UNIVERSITY OF OSLO

The Research Council of Norway

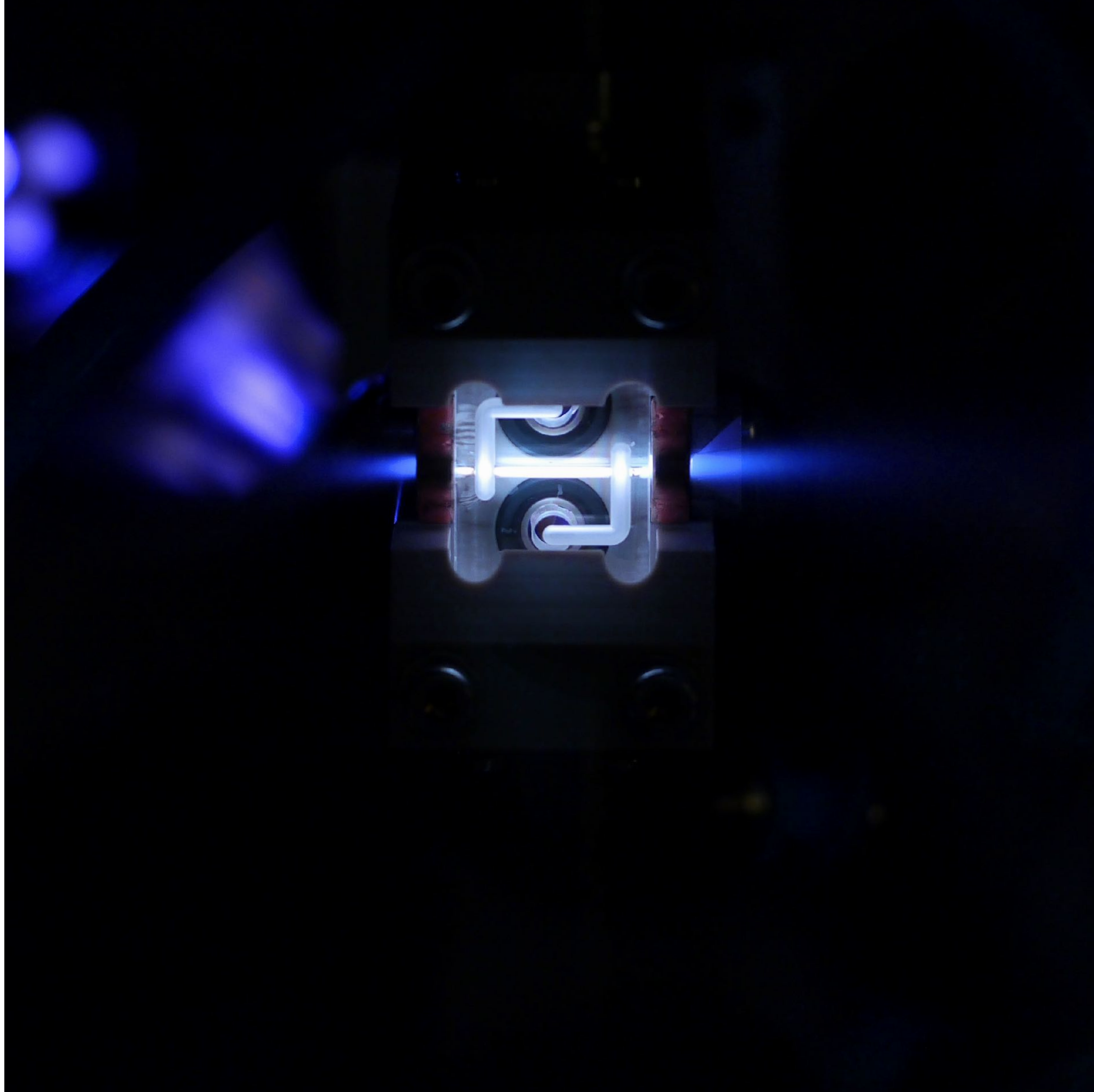
ABEL: A Start-to-End Simulation and Optimisation Framework for Plasma-Based Accelerators and Colliders

The Adaptable Beginning-to-End Linac simulation framework

Jian Bin Ben Chen

Department of Physics, University of Oslo

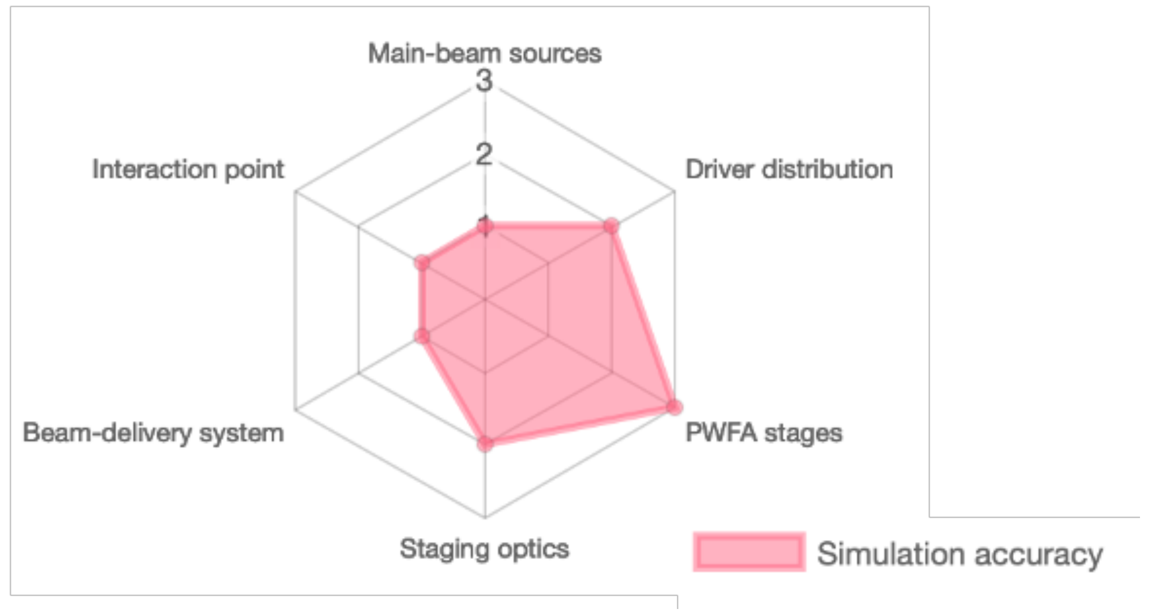
23 September 2025



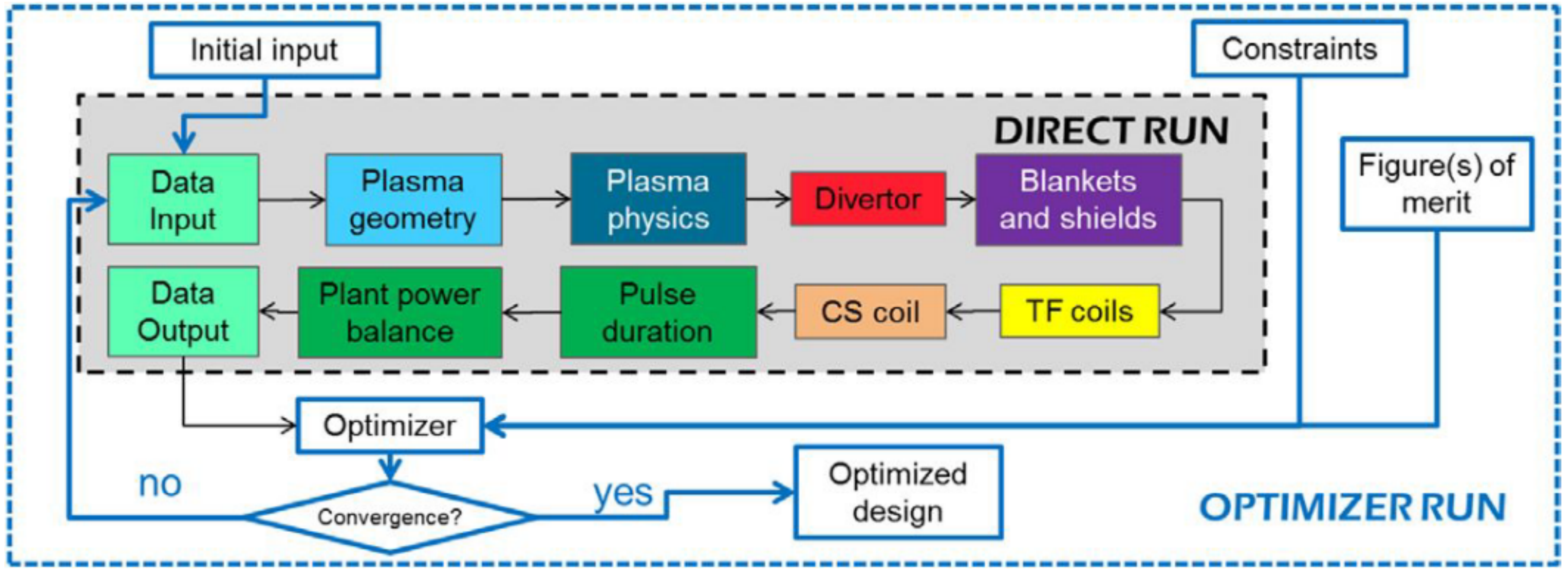
The Adaptable Beginning-to-End Linac simulation framework

Enabling agile design studies of plasma-based linacs/colliders

- A plasma-based linac or collider consists of many different beamline elements requiring specialised, often incompatible codes, limiting direct transfer of simulation outputs.
  - ABEL enables self-consistent simulation of entire beamlines by linking a suite of specialised codes using openPMD.
  - Supports third-party codes (HiPACE++, Wake-T, ELEGANT, GUINEA-PIG, CLICopti, ImpactX) as well as simplified built-in models.
  - Modular structure, written in Python.
  - Beamline elements represented as Python classes, with subclasses offering different speeds and levels of fidelity.

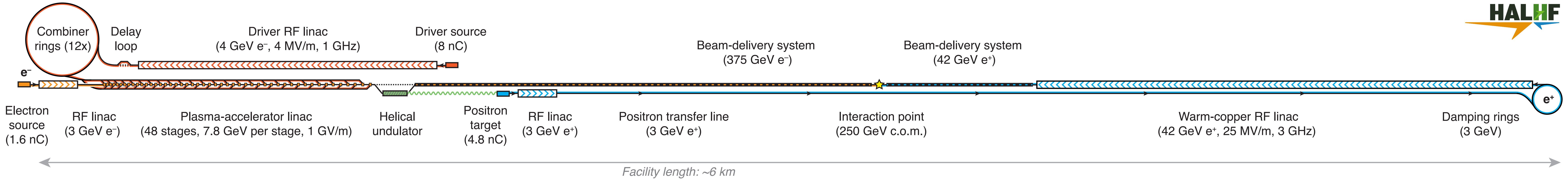


- Also functions as a system-level tool for optimisation and machine design similar to system codes in the nuclear fusion community.
  - Often rely on integrating reduced physics, engineering and economics models to model (or design) the entire fusion reactor facility.
  - Allows researchers to explore the feasibility, optimisation, and trade-offs of reactor designs at a whole-plant level.
  - But often lack modularity and flexibility.



SYCOMORE (modular) workflow

Structure

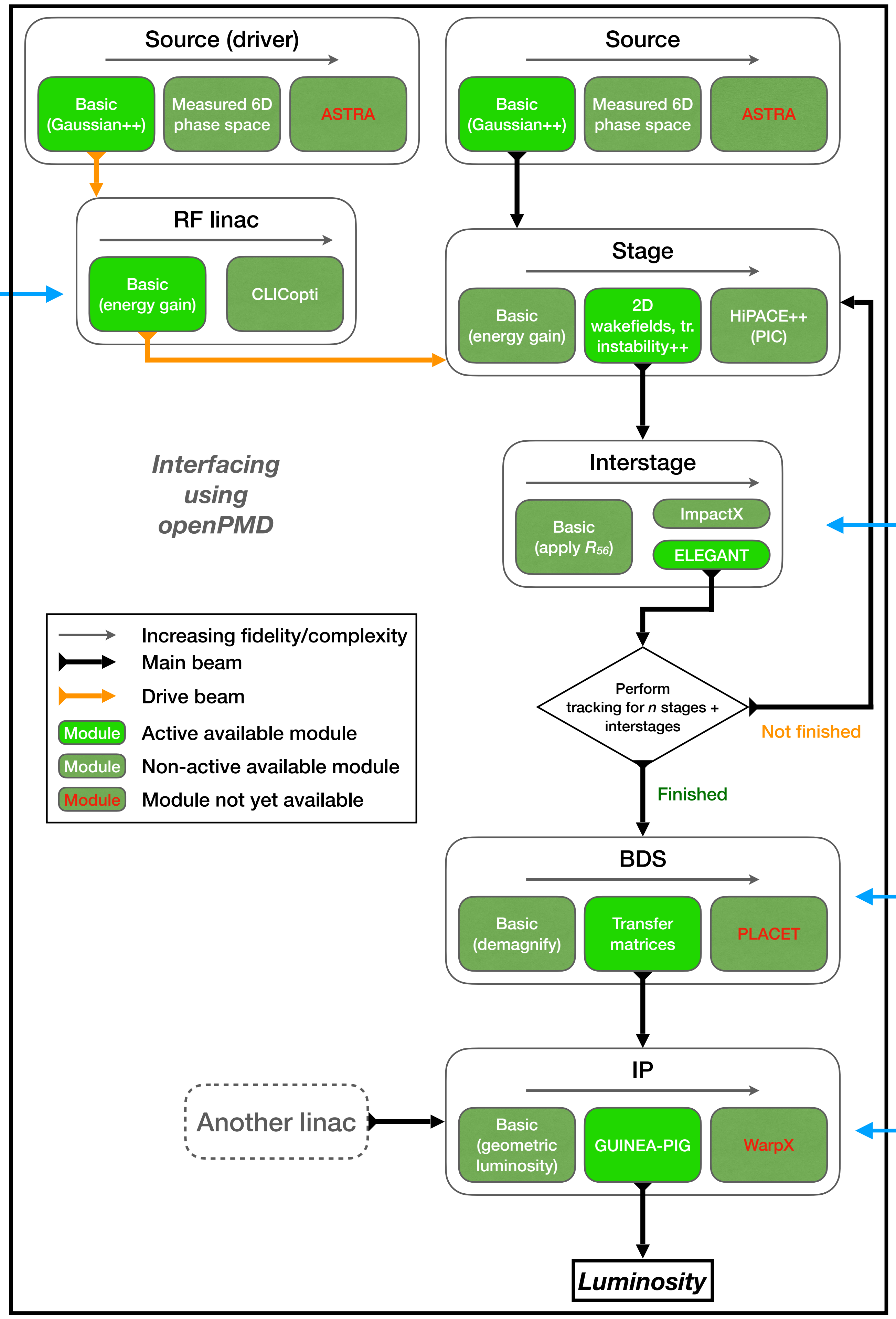


Generate the drive beam used to excite plasma waves in the plasma stages.

Accelerate the drive beam to desired energy before injecting into the plasma.

```
############# Define the stages #############
stage = StageReducedModels()
#stage = StageBasic()
stage.time_step_mod = 0.03
stage.nom_energy_gain = 7.8e9
stage.length_flattop = 7.8
stage.plasma_density = 6.0e+20
stage.driver_source = driver
stage.ramp_beta_mag = 5.0
stage.enable_tr_instability = True
stage.enable_radiation_reaction = True
stage.enable_ion_motion = True

############# Define interstages #############
interstage = InterstageElegant()
#interstage = InterstageBasic()
interstage.beta0 = lambda energy: stage.matched_beta_function(energy)
interstage.dipole_length = lambda energy: 1 * np.sqrt(energy/10e9)
interstage.dipole_field = lambda energy: np.min([0.52, 40e9/energy])

############# Define linac #############
linac = PlasmaLinac(source=main, stage=stage,
                    interstage=interstage, num_stages=48)
```

Can switch between implementations with a single line, not worrying about different input/outputs!



Generate the main beam to be accelerated.

Accelerate the beam.

Discard depleted drive beam, inject new drive beam and in-/out couple the main beam.

Focuses and steers the beams into the interaction point.

Beam interaction. Calculate luminosity and background.

Simulation capabilities

> Beam-tracking simulation in ABEL are performed as shots similar to experiments.

> Single shot

> Multi-shot:

  > Track beam multiple times through the same machine. Can be done in parallel

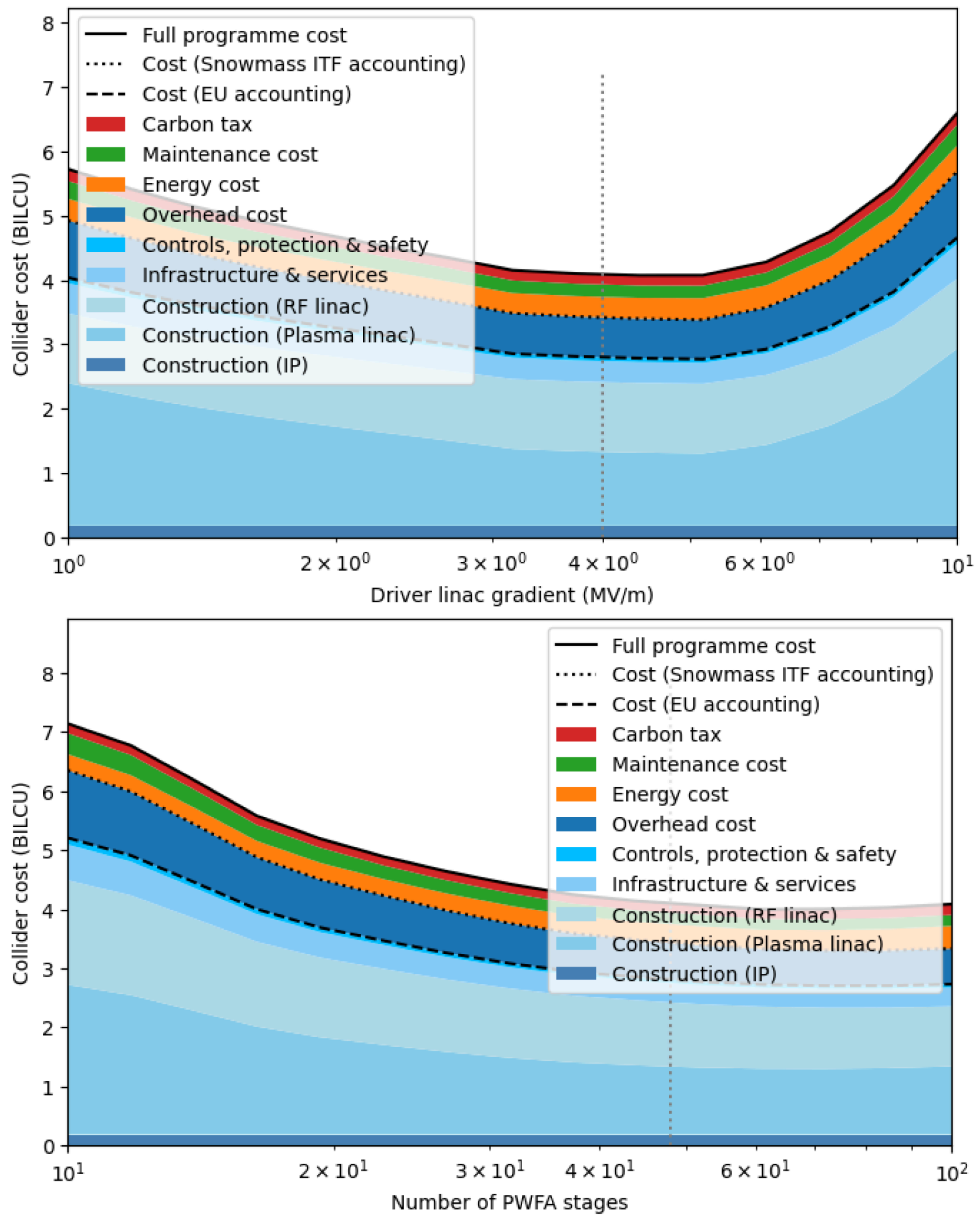
  > Easy to study stochastic shot-to-shot imperfections such as beam position, temporal and energy jitter.

> Automated parameter scans:

  > Supports multiple shots per scan step.

> Optimisation

  > Multi-dimensional parameter optimisation using Bayesian optimisation.



Example outputs of optimisation and cost model. C. A. Lindstrøm

E. Adli et al. *"HALHF: a hybrid, asymmetric, linear Higgs factory using plasma- and RF-based acceleration. Backup Document"*, arXiv:2503.23489

Also see arXiv:2505.21654

Example: single HALHF stage PIC simulation on LUMI

Define the driver source, plasma stage, and trailing-bunch source

```
[70]: # define driver
driver_source = SourceTrapezoid()
driver_source.charge = -8e-9 # [C]
driver_source.energy = 4.0e9 # [eV]
driver_source.rel_energy_spread = 0.01
driver_source.bunch_length = 1050e-6 # [m]
driver_source.gaussian_blur = 50e-6 # [m]
driver_source.current_head = 0.1e3
driver_source.z_offset = 1530e-6 # [m]
driver_source.emit_nx, driver_source.emit_ny = 50e-6, 100e-6 # [m rad]
driver_source.beta_x, driver_source.beta_y = 0.5, 0.5 # [m]
driver_source.num_particles = 1000000
driver_source.symmetrize = True

# define stage
#stage = StageQuasistatic2d()
stage = StageHipace()
stage.driver_source = driver_source
stage.ion_motion = True
stage.beam_ionization = True
stage.ion_species = 'He'
stage.nom_energy_gain = 7.75e9 # [eV]
stage.nom_accel_gradient = 1e9 # [GV/m]
stage.plasma_density = 6e20 # [m^-3]
stage.ramp_beta_mag = 10
stage.calculate_evolution = True
stage.upramp = stage.__class__()
stage.downramp = stage.__class__()

# define beam
main_beam_source = SourceTrapezoid()
main_beam_source.charge = -1e10 * SI.e # [C]
main_beam_source.energy = 58e9 # [eV]
main_beam_source.rel_energy_spread = 0.01
main_beam_source.bunch_length = 140e-6 # [m]
main_beam_source.z_offset = 0 # [m]
main_beam_source.current_head = -4.2e3 # [A]
main_beam_source.gaussian_blur = 10e-6 # [m]
main_beam_source.emit_nx = 90e-6 # [m rad]
main_beam_source.emit_ny = 0.315e-6 # [m rad]
main_beam_source.beta_x = stage.matched_beta_function(main_beam_source.energy)
main_beam_source.beta_y = main_beam_source.beta_x
main_beam_source.num_particles = int(driver_source.num_particles/10)

# define linac
linac = PlasmaLinac()
linac.source = main_beam_source
linac.stage = stage
```

- **Automated workflow integration**: Modify input files for third-party codes (e.g., HiPACE++), generate cluster job scripts according to specifications, and submit computation jobs.
- **Simple interface**: Full setup and control directly through Python.

Run simulation

```
linac.stage.upramp.num_nodes = 5
linac.stage.downramp.num_nodes = 5
linac.stage.num_nodes = 24
linac.run('halhf_single_stage_example_pic', overwrite=False); # takes 10 mins
```

```
Tracked #0  SourceTrapezoid     (s =    0.0 m) :   E =  58.0 GeV, Q = -1.60 nC, σz =  41.3 µm, σE =  1.0%, ε =  89.9/0.3 mm-mrad
>> Finished HiPACE++ (job 11255091): 100%|████████████████████| 166/166 [00:41<00:00,  4.04 steps/s]
>> Finished HiPACE++ (job 11255104): 100%|████████████████████| 6876/6876 [10:20<00:00, 11.09 steps/s]
>> Finished HiPACE++ (job 11255551): 100%|████████████████████| 306/306 [01:31<00:00,  3.33 steps/s]
  ... #2  StageHipace #1        (s =    8.3 m) :   E =  65.6 GeV, Q = -1.60 nC, σz =  41.3 µm, σE =  0.9%, ε =  93.1/0.3 mm-mrad
```

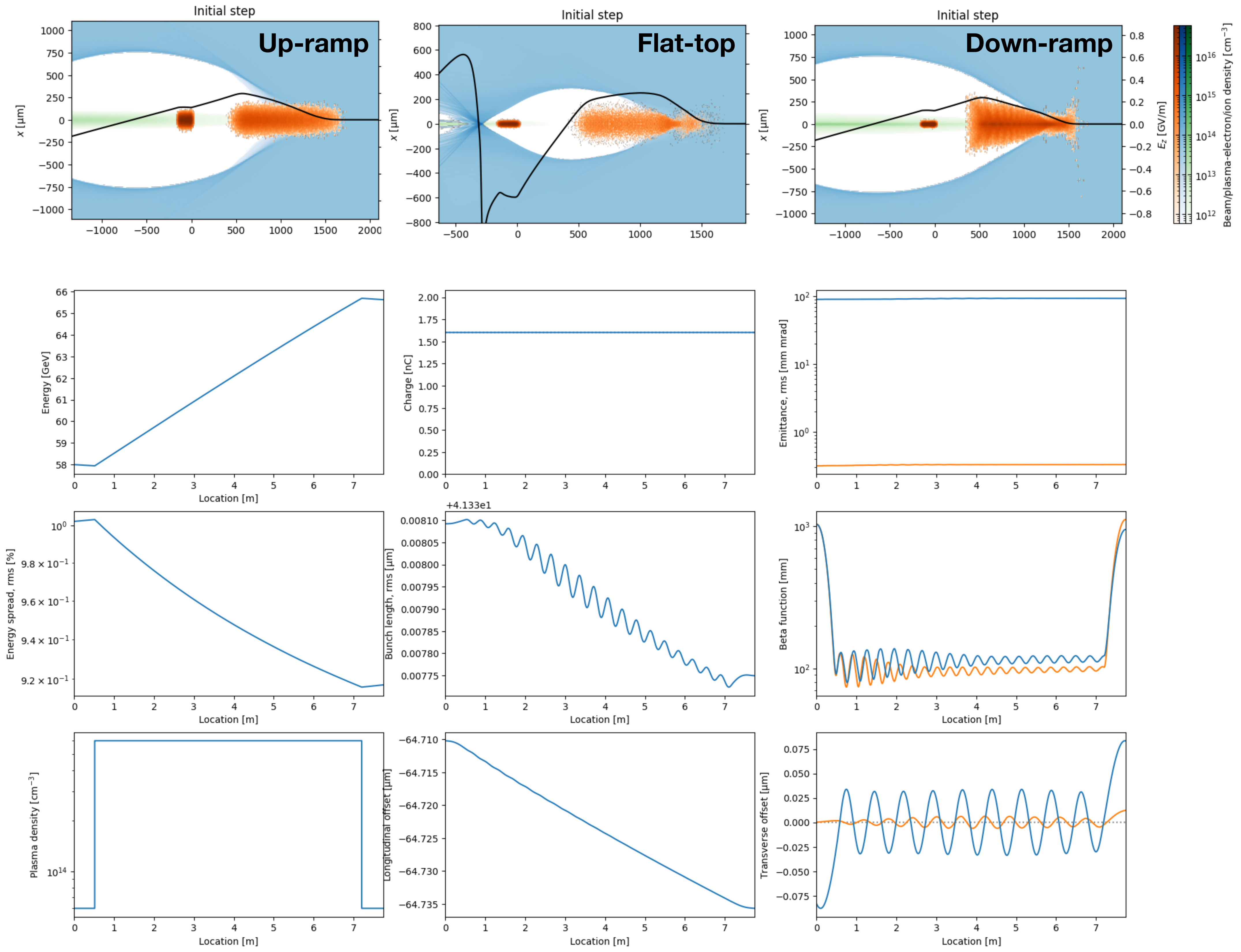
C. A. Lindstrøm (has been upgraded to running ramps and stage in one single job)

Example: single HALHF stage PIC simulation on LUMI

Define the driver source, plasma stage, and trailing-bunch source

```
[70]: # define driver
driver_source = SourceTrapezoid()
driver_source.charge = -8e-9 # [C]
driver_source.energy = 4.0e9 # [eV]
driver_source.rel_energy_spread = 0.01
driver_source.bunch_length = 1050e-6 # [m]
driver_source.gaussian_blur = 50e-6 # [m]
driver_source.current_head = 0.1e3
driver_source.z_offset = 1530e-6 # [m]
driver_source.emit_nx, driver_source.emit_ny = 50e-6, 100e-6 # [m rad]
driver_source.beta_x, driver_source.beta_y = 0.5, 0.5 # [m]
driver_source.num_particles = 1000000
driver_source.symmetrize = True

# define stage
#stage = StageQuasistatic2d()
stage = StageHipace()
stage.driver_source = driver_source
stage.ion_motion = True
stage.beam_ionization = True
stage.ion_species = 'He'
stage.nom_energy_gain = 7.75e9 # [eV]
stage.nom_accel_gradient = 1e9 # [GV/m]
stage.plasma_density = 6e20 # [m^-3]
stage.ramp_beta_mag = 10
stage.calculate_evolution = True
stage.upramp = stage.__class__()
stage.downramp = stage.__class__()

# define beam
main_beam_source = SourceTrapezoid()
main_beam_source.charge = -1e10 * SI.e # [C]
main_beam_source.energy = 58e9 # [eV]
main_beam_source.rel_energy_spread = 0.01
main_beam_source.bunch_length = 140e-6 # [m]
main_beam_source.z_offset = 0 # [m]
main_beam_source.current_head = -4.2e3 # [A]
main_beam_source.gaussian_blur = 10e-6 # [m]
main_beam_source.emit_nx = 90e-6 # [m rad]
main_beam_source.emit_ny = 0.315e-6 # [m rad]
main_beam_source.beta_x = stage.matched_beta_function(main_beam_source.energy)
main_beam_source.beta_y = main_beam_source.beta_x
main_beam_source.num_particles = int(driver_source.num_particles/10)

# define linac
linac = PlasmaLinac()
linac.source = main_beam_source
linac.stage = stage
```

> **Data handling**: Extract simulation outputs and perform analysis natively within ABEL.



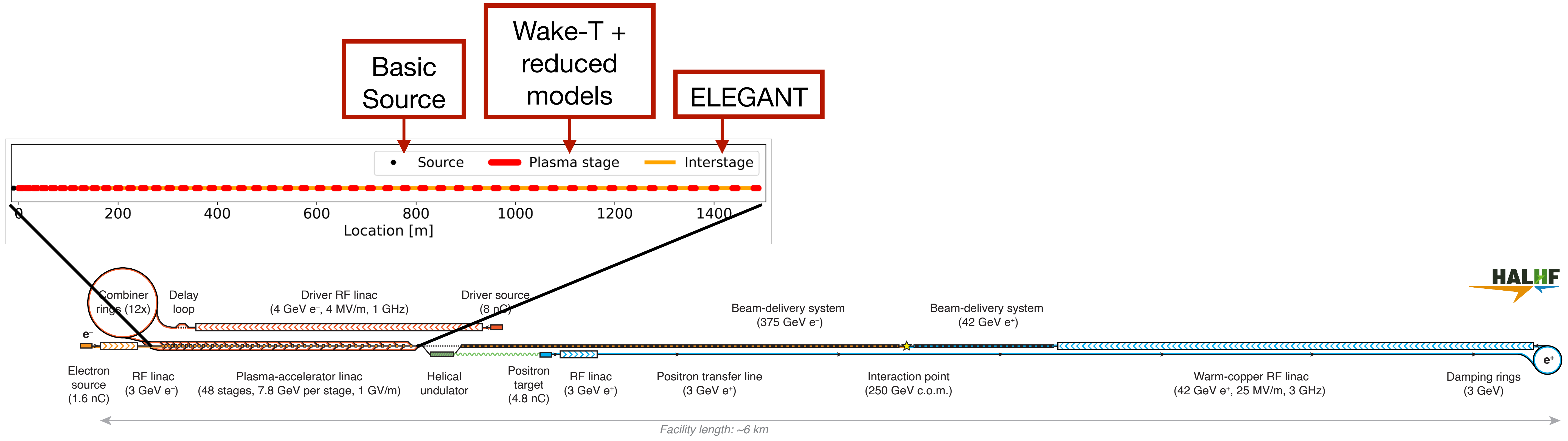
C. A. Lindstrøm

Example: HALHF plasma linac using reduced models

```
############# Define drive beam source #############
driver = SourceTrapezoid()
driver.current_head = 0.1e3                                          # [A]
driver.bunch_length = 1050e-6                                        # [m]
driver.z_offset = 1615e-6                                            # [m]
driver.num_particles = 30000
driver.charge = 5.0e10 * -SI.e                                       # [C]
driver.energy = 4.0e9                                                # [eV]
driver.gaussian_blur = 50e-6                                         # [m]
driver.rel_energy_spread = 0.01
driver.emit_nx, driver.emit_ny = 50e-6, 100e-6                       # [m rad]
driver.beta_x, driver.beta_y = 0.5, 0.5                              # [m]
driver.jitter.x = 100e-9                                             # [m], std
driver.jitter.y = 100e-9                                             # [m], std
driver.symmetrize = True

############# Define main beam source #############
main = SourceBasic()
main.bunch_length = 40.0e-06                                         # [m], rms
main.num_particles = 10000
main.charge = -e * 1.0e10                                            # [C]
main.energy = 3e9                                                    # [eV]
main.rel_energy_spread = 0.02                                        # Relative rms energy spread
main.emit_nx, main.emit_ny = 15e-6, 0.1e-6                           # [m rad]
main.beta_x = beta_matched(plasma_density, main.energy) * 10.0       # [m]
main.beta_y = main.beta_x                                            # [m]
main.z_offset = 0.00e-6                                              # [m]
main.symmetrize_6d = True

############# Define the stages #############
stage = StageReducedModels()
stage.time_step_mod = 0.04                                           # In units of betatron wavelengths/c.
stage.nom_energy_gain = 7.8e9                                        # [eV]
stage.length_flattop = 7.8                                           # [m]
stage.plasma_density = 6.0e+20                                       # [m^-3]
stage.driver_source = driver
stage.ramp_beta_mag = 10.0
stage.enable_tr_instability = True
stage.enable_radiation_reaction = True
stage.enable_ion_motion = True
stage.ion_charge_num = 1.0                                           # [e]
stage.ion_mass = 6.646477e-27                                        # [kg], He mass
stage.upramp = PlasmaRamp()
stage.downramp = PlasmaRamp()

############# Define interstages #############
interstage = InterstageElegant()
interstage.beta0 = lambda energy: stage.matched_beta_function(energy)
interstage.length_dipole = lambda energy: 1.0 * np.sqrt(energy/10e9)  # [m(eV)], energy-dependent length
interstage.field_dipole = lambda energy: np.min([0.52, 40e9/energy])  # [T]

############# Define linac #############
linac = PlasmaLinac(source=main, stage=stage, interstage=interstage, num_stages=48)
```

*The standard interstage lattice has been changed since this simulation, such that this setup no longer produces the same results.

- **Reduced models**: transverse intra-beam instability, radiation reaction, ion motion
- **Multi-shot**: 5 shots for studying driver xy-jitter.
- **Affordable computational cost**: Run on a laptop.
- **Flexible data handling**: Extensive diagnostics at single stage, linac single shot and linac multi-shot level.

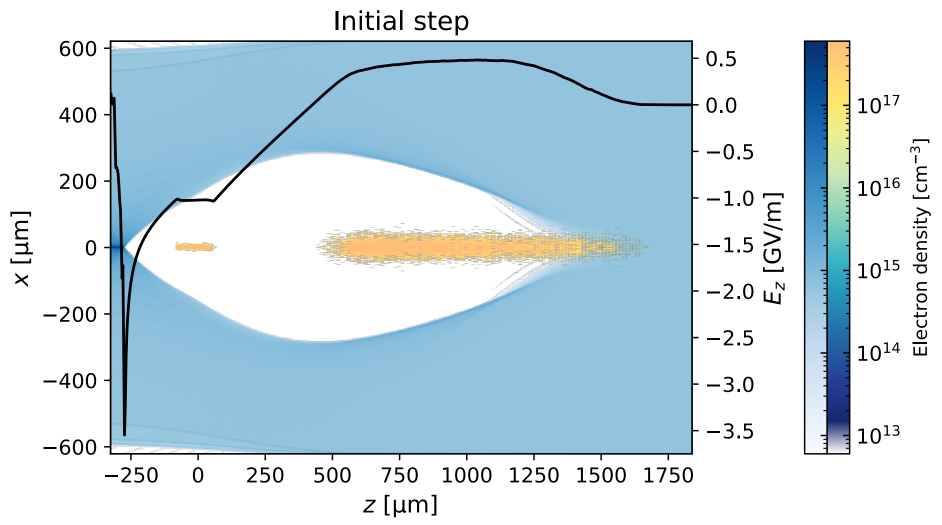


Example: HALHF plasma linac using reduced models

```
############# Define drive beam source #############
driver = SourceTrapezoid()
driver.current_head = 0.1e3                                      # [A]
driver.bunch_length = 1050e-6                                    # [m]
driver.z_offset = 1615e-6                                        # [m]
driver.num_particles = 30000
driver.charge = 5.0e10 * -SI.e                                   # [C]
driver.energy = 4.0e9                                            # [eV]
driver.gaussian_blur = 50e-6                                     # [m]
driver.rel_energy_spread = 0.01
driver.emit_nx, driver.emit_ny = 50e-6, 100e-6                   # [m rad]
driver.beta_x, driver.beta_y = 0.5, 0.5                          # [m]
driver.jitter.x = 100e-9                                         # [m], std
driver.jitter.y = 100e-9                                         # [m], std
driver.symmetrize = True

############# Define main beam source #############
main = SourceBasic()
main.bunch_length = 40.0e-06                                     # [m], rms
main.num_particles = 10000
main.charge = -e * 1.0e10                                        # [C]
main.energy = 3e9                                                # [eV]
main.rel_energy_spread = 0.02                                    # Relative rms energy spread
main.emit_nx, main.emit_ny = 15e-6, 0.1e-6                       # [m rad]
main.beta_x = beta_matched(plasma_density, main.energy) * 10.0   # [m]
main.beta_y = main.beta_x                                        # [m]
main.z_offset = 0.00e-6                                          # [m]
main.symmetrize_6d = True

############# Define the stages #############
stage = StageReducedModels()
stage.time_step_mod = 0.04                                       # In units of betatron wavelengths/c.
stage.nom_energy_gain = 7.8e9                                    # [eV]
stage.length_flattop = 7.8                                       # [m]
stage.plasma_density = 6.0e+20                                   # [m^-3]
stage.driver_source = driver
stage.ramp_beta_mag = 10.0
stage.enable_tr_instability = True
stage.enable_radiation_reaction = True
stage.enable_ion_motion = True
stage.ion_charge_num = 1.0                                       # [e]
stage.ion_mass = 6.646477e-27                                    # [kg], He mass
stage.upramp = PlasmaRamp()
stage.downramp = PlasmaRamp()

############# Define interstages #############
interstage = InterstageElegant()
interstage.beta0 = lambda energy: stage.matched_beta_function(energy)
interstage.length_dipole = lambda energy: 1.0 * np.sqrt(energy/10e9)   # [m(eV)], energy-dependent length
interstage.field_dipole = lambda energy: np.min([0.52, 40e9/energy])   # [T]

############# Define linac #############
linac = PlasmaLinac(source=main, stage=stage, interstage=interstage, num_stages=48)
```

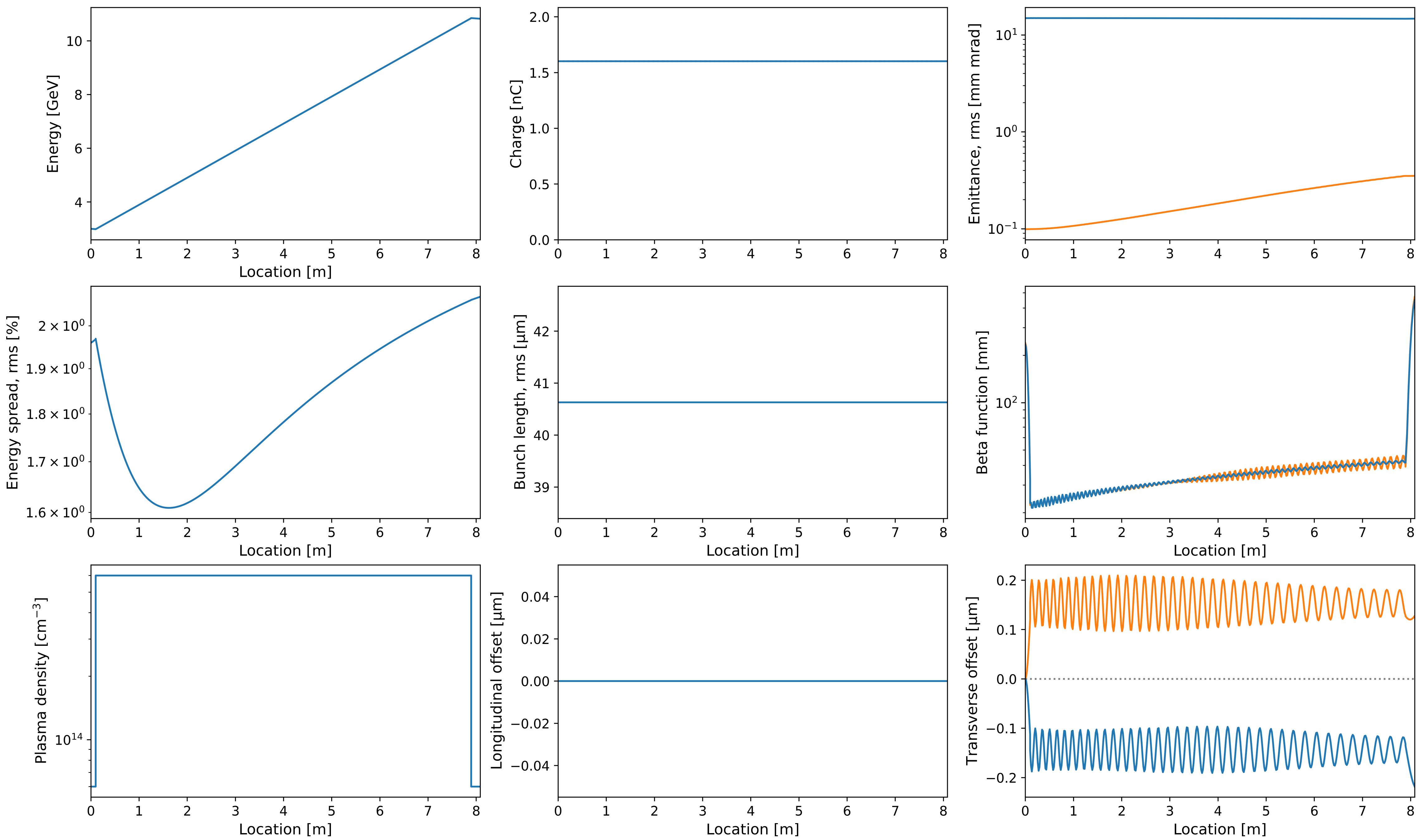
*The standard interstage lattice has been changed since this simulation, such that this setup no longer produces the same results.

> **Single stage diagnostics**:



Input beam, last stage

Stage 1, beam

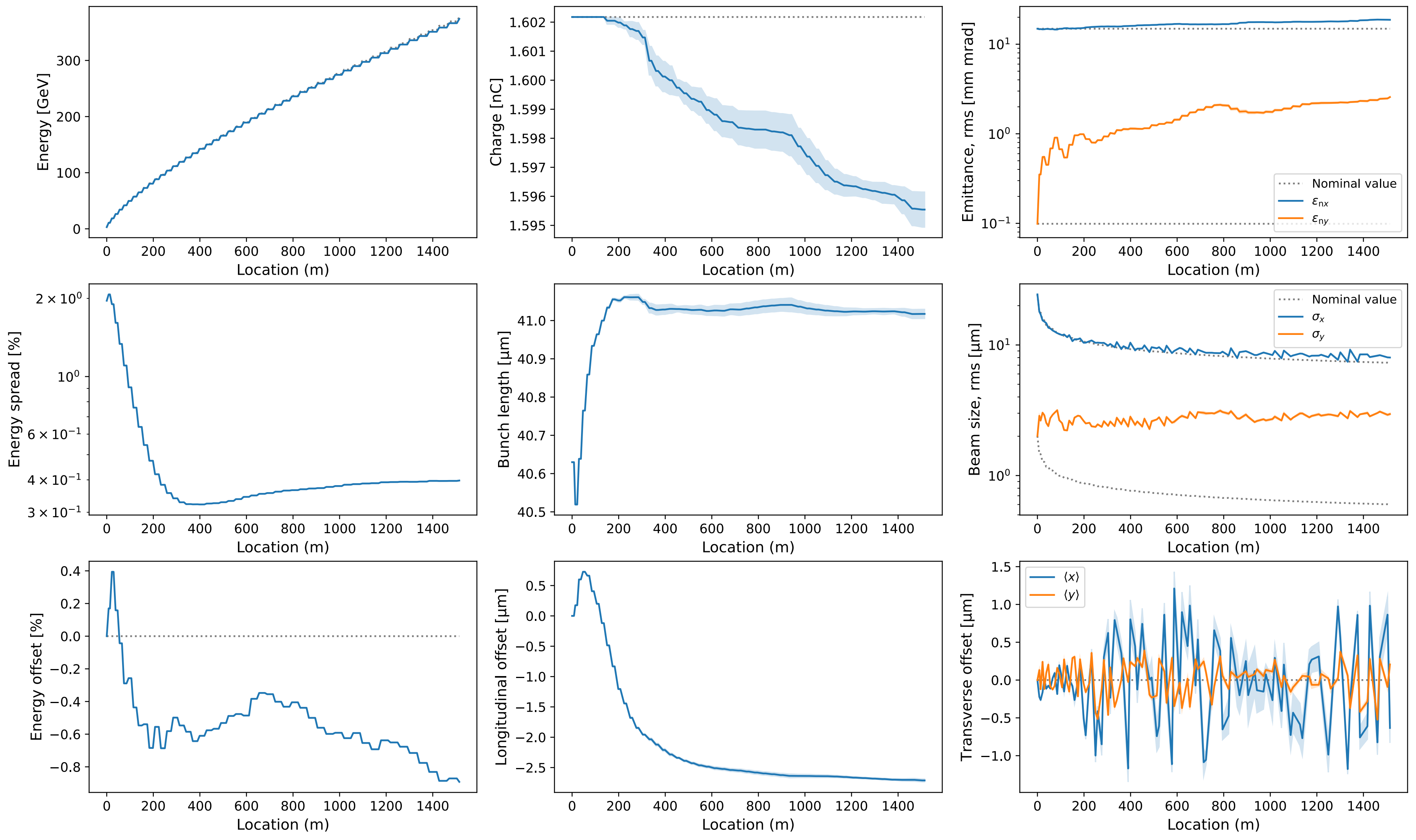


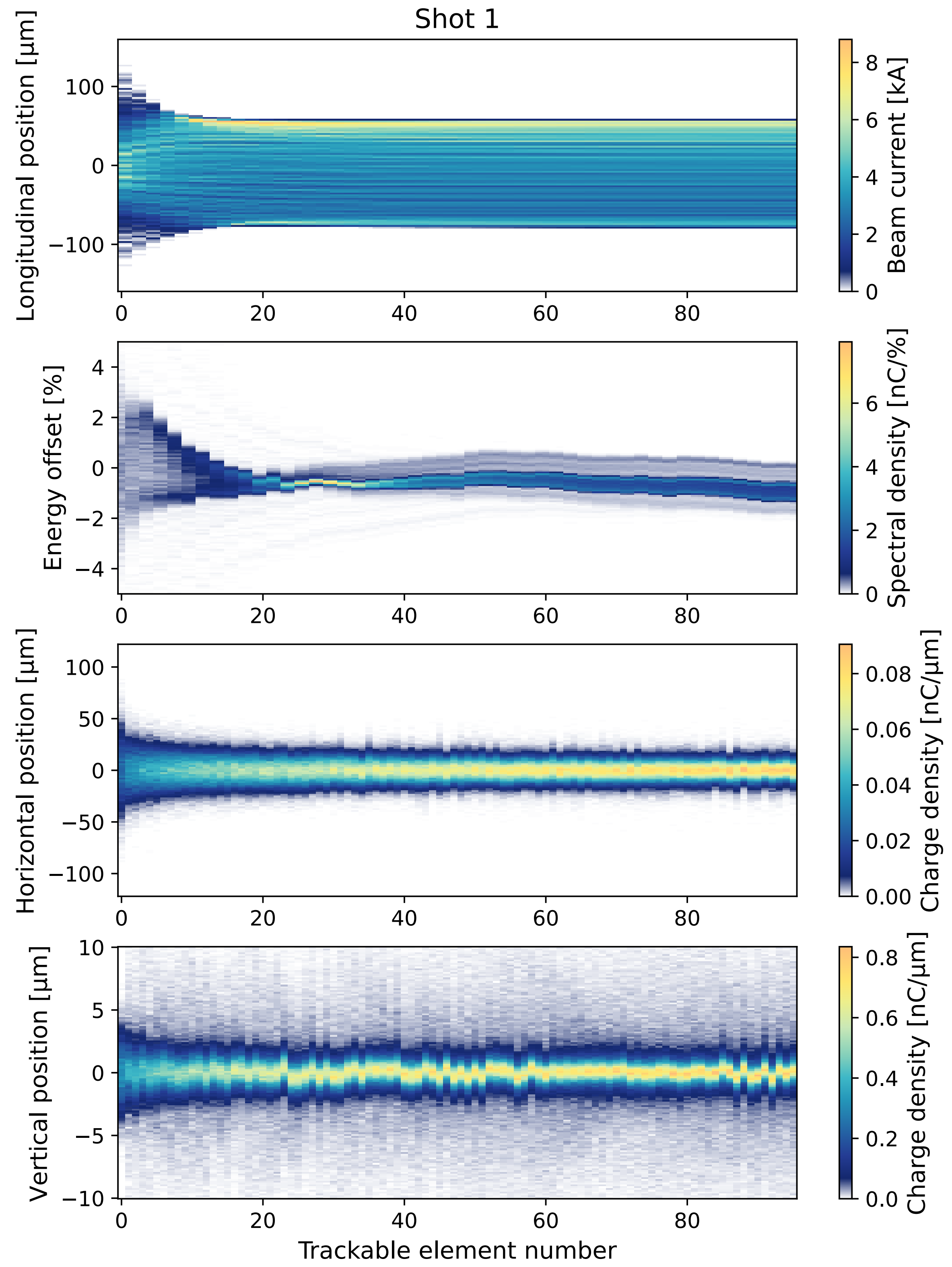
Example: HALHF plasma linac using reduced models

```python
############# Define drive beam source #############
driver = SourceTrapezoid()
driver.current_head = 0.1e3                                  # [A]
driver.bunch_length = 1050e-6                                # [m]
driver.z_offset = 1615e-6                                    # [m]
driver.num_particles = 30000
driver.charge = 5.0e10 * -SI.e                               # [C]
driver.energy = 4.0e9                                        # [eV]
driver.gaussian_blur = 50e-6                                 # [m]
driver.rel_energy_spread = 0.01
driver.emit_nx, driver.emit_ny = 50e-6, 100e-6               # [m rad]
driver.beta_x, driver.beta_y = 0.5, 0.5                      # [m]
driver.jitter.x = 100e-9                                     # [m], std
driver.jitter.y = 100e-9                                     # [m], std
driver.symmetrize = True

############# Define main beam source #############
main = SourceBasic()
main.bunch_length = 40.0e-06                                 # [m], rms
main.num_particles = 10000
main.charge = -e * 1.0e10                                    # [C]
main.energy = 3e9                                            # [eV]
main.rel_energy_spread = 0.02                                # Relative rms energy spread
main.emit_nx, main.emit_ny = 15e-6, 0.1e-6                   # [m rad]
main.beta_x = beta_matched(plasma_density, main.energy) * 10.0   # [m]
main.beta_y = main.beta_x                                    # [m]
main.z_offset = 0.00e-6                                      # [m]
main.symmetrize_6d = True

############# Define the stages #############
stage = StageReducedModels()
stage.time_step_mod = 0.04                                   # In units of betatron wavelengths/c.
stage.nom_energy_gain = 7.8e9                                # [eV]
stage.length_flattop = 7.8                                   # [m]
stage.plasma_density = 6.0e+20                               # [m^-3]
stage.driver_source = driver
stage.ramp_beta_mag = 10.0
stage.enable_tr_instability = True
stage.enable_radiation_reaction = True
stage.enable_ion_motion = True
stage.ion_charge_num = 1.0                                   # [e]
stage.ion_mass = 6.646477e-27                                # [kg], He mass
stage.upramp = PlasmaRamp()
stage.downramp = PlasmaRamp()

############# Define interstages #############
interstage = InterstageElegant()
interstage.beta0 = lambda energy: stage.matched_beta_function(energy)
interstage.length_dipole = lambda energy: 1.0 * np.sqrt(energy/10e9)   # [m(eV)], energy-dependent length
interstage.field_dipole = lambda energy: np.min([0.52, 40e9/energy])   # [T]

############# Define linac #############
linac = PlasmaLinac(source=main, stage=stage, interstage=interstage, num_stages=48)
```

*The standard interstage lattice has been changed since this simulation, such that this setup no longer produces the same results.

- **Linac level diagnostics**:





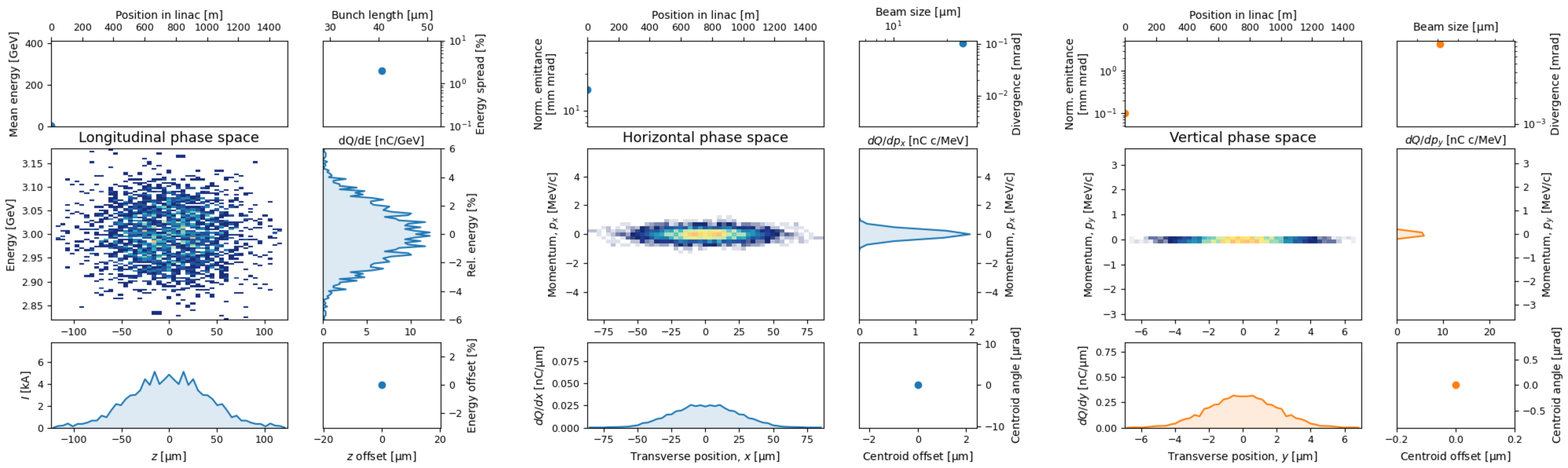
More on transverse instability and tolerances, see E. Adli's talk tomorrow 17:20, Sala Biodola

Example: HALHF plasma linac using reduced models

```
############# Define drive beam source #############
driver = SourceTrapezoid()
driver.current_head = 0.1e3                                   # [A]
driver.bunch_length = 1050e-6                                 # [m]
driver.z_offset = 1615e-6                                     # [m]
driver.num_particles = 30000
driver.charge = 5.0e10 * -SI.e                                # [C]
driver.energy = 4.0e9                                         # [eV]
driver.gaussian_blur = 50e-6                                  # [m]
driver.rel_energy_spread = 0.01
driver.emit_nx, driver.emit_ny = 50e-6, 100e-6                # [m rad]
driver.beta_x, driver.beta_y = 0.5, 0.5                       # [m]
driver.jitter.x = 100e-9                                      # [m], std
driver.jitter.y = 100e-9                                      # [m], std
driver.symmetrize = True

############# Define main beam source #############
main = SourceBasic()
main.bunch_length = 40.0e-06                                  # [m], rms
main.num_particles = 10000
main.charge = -e * 1.0e10                                     # [C]
main.energy = 3e9                                             # [eV]
main.rel_energy_spread = 0.02                                 # Relative rms energy spread
main.emit_nx, main.emit_ny = 15e-6, 0.1e-6                    # [m rad]
main.beta_x = beta_matched(plasma_density, main.energy) * 10.0  # [m]
main.beta_y = main.beta_x                                     # [m]
main.z_offset = 0.00e-6                                       # [m]
main.symmetrize_6d = True

############# Define the stages #############
stage = StageReducedModels()
stage.time_step_mod = 0.04                                    # In units of betatron wavelengths/c.
stage.nom_energy_gain = 7.8e9                                 # [eV]
stage.length_flattop = 7.8                                    # [m]
stage.plasma_density = 6.0e+20                                # [m^-3]
stage.driver_source = driver
stage.ramp_beta_mag = 10.0
stage.enable_tr_instability = True
stage.enable_radiation_reaction = True
stage.enable_ion_motion = True
stage.ion_charge_num = 1.0                                    # [e]
stage.ion_mass = 6.646477e-27                                 # [kg], He mass
stage.upramp = PlasmaRamp()
stage.downramp = PlasmaRamp()

############# Define interstages #############
interstage = InterstageElegant()
interstage.beta0 = lambda energy: stage.matched_beta_function(energy)
interstage.length_dipole = lambda energy: 1.0 * np.sqrt(energy/10e9)   # [m(eV)], energy-dependent length
interstage.field_dipole = lambda energy: np.min([0.52, 40e9/energy])   # [T]

############# Define linac #############
linac = PlasmaLinac(source=main, stage=stage, interstage=interstage, num_stages=48)
```

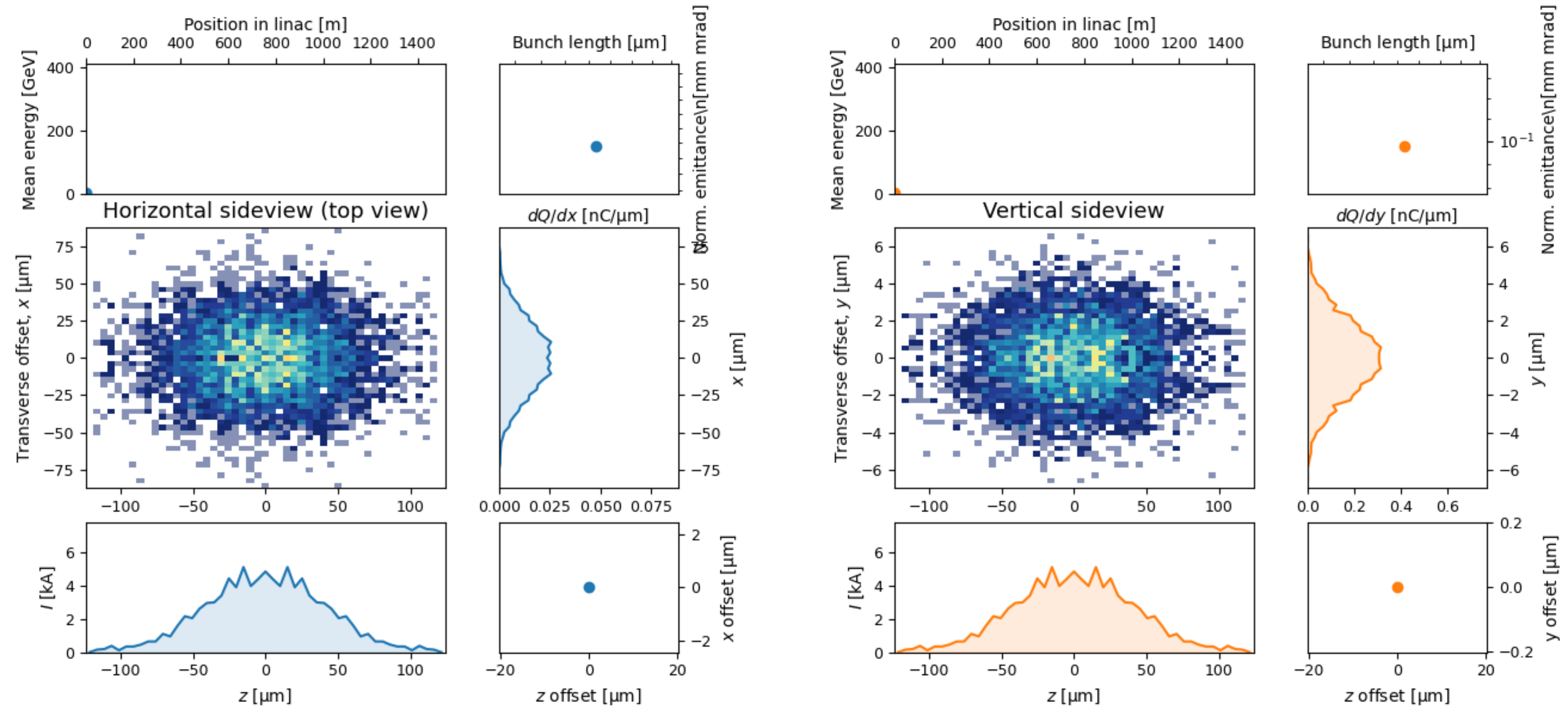
*The standard interstage lattice has been changed since this simulation, such that this setup no longer produces the same results.

> **Linac level diagnostics**:



Self-correction is applied through the interstages!

See C. A. Lindstrøm and P. Drobniak's contributions

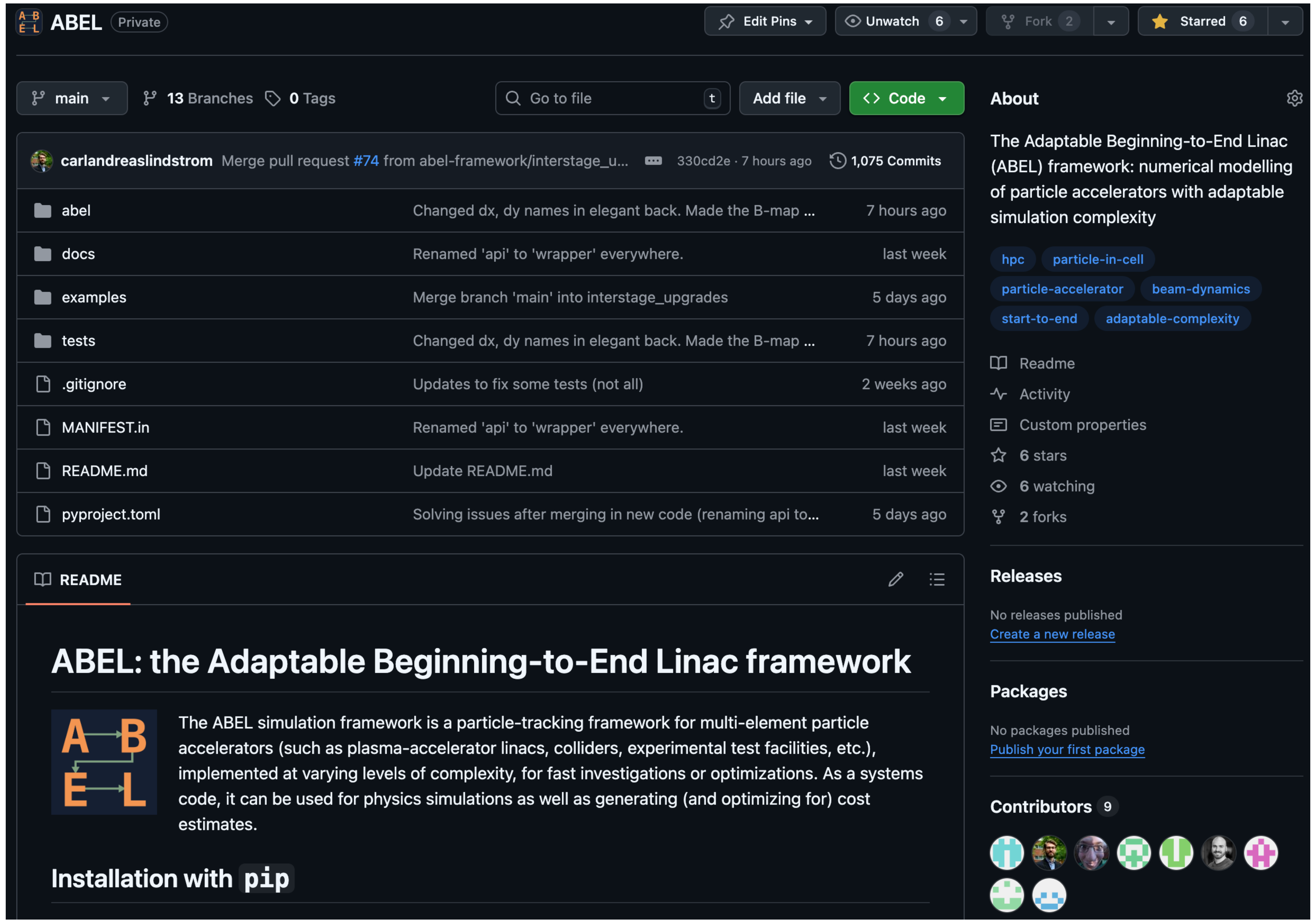


Summary

- The adaptable and modular approach of ABEL allows for flexible simulation runs of entire machines with desired accuracy and speed.
- Extensive simulation and diagnostic capabilities.
- Integrated workflow: especially convenient for simulations on cluster.
- Extendable to other applications, including FELs, strong-field QED experiments, and accelerator test facilities.
- IPAC 2025 Proceeding: https://arxiv.org/abs/2505.22415
- Repository: https://github.com/abel-framework/ABEL
- Open source (GPL 3.0)!
- Beta release on Thursday!

Come to the ABEL tutorial on Thursday 14:30 at Sala Elena for hands-on experience!

N. H. Abel



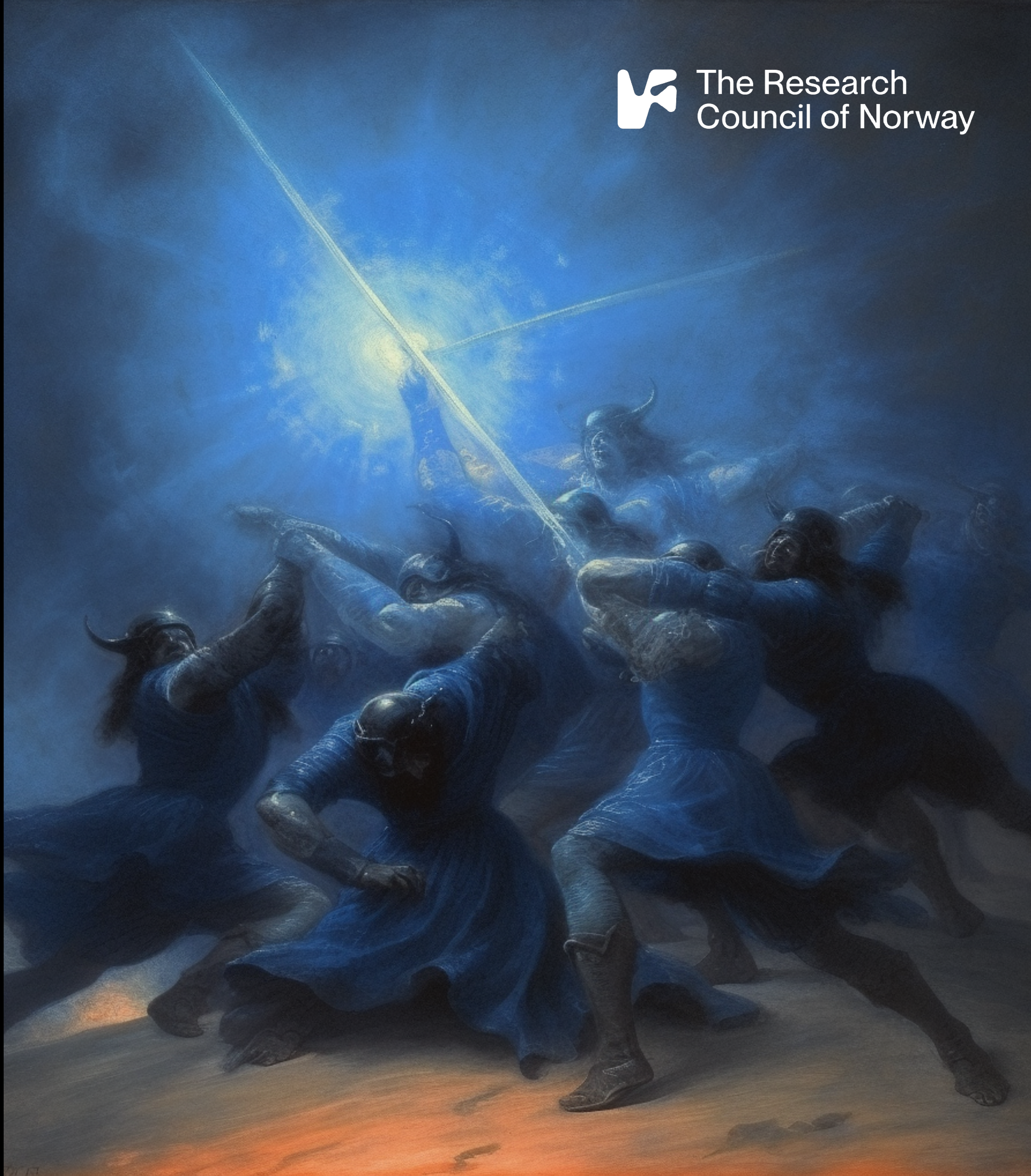
Acknowledgements

Oslo accelerator group ABEL contributors:
Erik Adli, Kyrre N. Sjøbæk, Carl A. Lindstrøm, J. B. Ben Chen, Ole Gunnar Finnerud, Daniel Kalvik, Pierre Drobniak, Felipe Peña, Eir E. Hørlyk

External contributors (LBNL):
Axel Huebl, Chad Mitchell

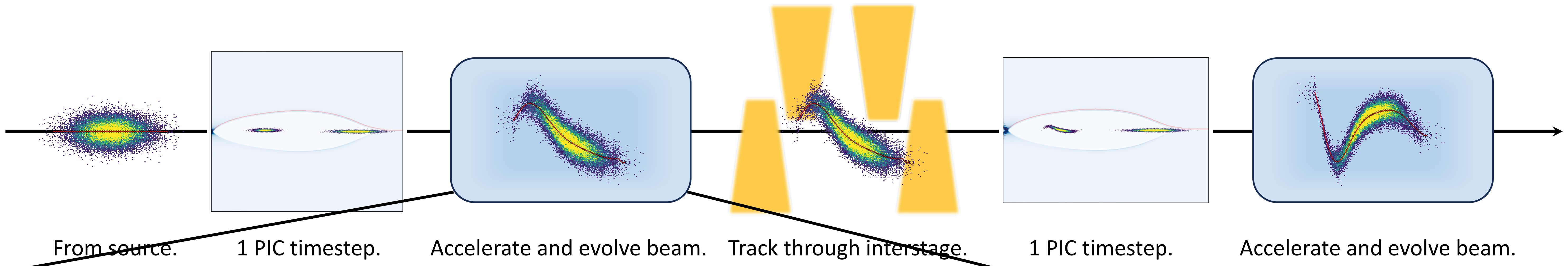
Funding:
European Research Council (ERC)
The Research Council of Norway

UNIVERSITY
OF OSLO



Simplified transverse wake instability model

- Wakefield formalism has been used in CLIC to study the limitations on charge and efficiency.
- Ansatz: for small offsets/perturbations, transverse instability in PWFA should behave similarly to BBU in conventional accelerators.

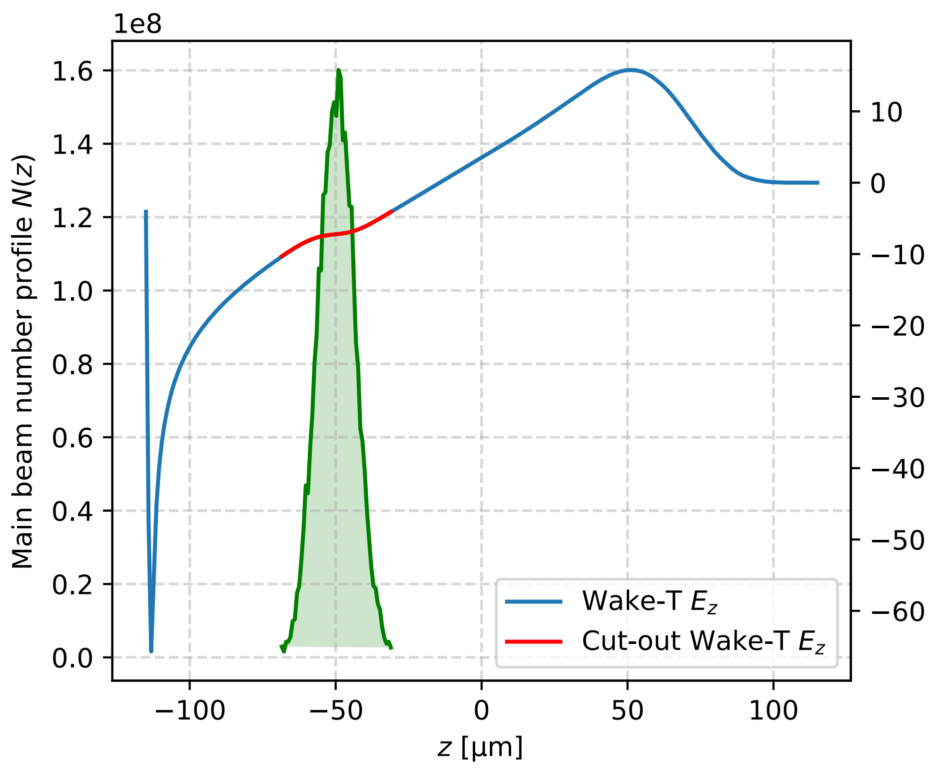


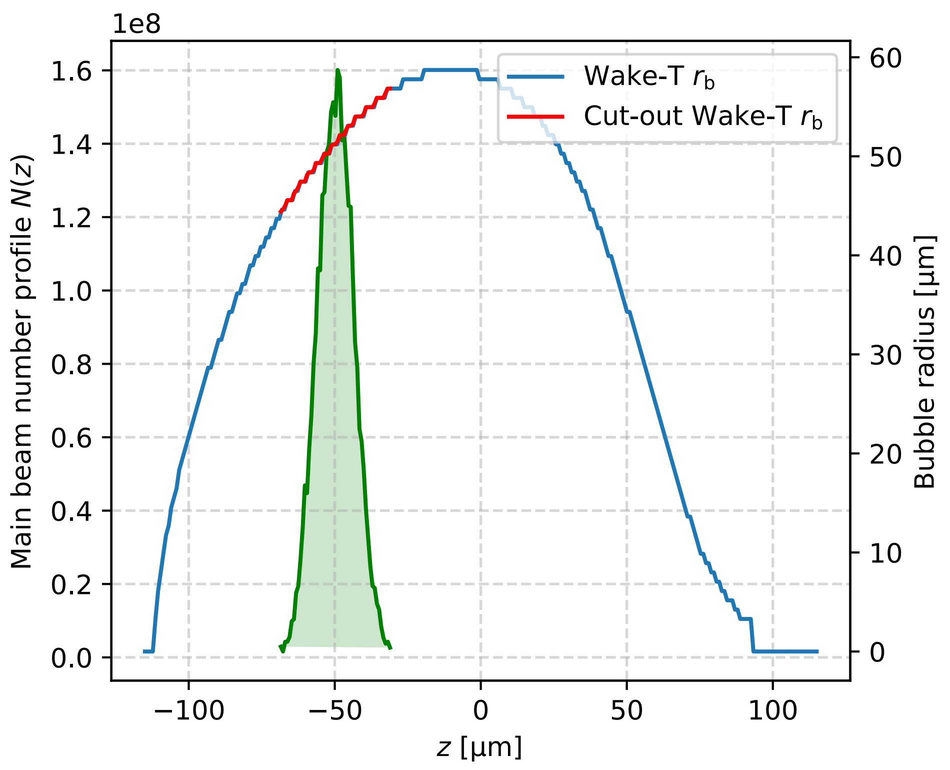
Outline for start-to-end simulation processes using simplified transverse instability model. Wake-T is used instead of PIC here.

- Transverse intra-beam wakefield (G. Stupakov):

$$\mathscr{W}_x(\xi, s) = -\frac{2e}{\pi\varepsilon_0}\int_{\xi_H}^{\xi}\frac{\xi' - \xi}{(r_b(\xi') + \alpha k_p^{-1})^4}\Theta(\xi' - \xi)\lambda(\xi', s)x(\xi', s)\,d\xi'$$

- Combine with Deng et al. equations for radiation reaction:
  - $$\frac{1}{c}\frac{du_x}{dt} \approx -\frac{1}{2}k_p^2 x - \frac{e}{m_e c^2}\mathscr{W}_x - \frac{1}{2}k_p^2 c\tau_R u_x\left(1 + \frac{1}{2}k_p^2\gamma(x^2 + y^2)\right)$$
  - $$\frac{1}{c}\frac{du_z}{dt} \approx k_p\frac{E_z}{E_0} - \frac{1}{4}k_p^4 c\tau_R\gamma^2(x^2 + y^2)$$
  - $$\frac{dx}{dt} \approx \frac{u_x}{u_z}c$$





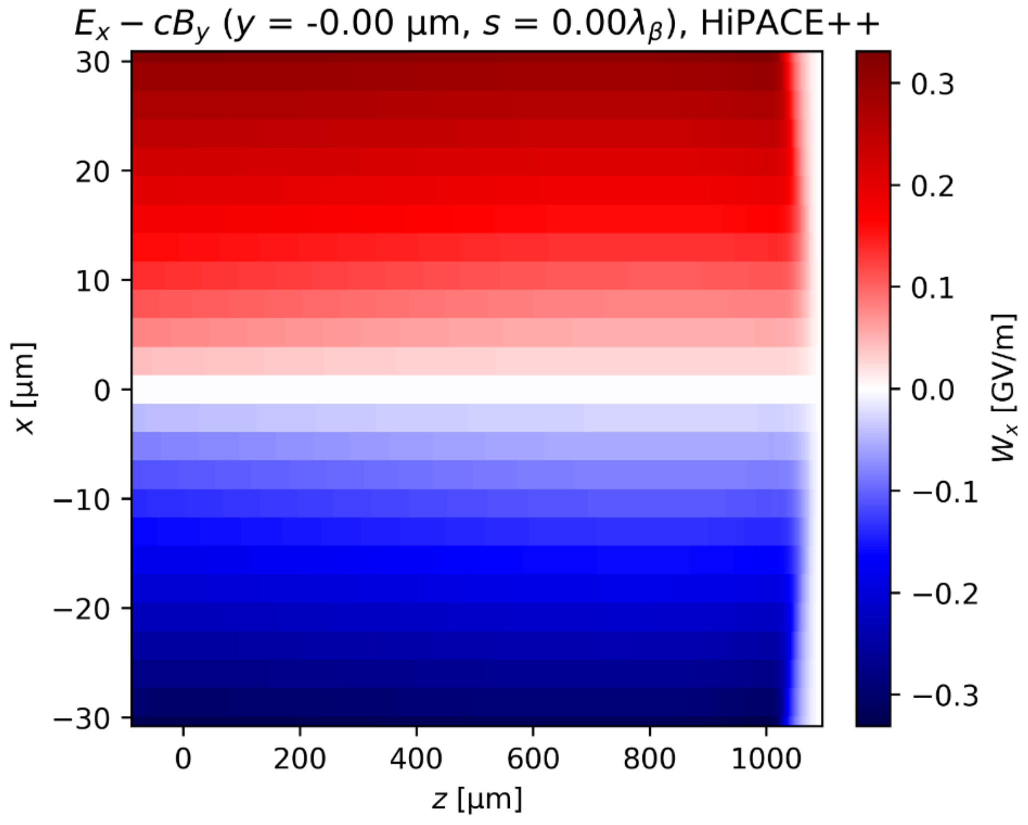
Need initial $E_z(\xi)$ and $r_b(\xi)$ as inputs from e.g. a PIC code (Wake-T used here).

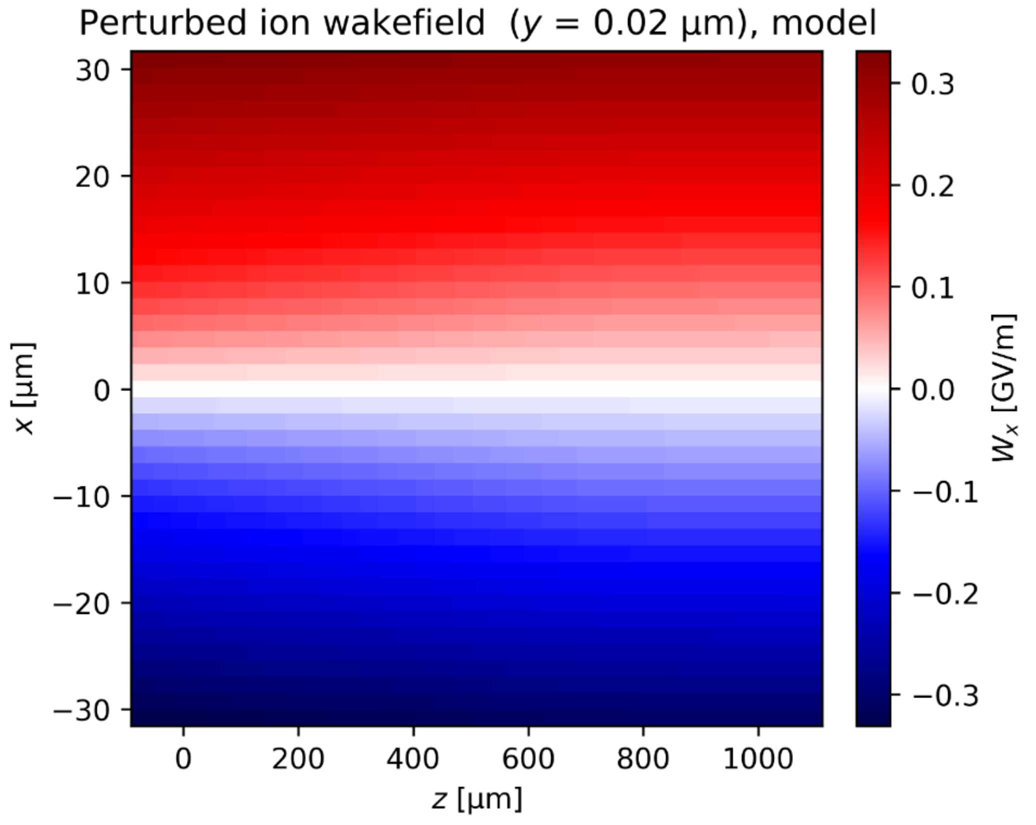
s: beam location.
ξ': long. coordinate of driving particle.
ξ: long. coordinate of reference particle.
ξ_H: long. coordinate of beam head.
α: numerical factor ~1.
k_p^{-1}: plasma skin depth.
$\Theta(\xi)$: Heaviside step function.
$\lambda(\xi, s)$: long. beam number density.
$x(\xi, s)$: particle transverse offset.
$\mathbf{u}(\xi, s) = \mathbf{p}/m_e c$: normalised e^- momentum.
$\tau_R = 2r_e/3c$
$E_0 = m_e c\omega_p/e$: wavebreaking field.

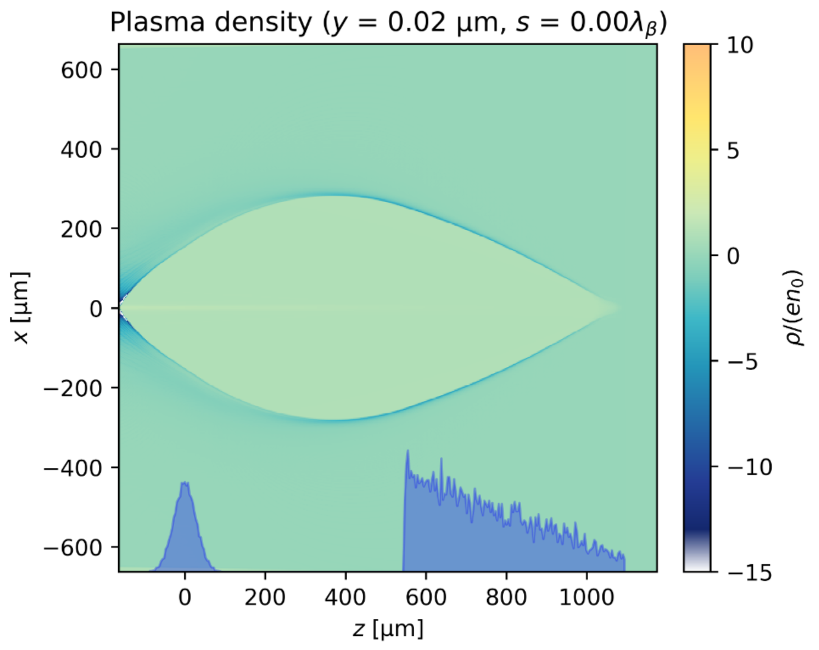
Benchmarks against HiPACE++

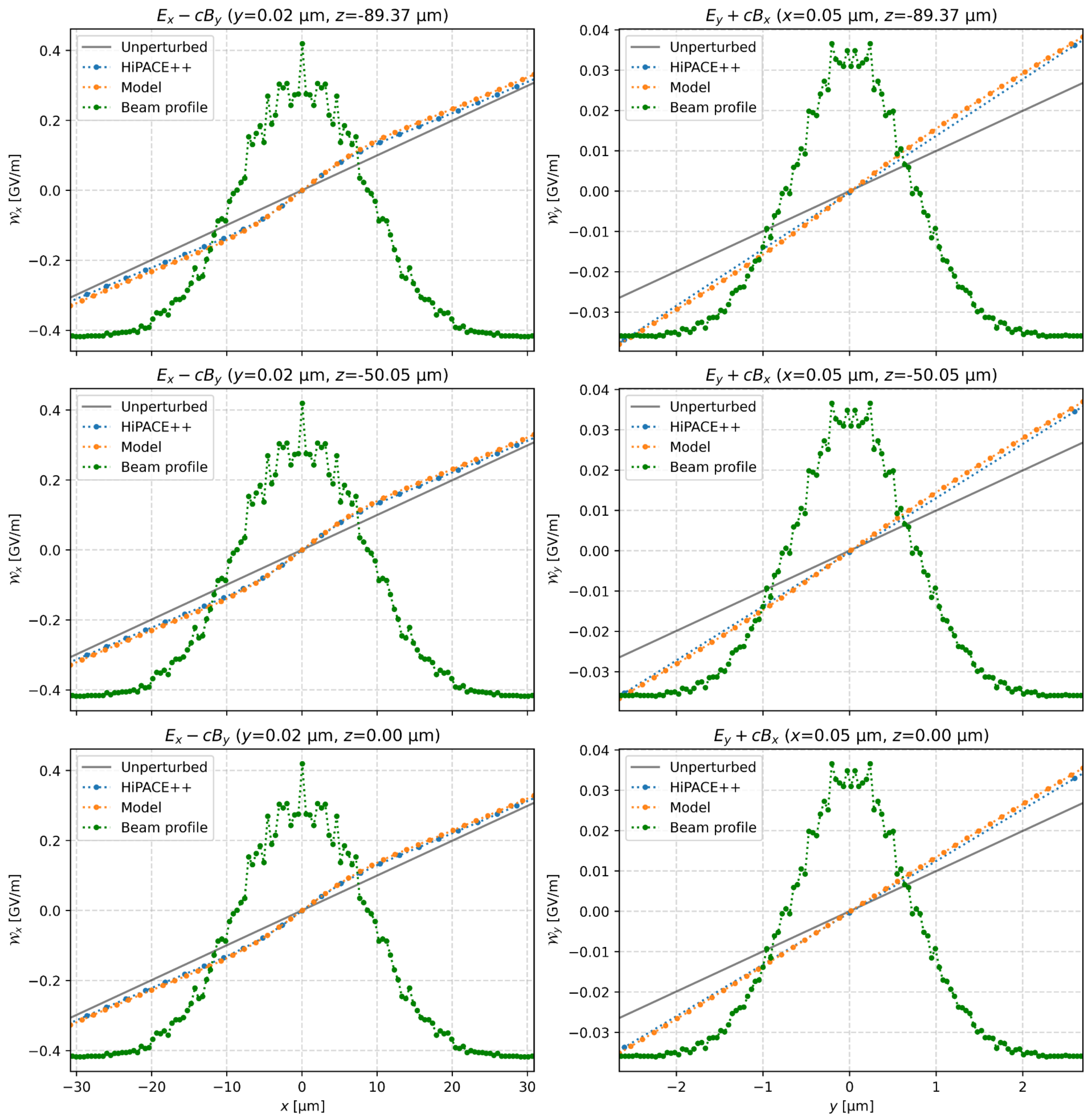
Benchmarks of wakefields

- [Benedetti et al.](#)
- A beam with transverse E-fields $\mathbf{E}_\perp(\mathbf{r},\zeta)$ perturbs the background focusing fields $k_\mathrm{p}\mathbf{r}/2$ so that (moderate non-relativistic ion motion)
- $$\frac{\mathscr{W}_\perp(\mathbf{r},\zeta)}{E_0} = \frac{k_\mathrm{p}}{2}\mathbf{r} + Z_\mathrm{i}\frac{m_\mathrm{e}}{M_\mathrm{i}}k_\mathrm{p}^2\int_\zeta^0(\zeta-\zeta')\frac{\mathbf{E}_\perp(\mathbf{r},\zeta')}{E_0}\mathrm{d}\zeta' = \frac{k_\mathrm{p}}{2}\mathbf{r} + \delta\mathscr{W}_\perp.$$
- I.e. integrate $E_{x,y}$ from head of drive beam to tail of main beam and modify the transverse eq.o.m. with a term $\sim\delta\mathscr{W}_{x,y}$.









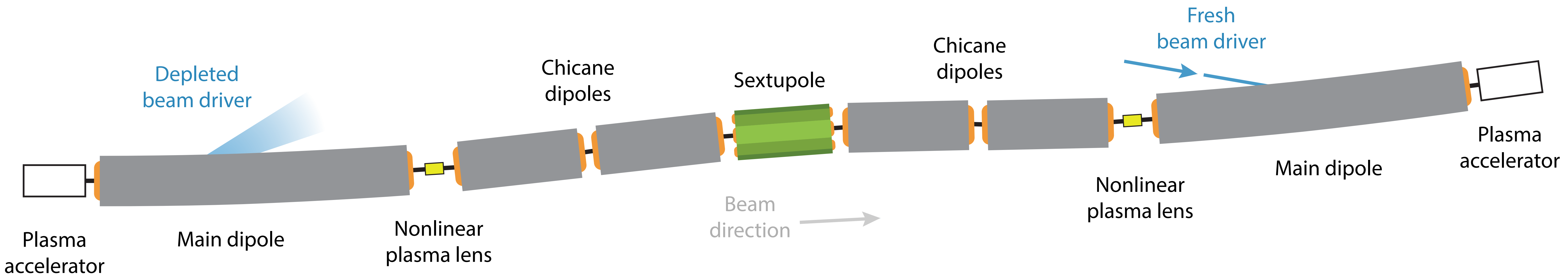
Cost model

E. Adli et al. *"HALHF: a hybrid, asymmetric, linear Higgs factory using plasma- and RF-based acceleration. Backup Document"*, arXiv:2503.23489

- Developed a cost model, accounting for the cost of all collider subsystems—scaled per length (and/or power) based on ILC/CLIC costs
- Defining a reasonable optimisation metric is non-trivial:

$$
\begin{aligned}
\text{Full Programme Cost} = \ & \text{Construction Cost (components and civil engineering)} \\
& + \text{Overheads (design, development, management, inspection, etc.)} \\
& + \text{Integrated Energy Cost (until integrated luminosity reached)} \\
& + \text{Maintenance Cost (over programme duration)} \\
& + \text{Carbon Shadow Cost (construction and operations emissions)}
\end{aligned}
$$

- Used Bayesian optimisation to find minimum cost—fewer than 100 iterations typically

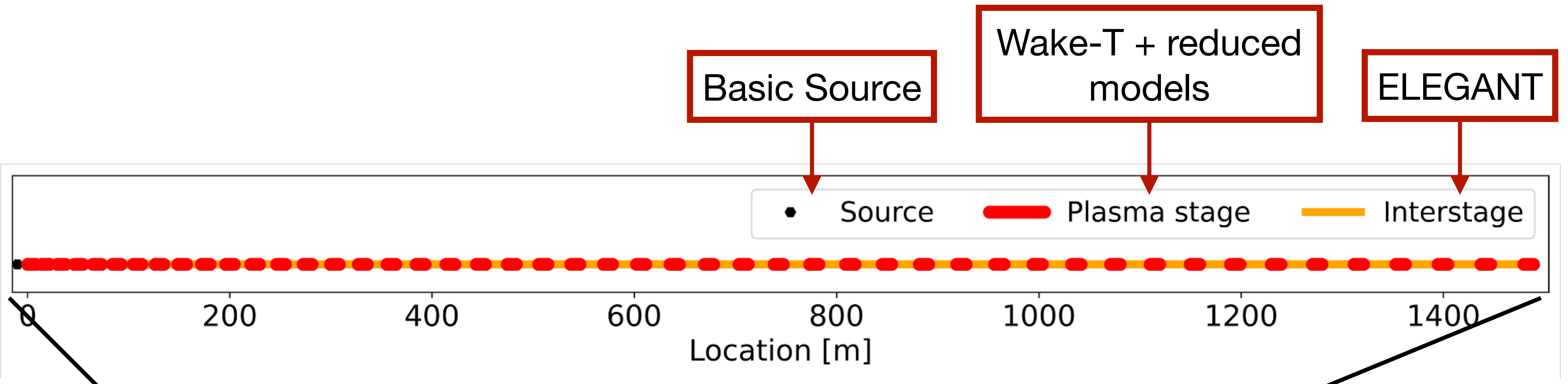

Depleted
beam driver
Chicane
dipoles
Sextupole
Chicane
dipoles
Fresh
beam driver
Plasma
accelerator
Main dipole
Nonlinear
plasma lens
Beam
direction
Nonlinear
plasma lens
Main dipole
Plasma
accelerator

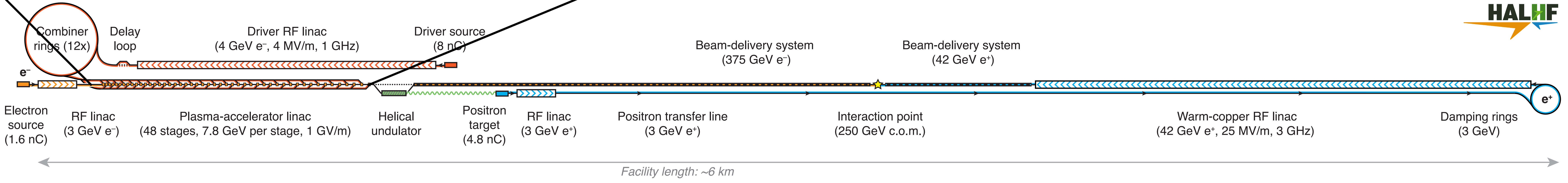
Diagnostics

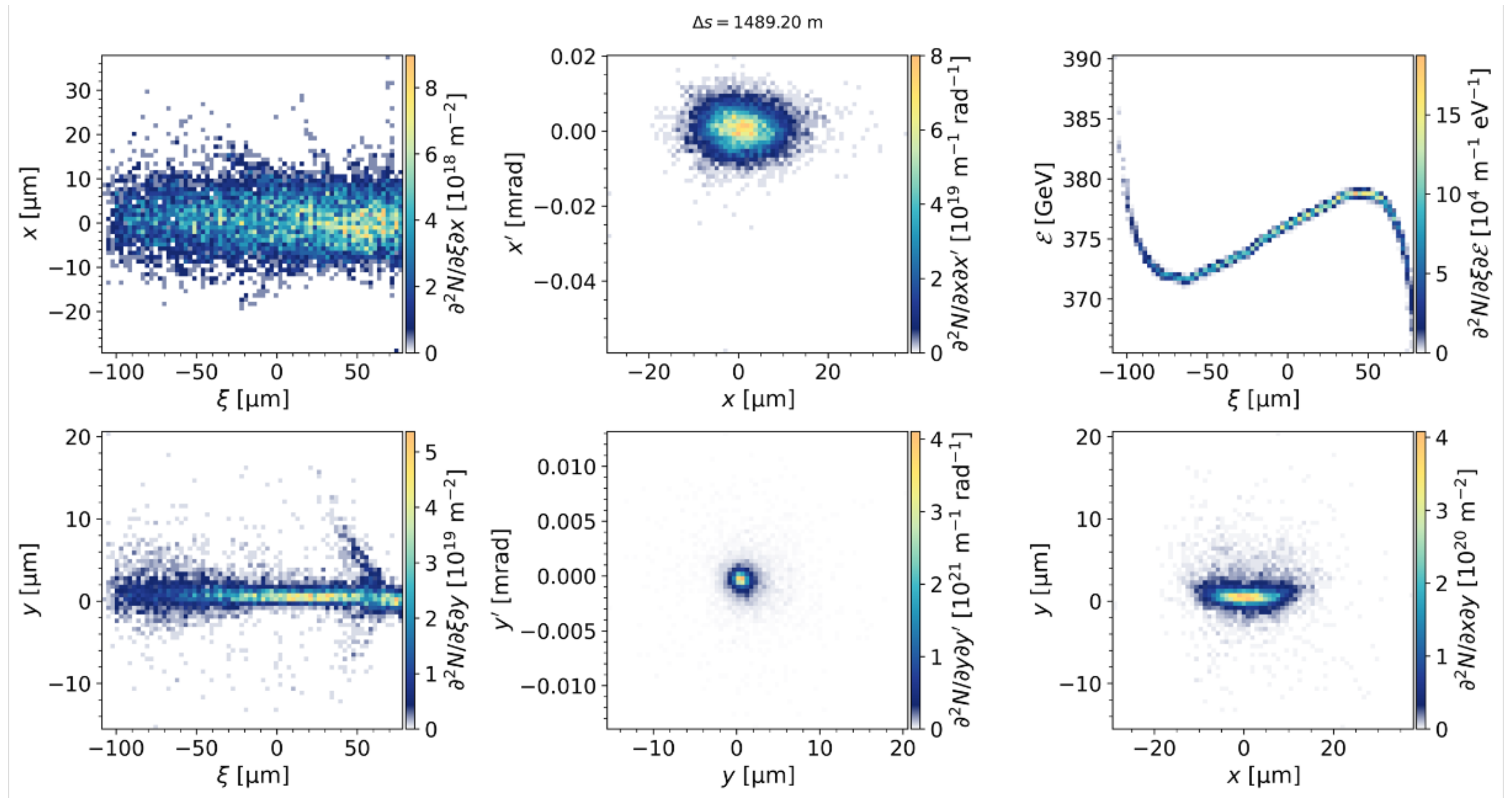
One shot, single stage diagnostic

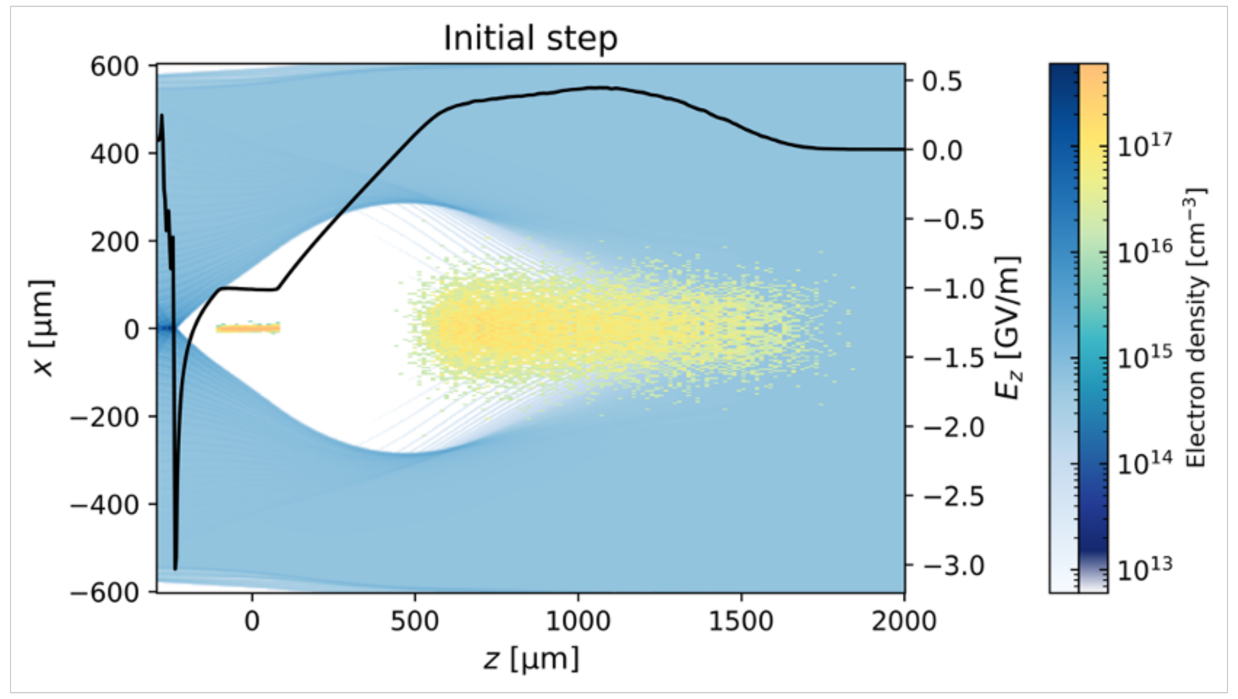
```python
############ Define the stages ############
stage = StageReducedModels()
stage.time_step_mod = 0.03                                # In units of betatron wavelengths/c.
stage.nom_energy_gain = 7.8e9                             # [eV]
stage.length_flattop = 7.8                                # [m]
stage.plasma_density = 6.0e+20                            # [m^-3]
stage.driver_source = driver
stage.ramp_beta_mag = 5.0
stage.enable_tr_instability = True
stage.enable_radiation_reaction = True
stage.enable_ion_motion = True

############ Define interstages ############
interstage = InterstageElegant()
interstage.beta0 = lambda energy: stage.matched_beta_function(energy)
interstage.dipole_length = lambda energy: 1 * np.sqrt(energy/10e9)      # [m(eV)], energy-dependent length
interstage.dipole_field = lambda energy: np.min([0.52, 40e9/energy])    # [T]

############ Define linac ############
linac = PlasmaLinac(source=main, stage=stage, interstage=interstage, num_stages=num_stages)
```

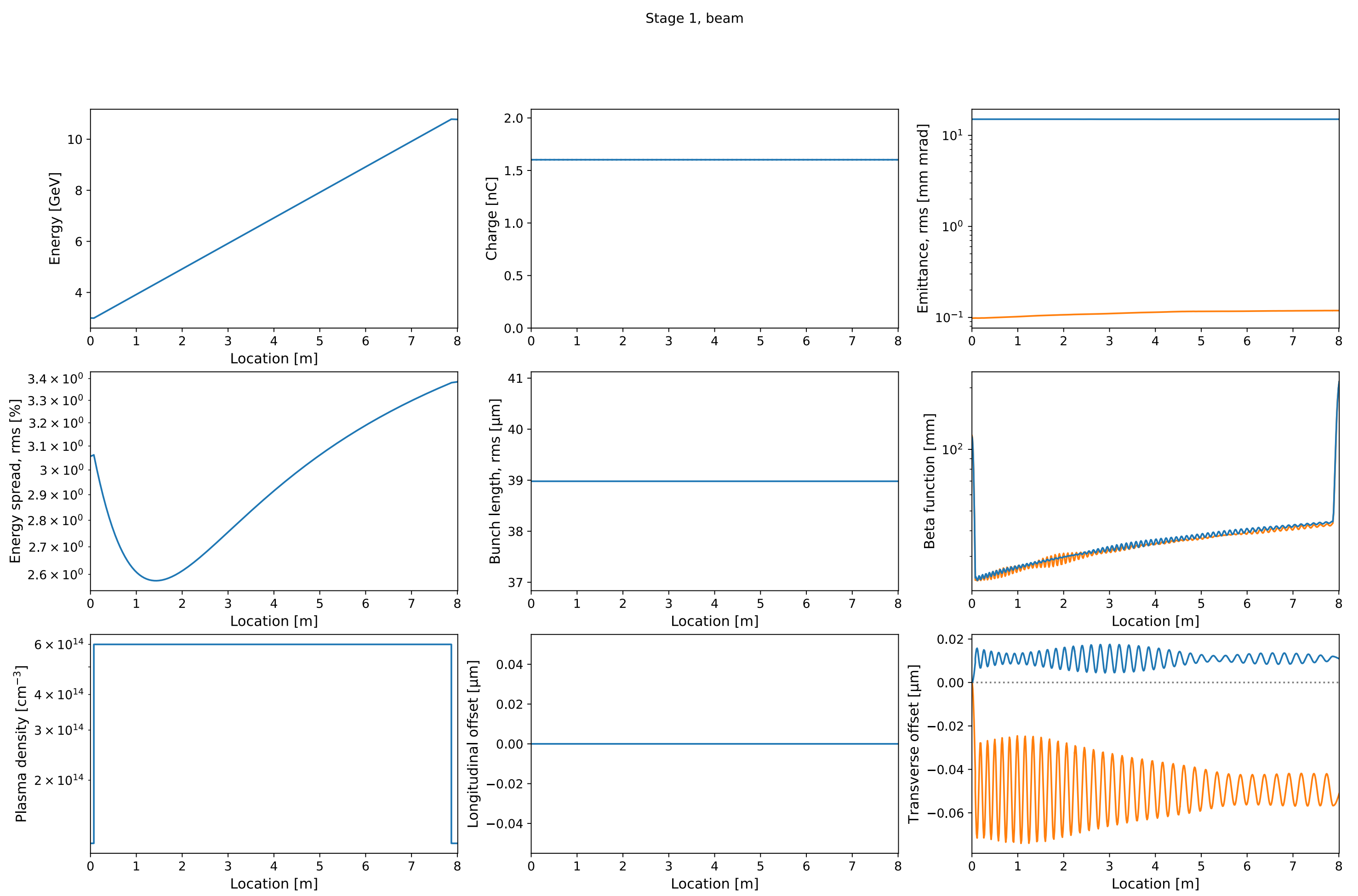
Use case: HALHF plasma linac with simplified models





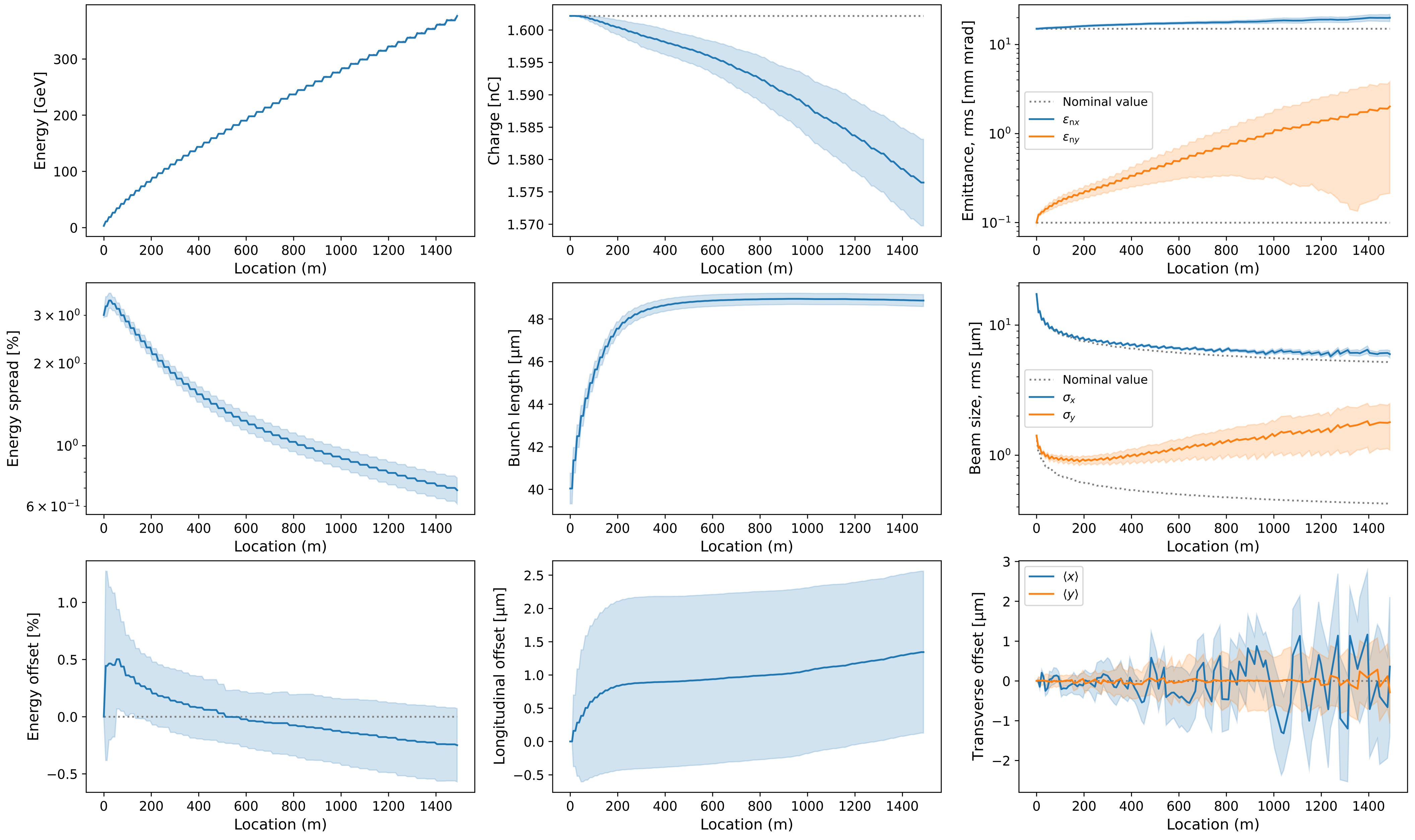






Diagnostics

Multi-shot



Multi-shot, multi-step scan