FBPIC

A multi-GPU Particle-In-Cell code for plasma acceleration

github.com/fbpic

Manuel Kirchen,

Rémi Lehe, Sören Jalas, Jean-Luc Vay, Andreas R. Maier

Center for Free-Electron Laser Science
University of Hamburg, Germany
manuel.kirchen@desy.de

lux.cfel.de

FBPIC

A spectral, quasi-3D PIC code

In collaboration with

Rémi Lehe, Berkeley Lab

R. Lehe et al., CPC, 2016

Development goals

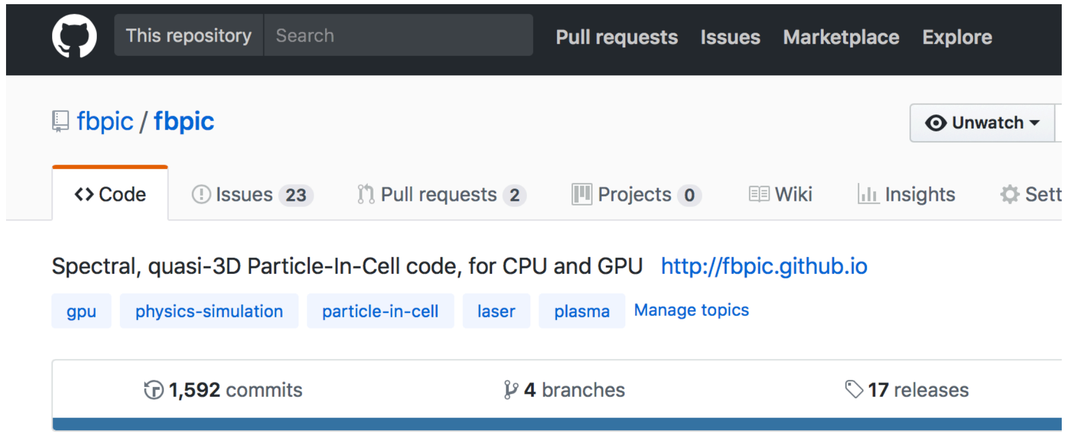
- Increase accuracy & lower costs
- Focus on plasma acceleration & ease of use

Key features

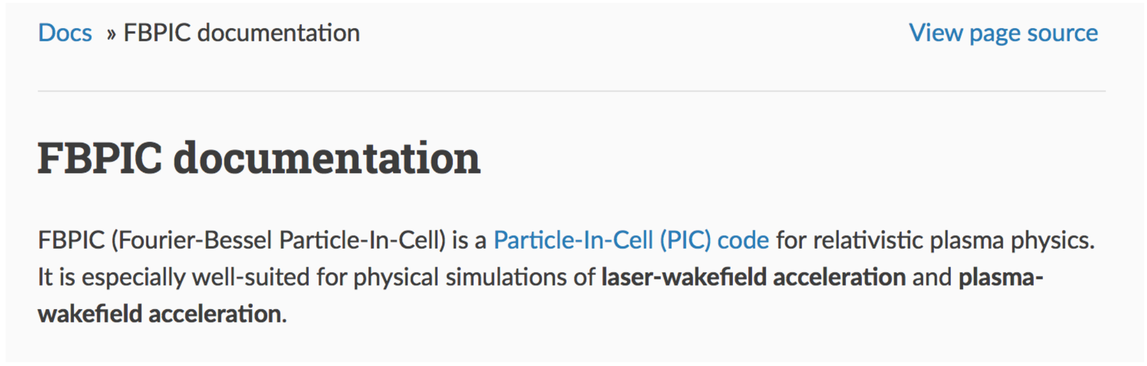
- **Quasi-3D geometry**
- **Parallel spectral solver**
- **Lorentz-boosted frame**
- Python, GPU acceleration, MPI parallelization

github.com/fbpic

Open source



Online documentation

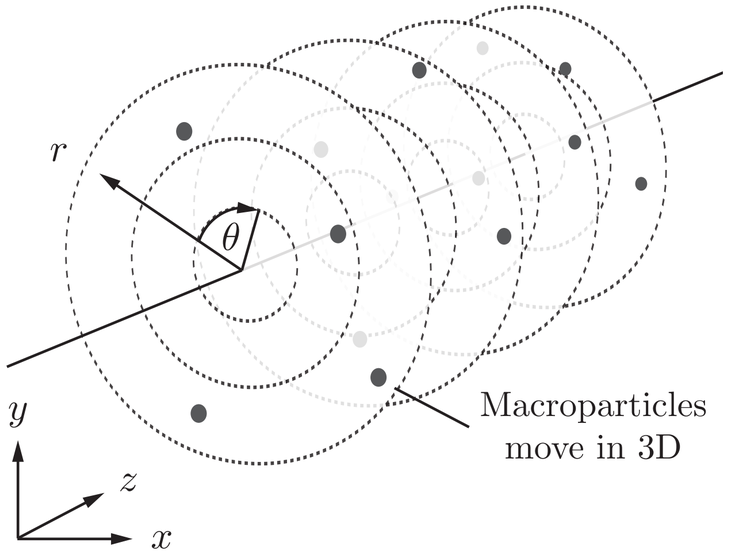


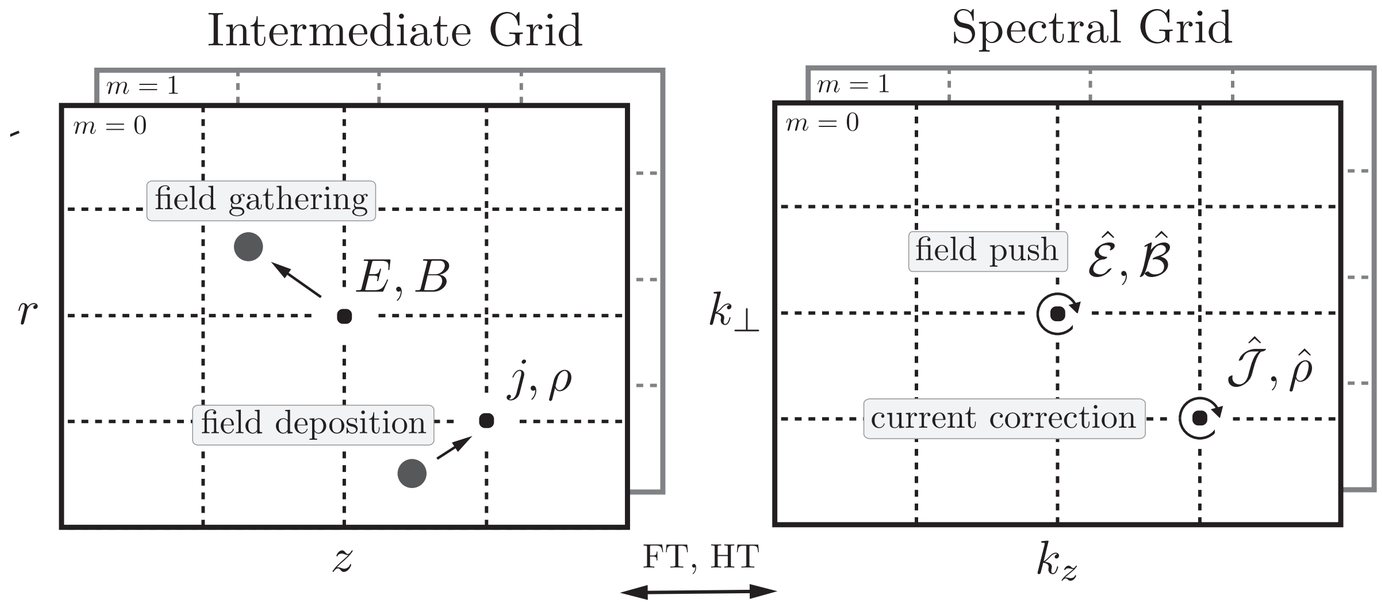
FBPIC

Quasi-3D geometry with spectral solver

Quasi-3D geometry

- Cylindrical geometry
- Decomposition in azimuthal modes
- Results as accurate as 3D for many cases
- Computational costs similar to 2D





Spectral solver

- Pseudo-spectral solver with analytical time integration (PSATD)
- Field-particle interaction in spatial domain
- Field integration in spectral domain
- No numerical dispersion, No CFL condition, No staggering

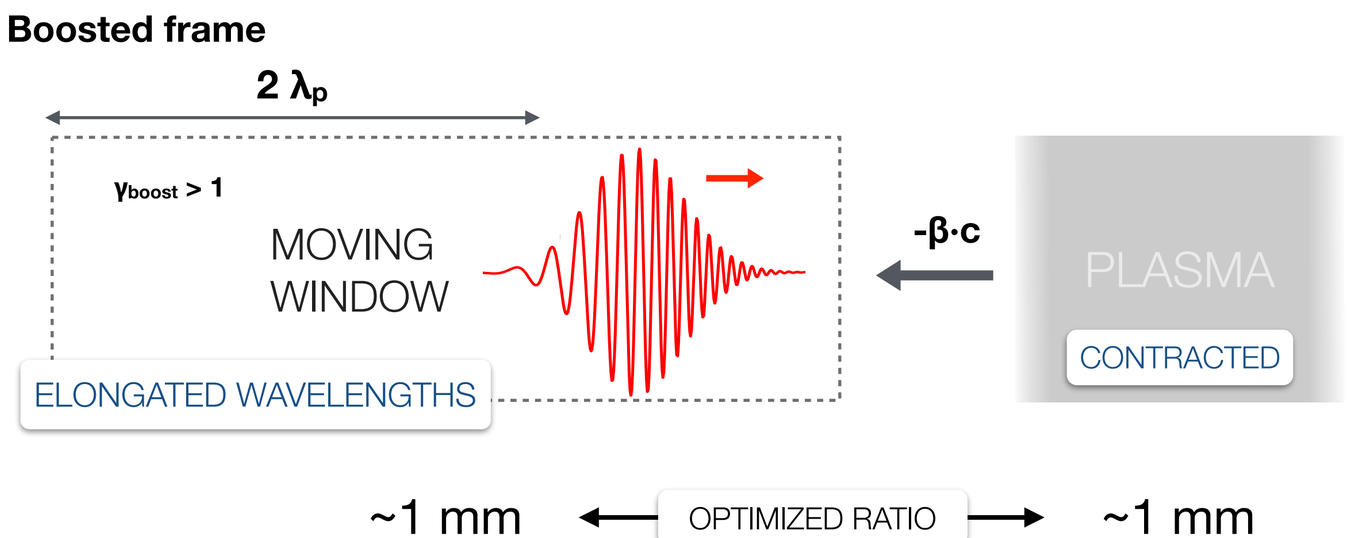
Lorentz-boosted frame

Speedup & elimination of Numerical Cherenkov Instability

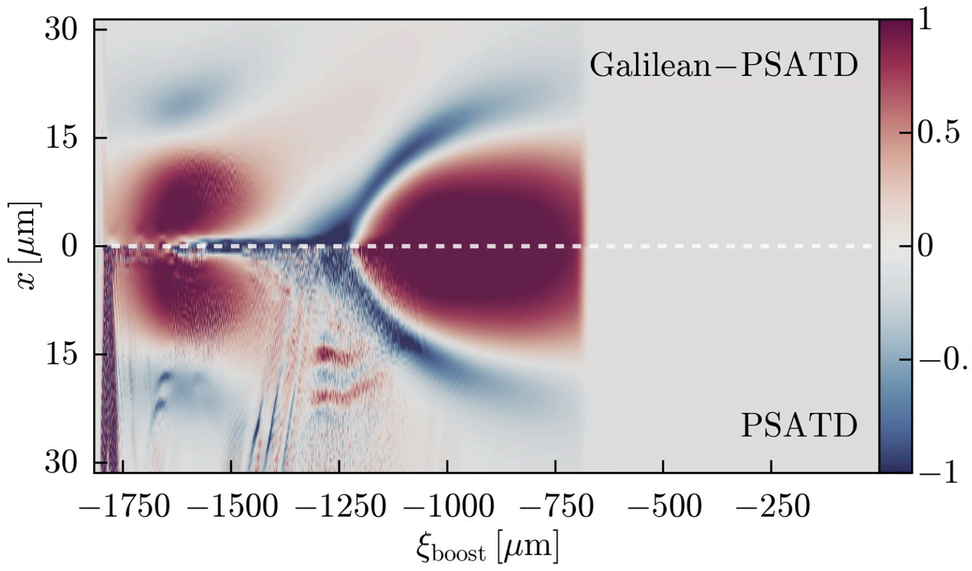
Idea & applications: Physics of Plasmas 23, 100704 (2016)

Math & stability analysis: Phys. Rev. E 94, 053305 (2016)

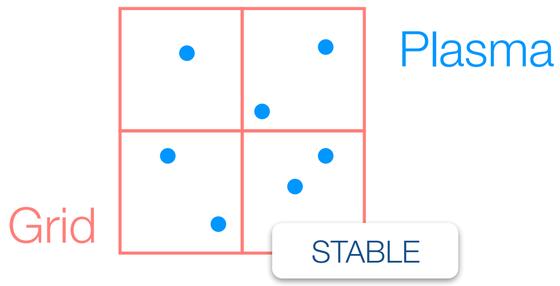
or checkout **lux.cfel.de/publications/**



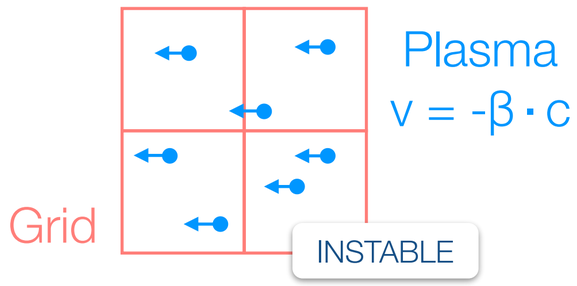
stability of Galilean spectral solver



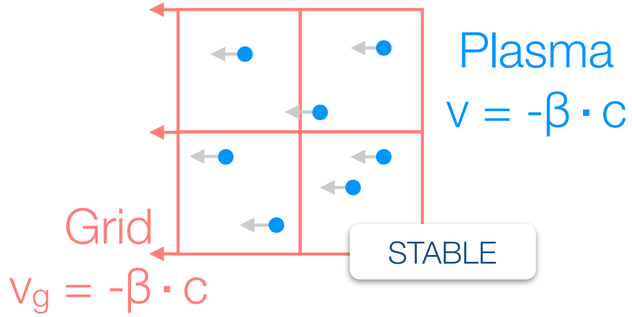
Lab frame



Boosted frame



Boosted frame + Galilean frame

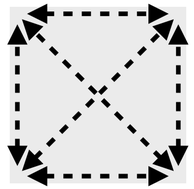


FBPIC

Implementation & Parallelisation

Intra-node parallelisation

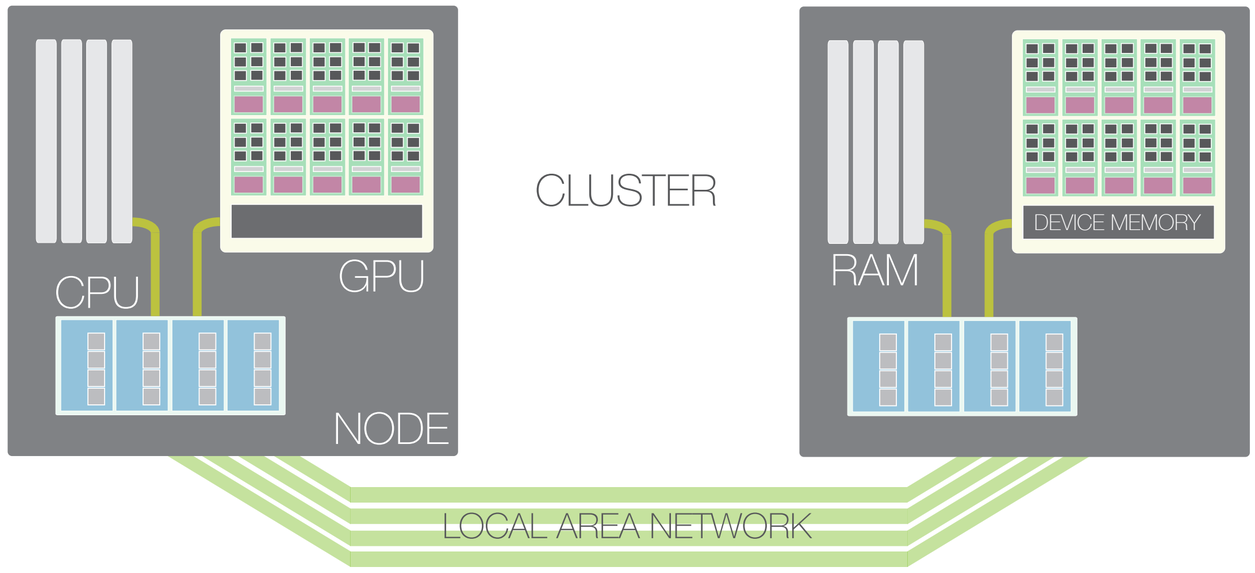
- Shared memory layout
- GPU or multi-core CPU
- *Parallel PIC methods & Transformations*
- ***Numba + cupy (CUDA)***

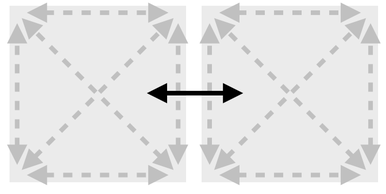


Inter-node parallelisation

- Distributed memory layout
- Multi-CPU / Multi-GPU
- *Spatial domain decomposition for spectral codes*
- ***mpi4py***

Typical infrastructure of a modern high-performance cluster





Using FBPIC

Python scriptable input

Running FBPIC

- Simple Python script as run file
- No compilation required

Starting a simulation is easy…

```
python fbpic_script.py
```

```
FBPIC (0.8.0)

Running on CPU (4 threads per process)

Initializing laser pulse on the mesh...
Done.

|                                    | 130/200, 0:00:08 left, 105 ms/step
```

part of a typical input script

```
# The density profile
ramp_start = 30.e-6
ramp_length = 40.e-6

def dens_func( z, r ) :
    """Returns relative density at position z and r"""
    # Allocate relative density
    n = np.ones_like(z)
    # Make linear ramp
    n = np.where( z<ramp_start+ramp_length,
                  (z-ramp_start)/ramp_length,
                   n )
    # Supress density before the ramp
    n = np.where( z<ramp_start, 0., n )
    return(n)

# Initialize the simulation object
sim = Simulation( Nz, zmax, Nr, rmax, Nm, dt,
    p_zmin, p_zmax, p_rmin, p_rmax, p_nz, p_nr, p_nt, n_e,
    dens_func=dens_func, zmin=zmin, boundaries='open',
    n_order=n_order, use_cuda=use_cuda )

# Define a Laguerre-Gauss laser profile
profile = LaguerreGaussLaser( a0=0.5, waist=5.e-6,
                              tau=30.e-15, z0=3.e-6,
                              p=1, m=0 )

# Add the laser pulse to the simulation
add_laser_pulse( sim, profile )

# Add a Gaussian electron bunch to the simulation
add_elec_bunch_gaussian( sim, sig_r=1.e-6, sig_z=1.e-6,
                         n_emit=1.e-6, gamma0=500, sig_gamma=5,
                         Q=1.e-9, N=10000 )
```

Using FBPIC

A lot of practical features

Useful features

- SI units
- Arbitrary density profiles as simple python function
- Custom or pre-defined laser profiles
- Custom 6D phase space or pre-defined particle beams
- Exact calculation of space-charge fields at initialization
- Relativistic Ionisation model (ADK)
- External fields
- Particle tracking

openPMD

Open Standard for Particle-Mesh Data Files

Created by **Axel Huebl**
Bekerley Lab & HZDR, et al.

Open source data format standard

- Supported by many PIC codes
- Data exchange and interoperability
- Common analysis frameworks

openPMD
Open Standard for Particle-Mesh Data Files
http://www.openPMD.org a.huebl@hzdr.de

github.com/openPMD

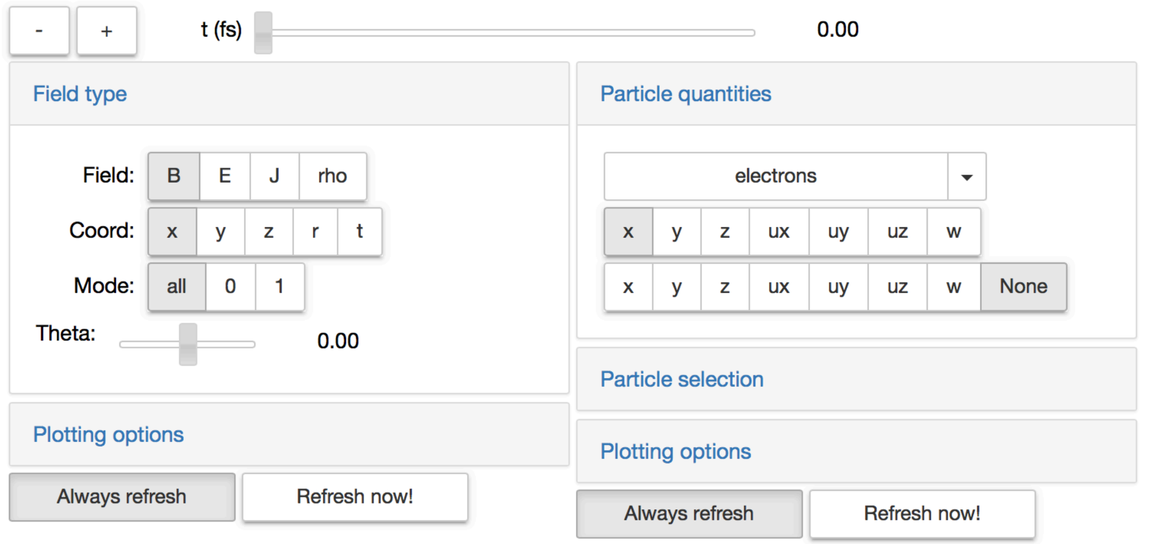
OpenPMD viewer

- Interactive visualisation
- Pre-defined analysis functions

```
from opmd_viewer.addons import LpaDiagnostics as opmd
```

```
data = opmd('./diags/hdf5/')
```

```
data.slider()
```



PICMI

Standard input format for Particle-In-Cell codes

Created by **J.-L. Vay's team**
Berkeley Lab, et al.

PICMI

github.com/picmi-standard/picmi

Particle-In-Cell Modelling Interface (PICMI)

- Dictionary as input syntax
- Primary implementation: Python classes
- Extensible for code-specific needs