A science case

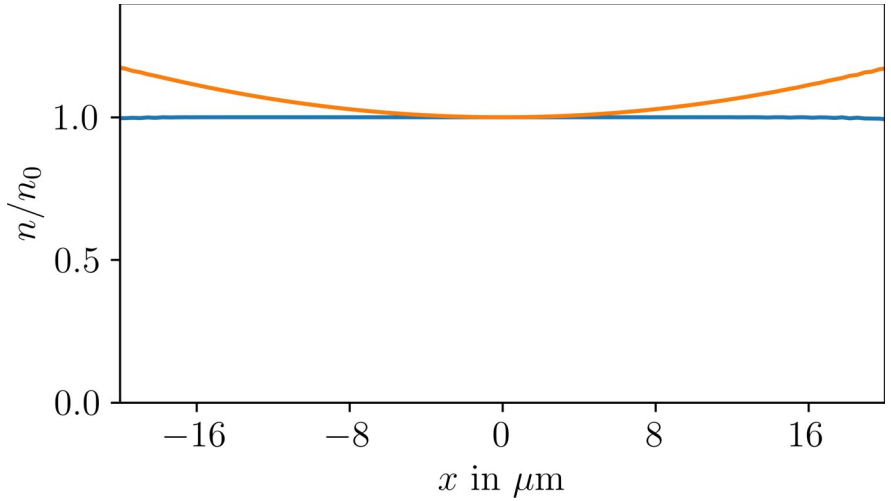
From an idea to (almost) a paper with HiPACE++

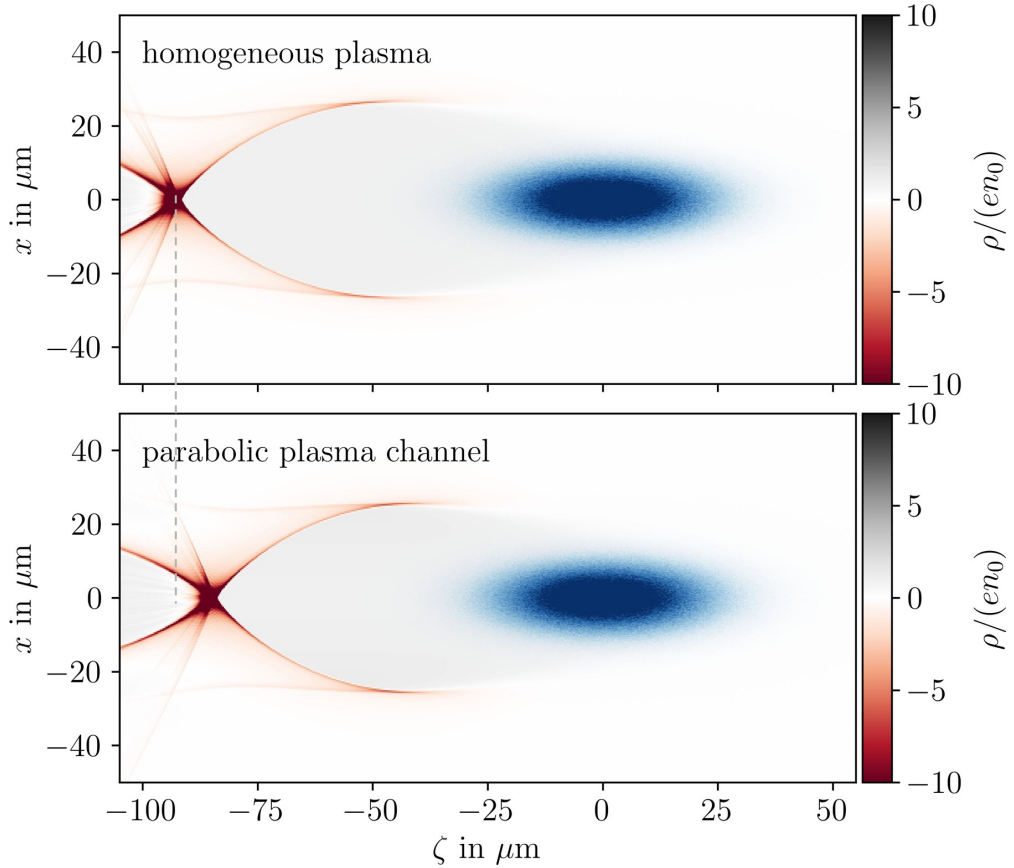
Severin Diederichs

HiPACE++ workshop, 11.07.2023

Stability of an electron beam in a parabolic plasma channel

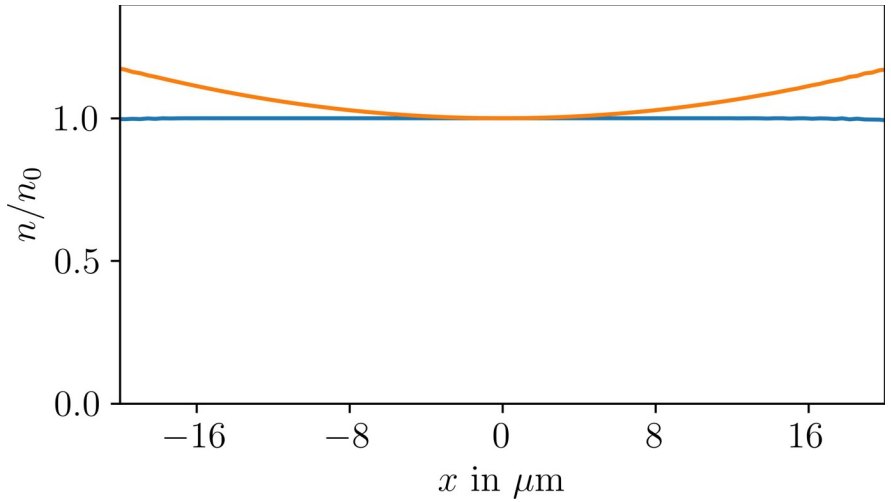
We want to study the stability of an electron beam in a parabolic plasma channel
(and compare vs homogeneous plasma)

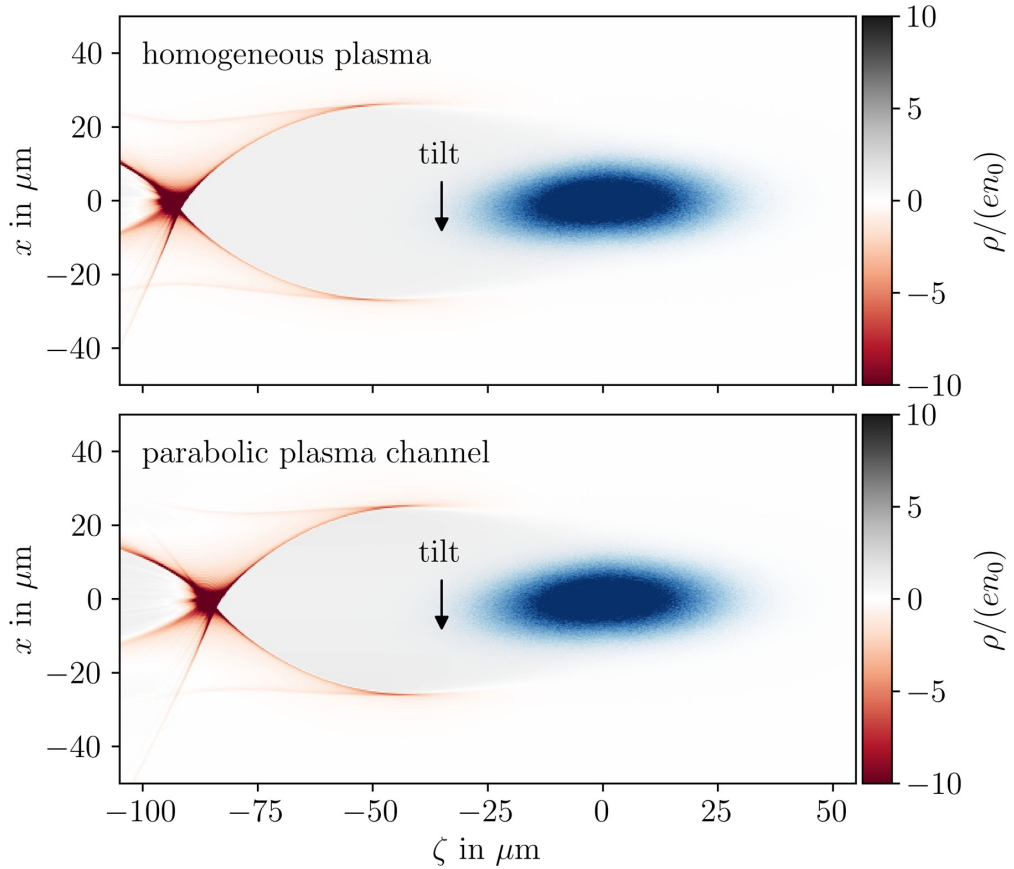




Stability of an electron beam in a parabolic plasma channel

We want to study the stability of an electron beam in a parabolic plasma channel
(and compare vs homogeneous plasma)





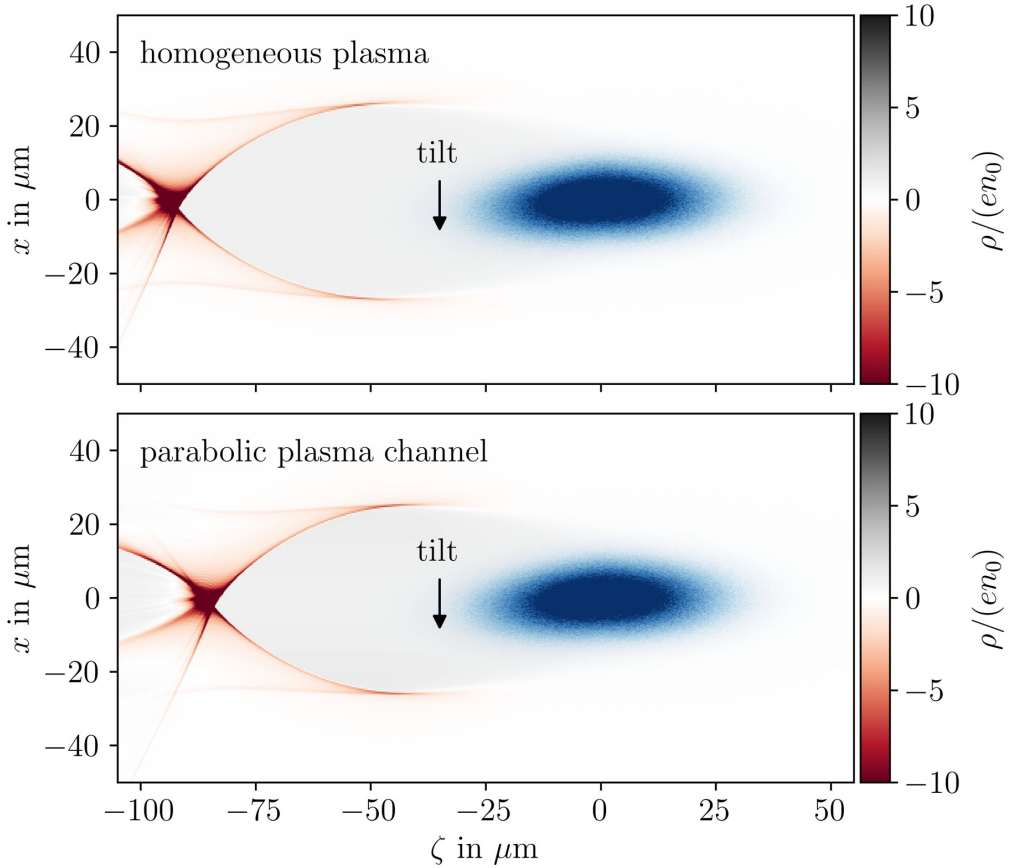
Stability of an electron beam in a parabolic plasma channel

Pick the correct code!

Quasi-static approximation is valid: ✓
- No injection
- Beam evolves slowly compared to plasma

Problem is full 3D: ✓
- Tilted beams
- Offset with respect to parabolic channel

HiPACE++ is a very good choice to model the problem



Getting the code to run

Choose the platform:

- Laptop / desktop computer preferably with an NVIDIA GPU
- Supercomputer preferably with NVIDIA or AMD GPUs

For demanding runs (high resolution, many particles) a supercomputer is advised

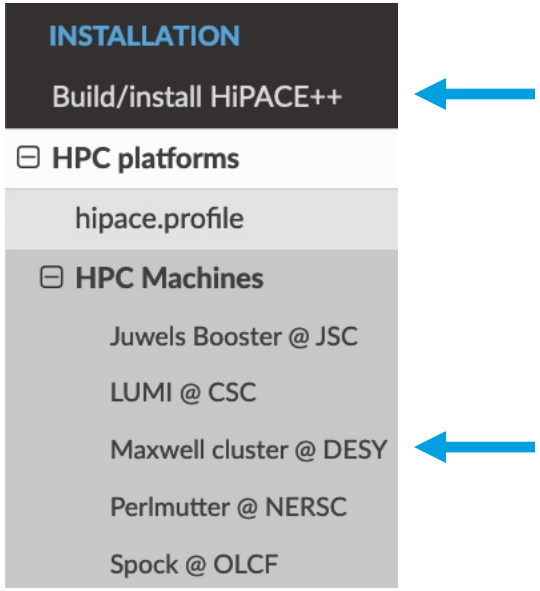
Today: local cluster Maxwell equipped with NVIDIA A100 GPUs

Getting the code to run

Choose the platform:

- Laptop / desktop computer preferably with an NVIDIA GPU
- Supercomputer preferably with NVIDIA or AMD GPUs

Documentation: https://hipace.readthedocs.io/en/latest/



General build instructions

Specified instructions for certain supercomputers

Is your favorite machine missing? Let us know and we work it out together

Getting the code to run

Documentation: https://hipace.readthedocs.io/en/latest/building/building.html

1. **Get dependencies**
   - via installing them using Spack or Homebrew
   - via loading modules on a supercomputer

Spack https://spack.readthedocs.io/en/latest/

```
spack env create hipace-dev
spack env activate hipace-dev

spack add cmake
spack add ccache # optional, faster compilation
spack add mpi
spack add hdf5
spack add adios2
# Choose depending on platform:
# for CPU:
spack add fftw
spack add pkgconfig
# for NVIDIA GPU
spack add cuda
# for AMD GPU
spack add hip
spack add rocfft
spack add rocprim
spack add rocrand

spack install
```

Maxwell

```
module purge
module load maxwell gcc/9.3 openmpi/4
module load maxwell cuda/11.8
module load hdf5/1.10.6
# pick correct GPU setting (this may differ for V100 nodes)
export GPUS_PER_SOCKET=2
export GPUS_PER_NODE=4
# optimize CUDA compilation for A100
export AMREX_CUDA_ARCH=8.0 # use 8.0 for A100 or 7.0 for V100
```

Using package managers or preinstalled software makes the installation quick and easy!*

*Please report issues, so we can add them to the documentation

Getting the code to run

Documentation: https://hipace.readthedocs.io/en/latest/building/building.html

1. Get dependencies
2. **Download HiPACE++** for the first time or the latest version

```
#  first time
git clone https://github.com/Hi-PACE/hipace.git $HOME/src/hipace

# latest version (if you are on the development branch)
git pull
```

Getting the code to run

Documentation: https://hipace.readthedocs.io/en/latest/building/building.html

1. Get dependencies
2. Download HiPACE++
3. **Compile the code** with cmake

for NVIDIA GPUs (other options: HIP for AMD, OMP for CPU)

```
rm -rf build
cmake -S . -B build -DHiPACE_COMPUTE=CUDA -DHiPACE_PRECISION=SINGLE
cmake --build build -j 8
```

Additional, optional **compile-time parameters**

All options can be found under:
https://hipace.readthedocs.io/en/latest/building/building.html#build-test

An executable will be created in */build/bin/* including a symbolic link ***/build/bin/hipace***

Getting the code to run

Documentation: https://hipace.readthedocs.io/en/latest/building/building.html

1. Get dependencies
2. Download HiPACE++
3. Compile the code
4. **Run the code**

Local computer:

```
# optional, for MPI parallelization
  mpirun –n 4 $HIPACE_PATH/build/bin/hipace input_script
```

name of the input script
(see next slides)

On a **supercomputer**, a job submission script is needed.
(unless an interactive node is used)

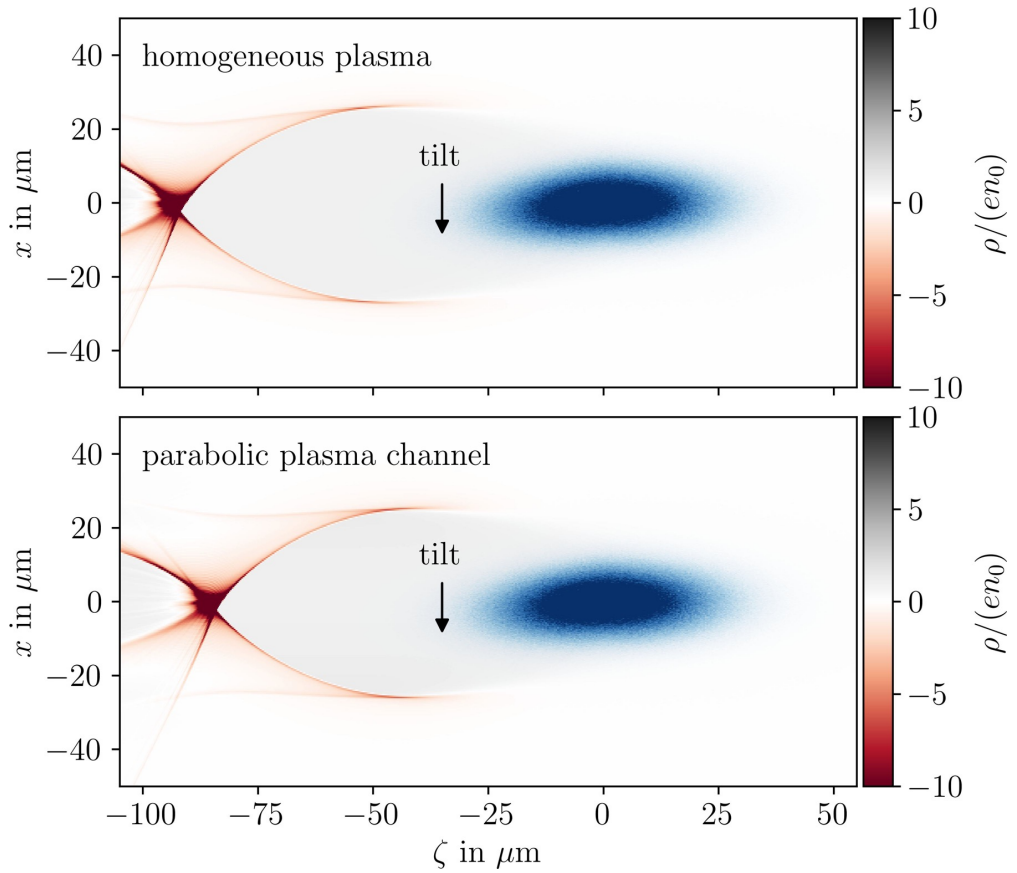
→ See your supercomputer's documentation

For Maxwell:

```
#! /usr/bin/env sh
#SBATCH --partition=<partition> # mpa # maxgpu # allgpu
#SBATCH --time=01:00:00
#SBATCH --nodes=1
#SBATCH --constraint=A100&GPUx4 # A100&GPUx1
#SBATCH --job-name=HiPACE
#SBATCH --output=hipace-%j-%N.out
#SBATCH --error=hipace-%j-%N.err

export OMP_NUM_THREADS=1
module load maxwell gcc/9.3 openmpi/4 cuda/11.8

mpiexec -n 4 -npernode 4 $HOME/src/hipace/build/bin/hipace inputs
```

First basic input script

This is what we want to get in the end



Simple drive beam in the homogeneous plasma (single time step)

```
amr.n_cell = 255 255 256
amr.max_level = 0

geometry.is_periodic = false  false  false
geometry.prob_lo     = -200.e-6 -200.e-6 -110.e-6
geometry.prob_hi     =  200.e-6  200.e-6   80.e-6

max_step = 0
hipace.dt = 0
hipace.verbose = 1

diagnostic.output_period = 1
diagnostic.diag_type = xz
hipace.deposit_rho = 1

my_constants.ne = 1e23

beams.names = driver
driver.injection_type = fixed_weight
driver.num_particles = 100000
driver.total_charge = 750e-12
driver.position_mean = 0 0 0
driver.position_std = 8e-6 8e-6 20e-6
driver.u_mean = 0 0 2000
driver.u_std = 5 5 0

plasmas.names = plasma
plasma.density(x,y,z) = ne
plasma.ppc = 2 2
plasma.element = electron
```

First basic input script

Output:

```
MPI initialized with 1 MPI processes
CUDA initialized with 1 device.
AMReX (23.07-12-gbc806f0f15ce) initialized
HiPACE++ (v23.05-66-g945fb8d119e0-dirty) running in double precision
using CUDA version 11.8.89
using the leapfrog plasma particle pusher
using R2C cuFFT of size 512 * 512 with 3 MiB of work area
Rank 0 started  step 0 at time = 0 with dt = 0

TinyProfiler total time across processes [min...avg...max]: 0.5744
... 0.5744 ... 0.5744

-------- Long list of performance data ---------
```

Plus a folder ***diags/hdf5/*** with the actual data

Simple drive beam in the homogeneous plasma (single time step)

```
amr.n_cell = 255 255 256
amr.max_level = 0

geometry.is_periodic = false  false  false
geometry.prob_lo     = -200.e-6 -200.e-6 -110.e-6
geometry.prob_hi     =  200.e-6  200.e-6   80.e-6

max_step = 0
hipace.dt = 0
hipace.verbose = 1

diagnostic.output_period = 1
diagnostic.diag_type = xz
hipace.deposit_rho = 1

my_constants.ne = 1e23

beams.names = driver
driver.injection_type = fixed_weight
driver.num_particles = 100000
driver.total_charge = 750e-12
driver.position_mean = 0 0 0
driver.position_std = 8e-6 8e-6 20e-6
driver.u_mean = 0 0 2000
driver.u_std = 5 5 0

plasmas.names = plasma
plasma.density(x,y,z) = ne
plasma.ppc = 2 2
plasma.element = electron
```

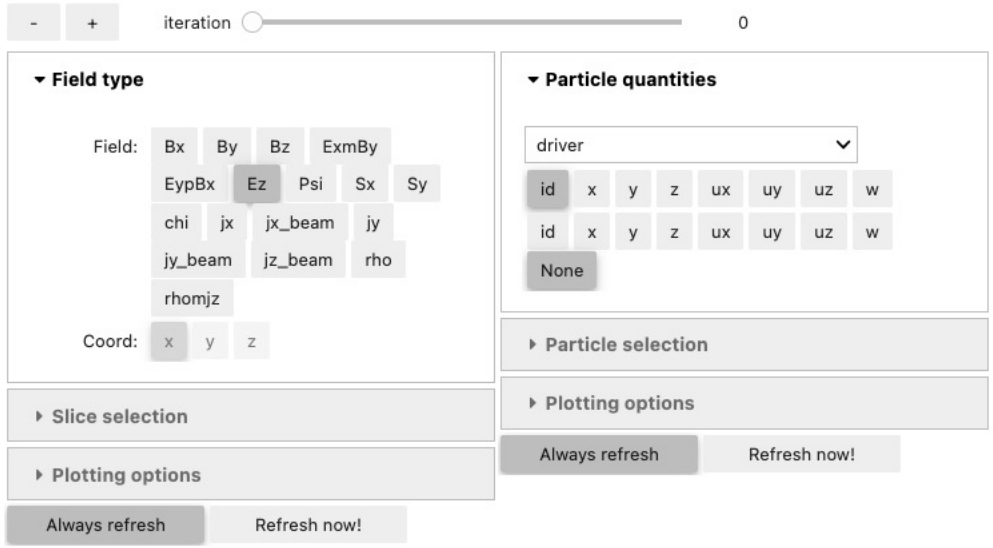
First visualization

Using the openPMD-viewer
Full documentation including installation and tutorials:
https://github.com/openPMD/openPMD-viewer

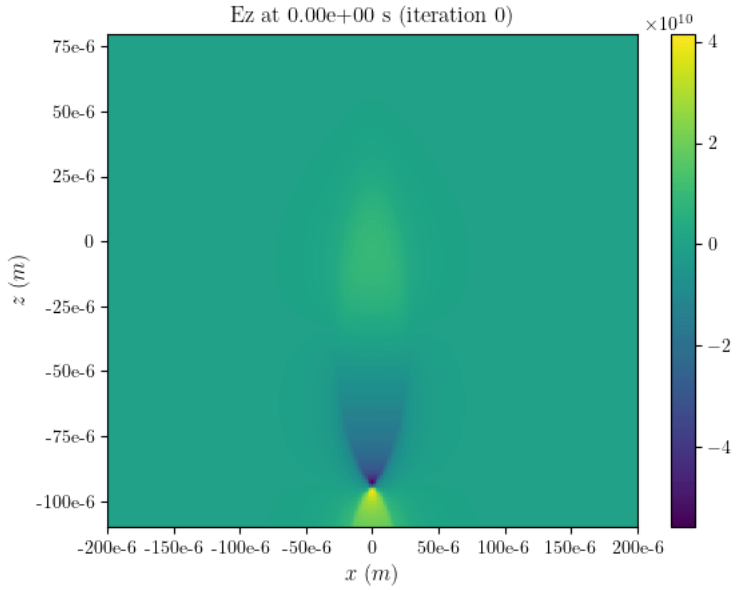
```
import numpy as np
%matplotlib widget
import matplotlib.pyplot as plt
from openpmd_viewer import OpenPMDTimeSeries
from openpmd_viewer.addons import LpaDiagnostics
from scipy import constants as scc

plt.rc('text', usetex=True)
plt.rc('font', family='serif')
```

```
ts = LpaDiagnostics('path_to_your_output/diags/hdf5/', check_all_files=False)
```

```
ts.slider()
```



using the slider for a quick assessment of the simulation…



First visualization

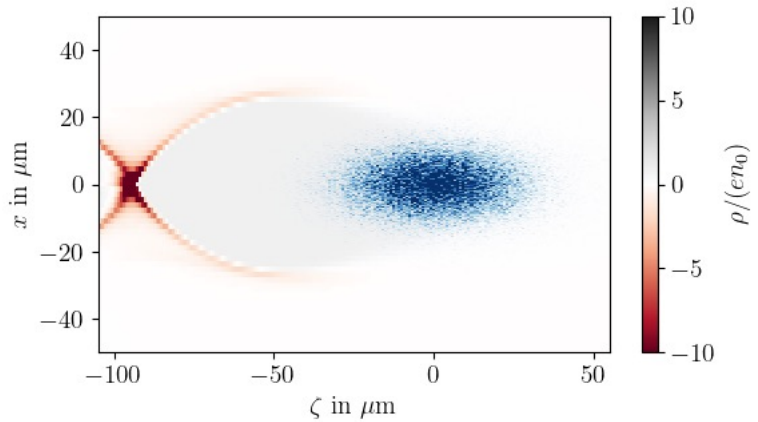
Using the openPMD-viewer
Full documentation including installation and tutorials:
https://github.com/openPMD/openPMD-viewer

```
from matplotlib.colors import ListedColormap
cmap = plt.cm.Blues
my_cmap = cmap(np.arange(cmap.N))
# Set alpha
my_cmap[:,-1] = np.linspace(0, 1, cmap.N)
# Create new colormap
mycmap = ListedColormap(my_cmap)
```

```
iteration = 0
rho, info = ts.get_field(field='rho', iteration=iteration)
Ez, info = ts.get_field(field='Ez', iteration=iteration)
ExmBy, info = ts.get_field(field='ExmBy', iteration=iteration)

x, y, ux, uy, w, z = ts.get_particle(var_list=['x','y', 'ux', 'uy' ,'w', 'z'], iteration=iteration, species="driver")
```

```
fig, ax =plt.subplots(figsize=(6,3.5))

# 2D plot for charge density
im = ax.pcolormesh(info.z*1e6, info.x*1e6, (np.transpose(rho))/scc.e/1e6/1e17, cmap='RdGy', vmin=-10, vmax=10)
cb = plt.colorbar(im, ax=ax)
cb.set_label(r'$\rho /(e n_0)$')

# histogram for beam particles
ax.hist2d(z[w>0]*1e6,x[w>0]*1e6, bins=[200, 200], cmap=mycmap, vmin=0, vmax=25)

ax.set_xlim(-105, +55)
ax.set_ylim(-50, 50)

ax.set_ylabel('$x$ in $\mu$m')
ax.set_xlabel('$\zeta$ in $\mu$m')
plt.tight_layout()
```

… then loading the data for proper analysis



Basics look fine, let's do a proper run

Beam propagation

Change the following input parameters

```
max_step = 300
hipace.dt = adaptive
diagnostic.output_period = 10
```

run time: 32s on 4 A100

```
# centroid in x
Xb = np.zeros(len(ts.iterations))

for i in range(len(ts.iterations)):
    x, w = ts.get_particle(var_list=['x','w'], iteration=ts.iterations[i], species="driver")
    Xb[i] = np.average(x, weights=w)
```

```
fig, ax = plt.subplots()

ax.plot(ts.t*scc.c, Xb*1e6)
ax.set_xlabel("$z$ in m")
ax.set_ylabel("$X_b$ in $\mu$m")
```

Beam propagation

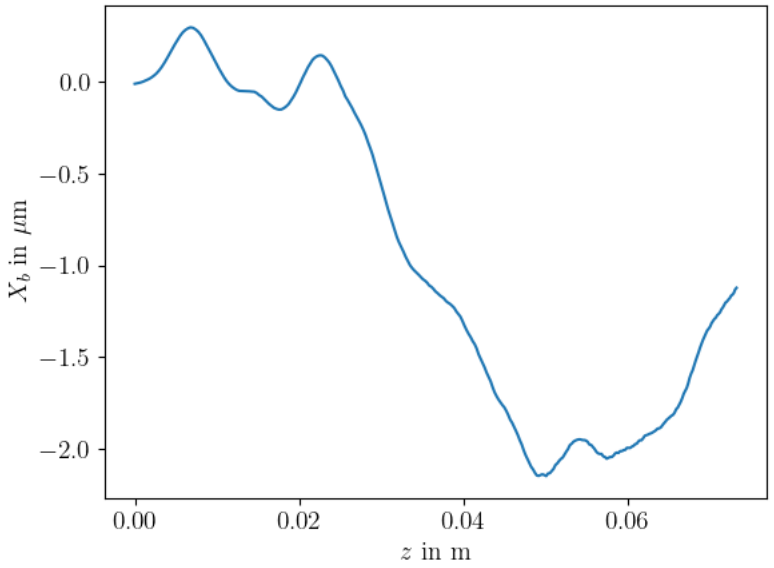
Change the following input parameters

```
max_step = 300
hipace.dt = adaptive
diagnostic.output_period = 10
```

run time: 32s on a A100

```
# centroid in x
Xb = np.zeros(len(ts.iterations))

for i in range(len(ts.iterations)):
    x, w = ts.get_particle(var_list=['x','w'], iteration=ts.iterations[i], species="driver")
    Xb[i] = np.average(x, weights=w)
```

```
fig, ax = plt.subplots()

ax.plot(ts.t*scc.c, Xb*1e6)
ax.set_xlabel("$z$ in m")
ax.set_ylabel("$X_b$ in $\mu$m")
```



Beam is unstable!
Hosing instability is physical, but simulations can have an uncontrolled seed

Beam propagation

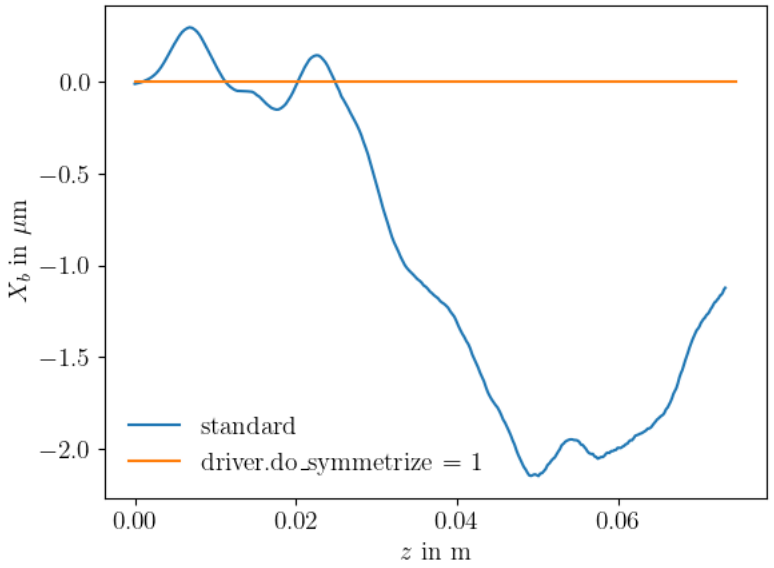
Change the following input parameters

```
max_step = 300
hipace.dt = adaptive
diagnostic.output_period = 10
```

run time: 32s on a A100

```
# centroid in x
Xb = np.zeros(len(ts.iterations))

for i in range(len(ts.iterations)):
    x, w = ts.get_particle(var_list=['x','w'], iteration=ts.iterations[i], species="driver")
    Xb[i] = np.average(x, weights=w)
```

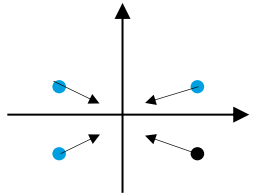
```
fig, ax = plt.subplots()

ax.plot(ts.t*scc.c, Xb*1e6)
ax.set_xlabel("$z$ in m")
ax.set_ylabel("$X_b$ in $\mu$m")
```



Beam is unstable!
Hosing instability is physical, but simulations can have an uncontrolled seed

To reduce the numerical seeding, use more beam particles
To suppress the numerical seeding, symmetrize the beam (or plasma for plasmas with a temperature) with `<beam name>.do_symmetrize = 1`

For each particle, another 3 particles are generated, so that the total momentum is 0 to machine precision



The parser allows for advanced input settings

E.g., to generate an advanced plasma profile or tilting the beam

- The parser allows to define own parameters
- Many physical constants are predefined
  (clight, q_e, m_e, m_p, epsilon0, …)
- Many different math functions available

More information:
https://hipace.readthedocs.io/en/latest/run/parameters.html#parser
https://amrex-codes.github.io/amrex/docs_html/Basics.html#parser

```
my_constants.dx_per_dzeta = 0.1
driver.position_mean = "if(z<0, z*dx_per_dzeta, 0)" 0 0

# pre-calculating parameter might be needed in single precision
my_constants.eme = 3182.607354 # e**2/(m_e*epsilon_0)
my_constants.omgp = sqrt(ne*eme)
my_constants.kp = omgp / clight
my_constants.Rm = 40.e-6

plasma.density(x,y,z) = "ne * (1 + 4*(x^2 + y^2)/(kp^2 * Rm^4))"
```

The parser allows for advanced input settings

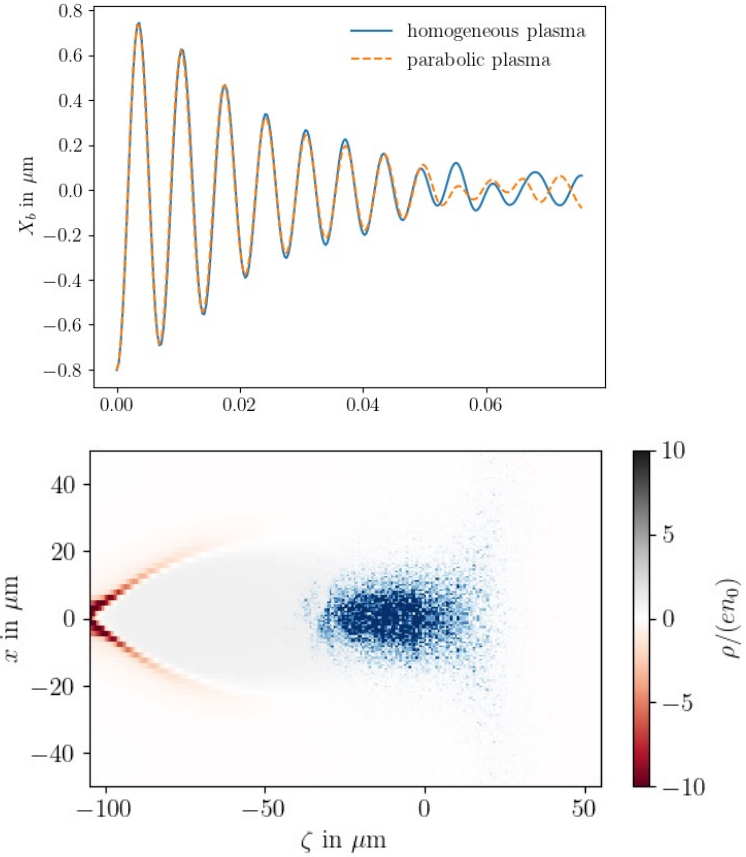
E.g., to generate an advanced plasma profile or tilting the beam

- The parser allows to define own parameters
- Many physical constants are predefined
  (clight, q_e, m_e, m_p, epsilon0, …)
- Many different math functions available

More information:
https://hipace.readthedocs.io/en/latest/run/parameters.html#parser
https://amrex-codes.github.io/amrex/docs_html/Basics.html#parser

```
my_constants.dx_per_dzeta = 0.1
driver.position_mean = "if(z<0, z*dx_per_dzeta, 0)" 0 0

# pre-calculating parameter might be needed in single precision
my_constants.eme = 3182.607354 # e**2/(m_e*epsilon_0)
my_constants.omgp = sqrt(ne*eme)
my_constants.kp = omgp / clight
my_constants.Rm = 40.e-6

plasma.density(x,y,z) = "ne * (1 + 4*(x^2 + y^2)/(kp^2 * Rm^4))"
```

1 minute later:



Results not conclusive, head erosion dominates

Let’s increase statistics and look at the tail!

Lighning-fast in-situ diagnostics for large beams

Data analysis can be computational heavy (> 1 billion beam particles, over 100s of time steps)
Solution: we are using high performance GPUs, let them do the work!

In-situ diagnostics calculate moments needed for energy spread, emittance, centroid, …
https://hipace.readthedocs.io/en/latest/run/parameters.html#in-situ-diagnostics

```
beams.insitu_period = 1
driver.num_particles = 10000000
```

Lighning-fast in-situ diagnostics for large beams

Data analysis can be computational heavy (> 1 billion beam particles, over 100s of time steps)
Solution: we are using high performance GPUs, let them do the work!

In-situ diagnostics calculate moments needed for energy spread, emittance, centroid, …
https://hipace.readthedocs.io/en/latest/run/parameters.html#in-situ-diagnostics

```
beams.insitu_period = 1
driver.num_particles = 10000000
```

Load in-situ diagnostics in analysis script

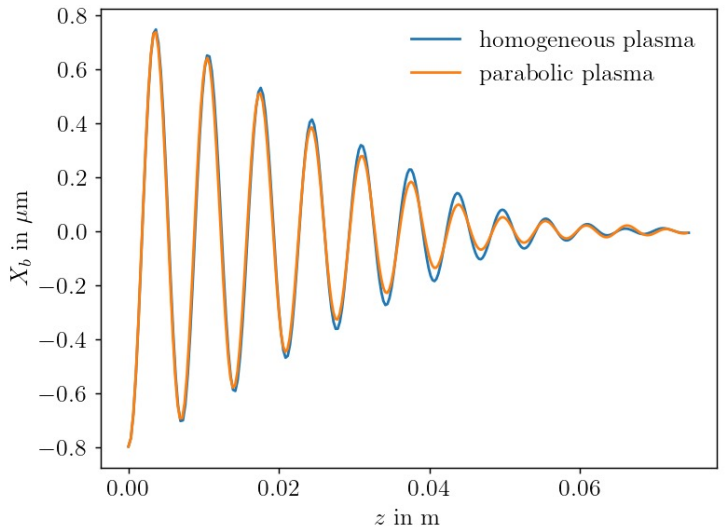
```
import sys
sys.path.append("../../software/hipace/tools/")
import read_insitu_diagnostics as diag
```

contains many useful functions
e.g., emittance_x(), …

```
data1 = diag.read_file('/path_to_your_insitu_data/homogeneous_insitu_data/reduced_driver.*.txt')
data2 = diag.read_file('/path_to_your_insitu_data/parabolic_insitu_data//reduced_driver.*.txt')
```

Plotting the data

```
fig, ax = plt.subplots()

ax.plot(data1["time"]*scc.c, data1["average"]["[x]"]*1e6, label='homogeneous plasma')
ax.plot(data2["time"]*scc.c, data2["average"]["[x]"]*1e6, label='parabolic plasma')

ax.set_xlabel("$z$ in m")
ax.set_ylabel("$X_b$ in $\mu$m")
ax.legend(frameon=False)
```



more statistics helped!

Lighning-fast in-situ diagnostics for large beams

Data analysis can be computational heavy (> 1 billion beam particles, over 100s of time steps)
Solution: we are using high performance GPUs, let them do the work!

In-situ diagnostics calculate moments needed for energy spread, emittance, centroid, … **per slice**
https://hipace.readthedocs.io/en/latest/run/parameters.html#in-situ-diagnostics

```
beams.insitu_period = 1
driver.num_particles = 10000000
```

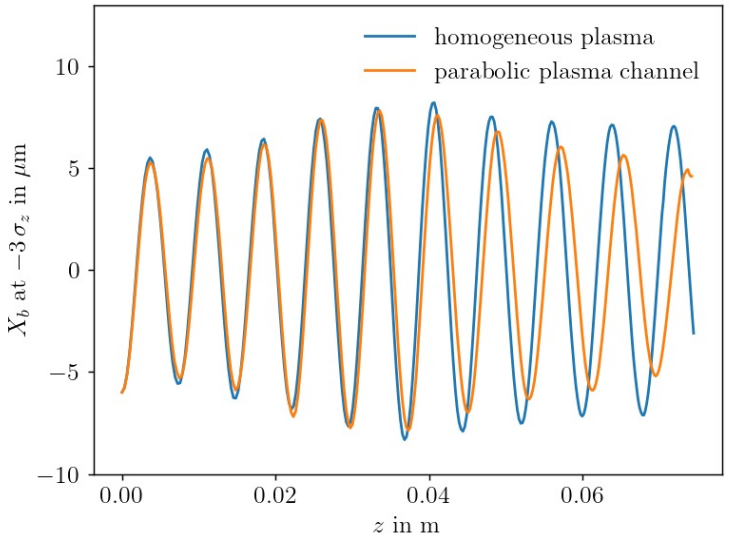
Looking at the tail of the bunch ($z = -3\sigma_z = -60$ µm)

```
print(diag.z_axis(data1)[67]*1e6)
```

```
-59.90234375000001
```

time step slice in z

```
fig, ax = plt.subplots()

ax.plot(data1["time"]*scc.c, data1["[x]"][:,67]*1e6, label='homogeneous plasma')
ax.plot(data2["time"]*scc.c, data2["[x]"][:,67]*1e6, label='parabolic plasma channel')

ax.set_xlabel("$z$ in m")
ax.set_ylabel("$X_b$ at $-3 \,\sigma_z$ in $\mu$m")
ax.legend(frameon=False)
ax.set_ylim(-10,13)
```

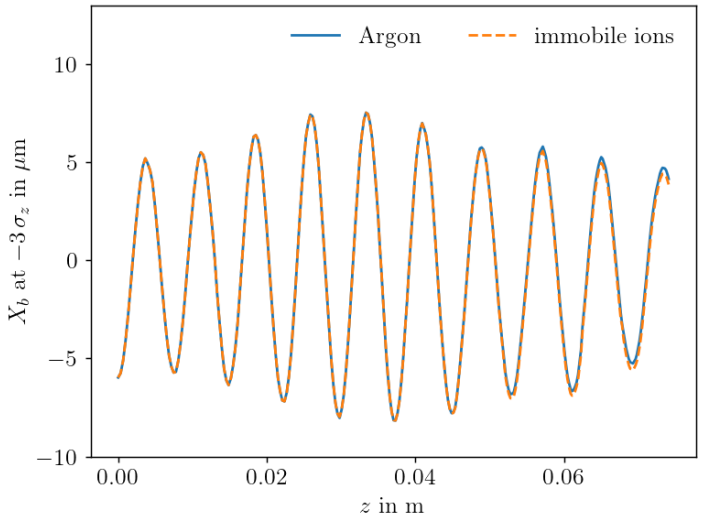


Add missing physics

Hosing instability can be mitigated by ion motion Mehrling et al., Phys. Rev. Lett. 121, 264802 (2018)
Let's assume an argon plasma (which is too heavy to move significantly in this case)

```
plasmas.names = plasma ions
ions.density(x,y,z) = ne * (1+4*(x^2+y^2)/(kp^2 * Rm^4 ))
ions.ppc = 2 2
ions.element = Ar
```

Everything looks good!
Ready for publication?



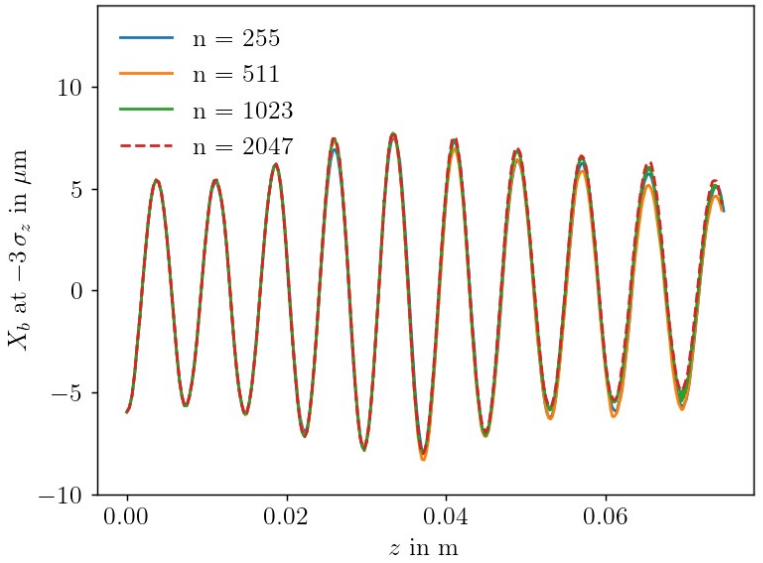
Convergence scans absolutely crucial

All results so far could be wrong!

Parameters that need to be checked
- transverse resolution
- longitudinal resolution
- number of plasma particles
- number of beam particles
- time step in the simulation
- box size!

Here, we do a transverse resolution scan:

```
mpiexec -n 4 -npernode 4 $HOME/software/hipace/build/bin/hipace inputs \
      amr.n_cell = 255 255 512 \
      hipace.file_prefix = output_512 \
      beams.insitu_file_prefix = insitu_output_512

mpiexec -n 4 -npernode 4 $HOME/software/hipace/build/bin/hipace inputs \
      amr.n_cell = 511 511 512 \
      hipace.file_prefix = output_512 \
      beams.insitu_file_prefix = insitu_output_512

mpiexec -n 4 -npernode 4 $HOME/software/hipace/build/bin/hipace inputs \
      amr.n_cell = 1023 1023 512 \
      hipace.file_prefix = output_1024 \
      beams.insitu_file_prefix = insitu_output_1024
```



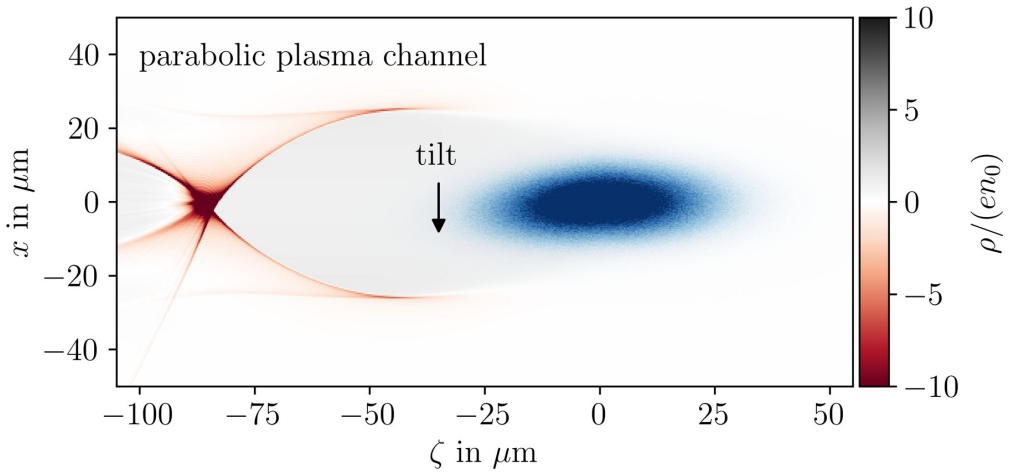
Result converges at ~ 1023 grid points

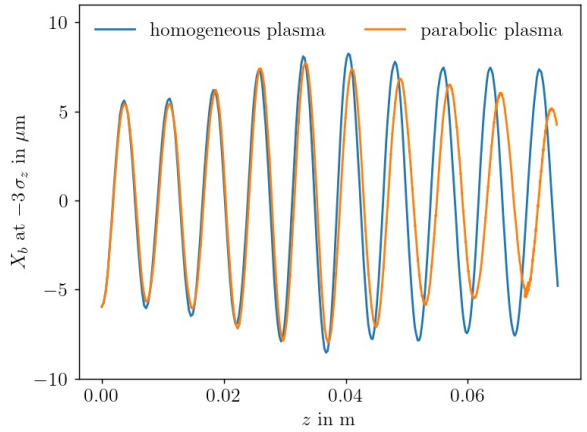
Run time: 5min on 8x A100 GPUs.

Finalizing the paper…

Only minor work left:

- Doing proper analytical calculations describing the physics
- Studying offsets in parabolic channel
- Polishing the plots
- Writing the manuscript





Finalizing the paper…

Only minor work left:

- Doing proper analytical calculations describing the physics
- Studying offsets in parabolic channel
- Polishing the plots
- Writing the manuscript

Please acknowledge HiPACE++ and cite the paper:

S. Diederichs, C. Benedetti, A. Huebl, R. Lehe, A. Myers, A. Sinn, J.-L. Vay, W. Zhang, M. Thévenet,
HiPACE++: A portable, 3D quasi-static particle-in-cell code.
Computer Physics Communications Volume 278, 108421, 2022

We value good scientific practice:
Please also publish your input scripts for reproducibility

Data availability.—The input scripts for the PIC simulations used in Figs. 2 and 4 are available online [35].

[35] S. Diederichs, C. Benedetti, E. Esarey, M. Thévenet, J. Osterhoff, and C. B. Schroeder, HiPACE++ input scripts for self-stabilizing positron acceleration in a plasma column, 10.5281/zenodo.6671371 (2022).

We are looking forward to seeing many papers written by you!