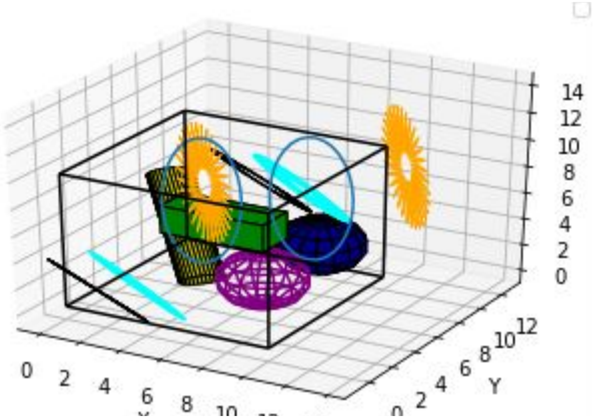
Accelerating the particle-in-cell method of plasma and particle beam simulation using CUDA tools.

Ivan Kadochnikov - kadivas@jinr.ru

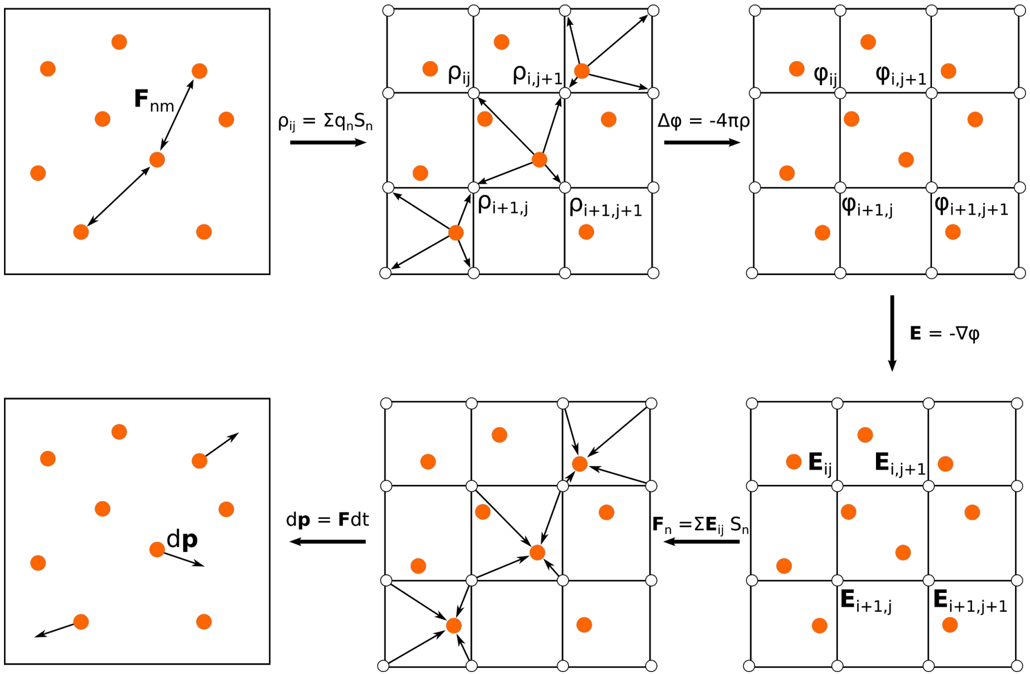
NEC 2019

Ef an ef_python

- Low-energy particle simulation using the Particle-in-Cell (PIC) method
- Open-source
- Focus on modeling ion sources and particle beams
- Support for external fields, conducting volumes and particle generators



Particle-in-cell



Particle mover

- Leapfrog second-order explicit method (Boris scheme)

$$\mathbf{r}_{i+1} = \mathbf{r}_i + \mathbf{v}_{i+1/2}\Delta t$$

$$\mathbf{v}_{i+1/2} = \mathbf{v}_+ + \frac{q\Delta t}{2m}\mathbf{E}(\mathbf{r}_i)$$

$$\mathbf{v}_+ = \mathbf{v}_- + \frac{q\Delta t}{2mc}(\mathbf{v}_+ + \mathbf{v}_-) \times \mathbf{B}(\mathbf{r}_i)$$

$$\mathbf{v}_- = \mathbf{v}_{i-1/2} + \frac{q\Delta t}{2m}\mathbf{E}(\mathbf{r}_i)$$

Field solver

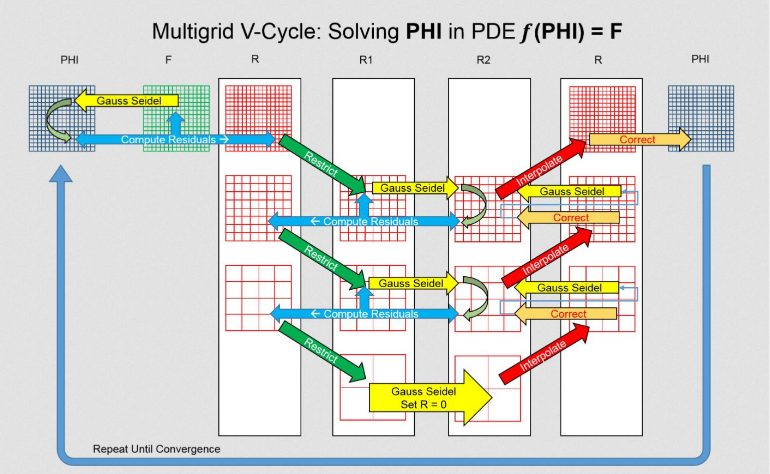
- Poisson equation (no dynamic magnetic field) $\Delta\phi = -4\pi\rho$
- Finite Difference Method (FDM) on a rectangular grid
- Initially solved by conjugate gradient method with scipy.sparse.linalg.cg

Each time step

- Push particles
  - Get electric field at particle positions(grid-to-particle, linear interpolation)
  - Push particle positions
- Generate new particles
- Solve field
  - Evaluate collective charge density (particle-to-grid)
  - Solve poisson equations for potential
  - Compute field from potential (just gradient)
- Prepare new particles
  - Get field at new particles(grid-to-particle, linear interpolation)
  - Set new particle velocities half a time step back

Algebraic Multi-Grid solver

- Multi-scale methods for faster FDM solving



PyAMG and AMGX

PyAMG

Algebraic Multigrid Solvers in Python

Non-GPU improvements

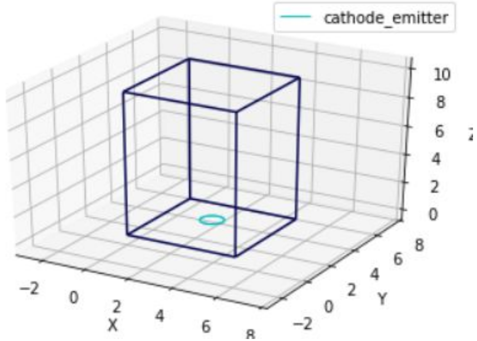
- Use diagonal sparse matrix functions of Scipy
- Caching inner nodes
- Grid-to-particle and particle-to-grid with numpy methods

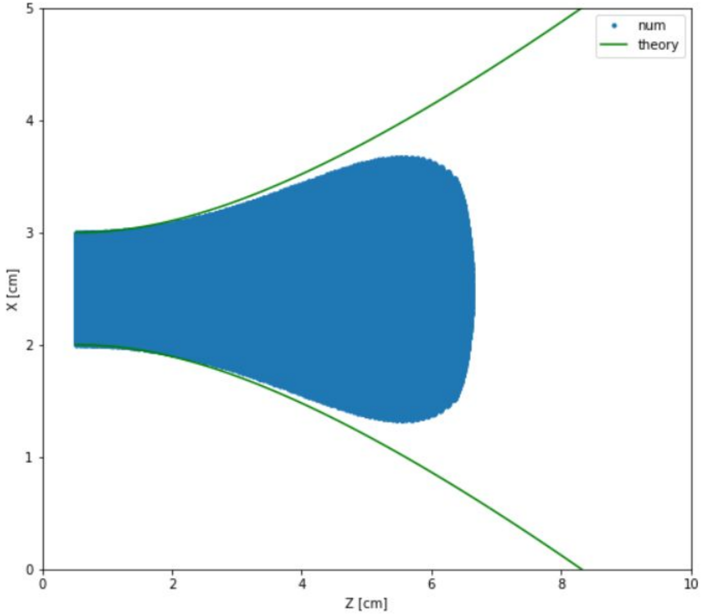
Numpy to Cupy

- Intended as a drop-in replacement/upgrade
- Some methods have new names
- No interpolator class
    - Custom CUDA kernel

Axially symmetric beam contour

- 50 x 50 x 100 grid
- 100 steps
- 5000 x 100 particles





Results

- 40x40x5000 grid, 2800 steps, 4x2800 particles
    - Simulated in 20 minutes on GPU (resources provided by JINR HybriLIT cluster)
- GPU accelerated every step of the simulation process
    - Optional, by default using pyamg and numpy
    - Except output and equation matrix creation
- 200+ tests with 91% coverage
- Next step:
    - OpenCL
    - MPI

Thank you!

- The reported study was funded by RFBR according to the research project № 18-32-00239