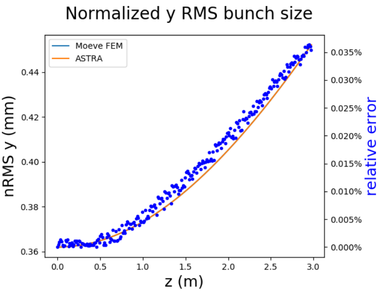
[8] K. Flöttmann. *ASTRA: A space charge tracking algorithm*. Manual, Version 3 (2014)

Bunch Profile for Tracking

Parameters of the electron bunch	
Number of macro particles	5,000
Beam energy	15 MeV
Beam energy spread	1.49 keV
Beam charge	-0.4 nC
Transverse emittance	1.0 π mradmm
Bunch length	0.88 mm
rms bunch radius	0.362 mm

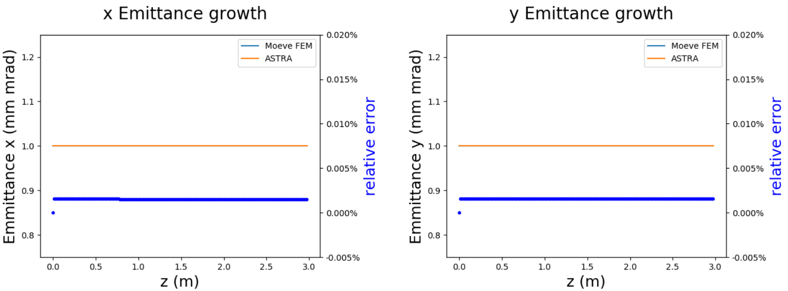
RMS Bunch Size Growth





Bunch size growth in transverse directions for a drift distance of 3.0 m without external electromagnetic fields as computed by MOEVE based on FEM and ASTRA, respectively. The relative error is shown as well.

Emittance Growth



Emittance growth in transverse directions for a drift distance of 3.0 m without external electromagnetic fields as computed by MOEVE based on FEM and ASTRA, respectively. The relative error is shown as well.

Results

- Results of MOEVE with the FEM-based FEniCS implementation and ASTRA agree very well both for the transverse bunch size growth and the emittance
- Relative error in the transverse bunch size growth follows a very similar functional behaviour as the transverse bunch size growth itself and reaches less than 0.04%
- Emittance stays constant over the drift - so does the relative error of about 0.003%

Summary

- MOEVE with new FEM-based FEniCS implementation to track electron bunches
- First study on simple model problem of tracking through a drift space without external electric field showed very good agreement with results obtained by ASTRA

Next Steps and Future Perspective

- Improve computational performance by MPI parallelization throughout the code
- Improve charge weighting with `PointSource` and speed of interpolating the electric field at particle position
- Implement adaptive hp-refinements, i.e. element size (h) and polynomial degree (p)
- Validation with measurements from ELSA in Bonn [9].
- Study ion cloud dynamics in bERLinPro[10]

[9] D. Sauerland, W. Hillert, A. Meseck. *Estimation of the ion density in accelerators using the beam transfer function technique*, Proceedings of IPACâĂŹ15 (2015)

[10] B. Kuske, N. Paulick, A. Jankowiak, J. Knobloch. *Conceptual Design Report* (2012)

Additional Slides on FEniCS Implementation

Some Aspects of FEniCS Implementation

- To be able to do this one has to import the Python module dolfin:

```
from dolfin import *
```

- specify the discrete function spaces (depending on the mesh used):

```
V = FunctionSpace(mesh,"CG",degree)
u = TrialFunction(V)
v = TestFunction(V)
```

- One can directly write down the bilinear form a as:

```
a = dot(grad(u), grad(v))*dx(mesh)
```

Preparation of System Matrix and Right Hand Side

- We can now prepare and assemble the system matrix `A` for the solver included in FEniCS:

```
template = PETScMatrix()
A = assemble(a,tensor=template)
```

- The linear form L in the weak formulation of Eq. (1) as given in Eq. (4) depends on the charge density arising from the charges in the domain.
- Starting from a constant $\rho = 0$ one can assemble the right hand side linear form L and the corresponding vector `rhs`:

```
L = Constant(0.0)*v*dx(mesh)
rhs = assemble(L)
```

Charge Weighting and Boundary Conditions

- We use the charge weighting implemented by the method `PointSource` from DOLFIN to add macro-particles to the right hand side:

```
macro_particles = []
for i in range(Number_of_Particles):
macro_particles.append((Particle[i], charge[i]/eps0))
delta = PointSource(V, macro_particles)
delta.apply(rhs)
```

- The definition and application of the boundary condition g_D on $\partial\Omega_D$[11]

```
bc = DirichletBC(V, g_D,"on_boundary")
bc.apply(A)
bc.apply(rhs)
```

[11] For ease of exposition, we choose to only show the implementation using a Dirichlet boundary

Solver Setup and Numerical Solution

- Setup of the solvers provided through DOLFIN (here the conjugate gradient method):

```
solver = PETScKrylovSolver("cg","default")
solver.parameters["relative_tolerance"] = residual
solver.set_operator(A)
```

- Solving for the unknown potential $u(x)$:

```
u_x = Function(V)
solver.solve(u_x.vector(), rhs)
```

Computation of Electric Field

- The electric field $\vec{E}$ can then be computed from the gradient $\nabla u(x)$ of the solution.

```
e_temp = -grad(u_x)
```

- To be applicable as an interpolated field it has to be projected onto an appropriate function space. For example the continuous Lagrange vector function space:

```
Efield = project(e_temp, VectorFunctionSpace(mesh,\
"CG", degree-1))
```

- The computed field can next be used to accelerate the particles using the well-known Boris pusher. [12]

[12] J.B. Boris. *Relativistic plasma simulation-optimization of a hybrid code"*. Proceedings of the 4th Conference on Numerical Simulation of Plasmas, November 1970. Naval Res. Lab., Washington, D.C.