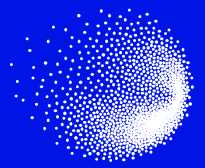


PSI Center for Accelerator Science and Engineering

Challenges in Numerical FEL Simulations

Sven Reiche
DESY, Nov 2025

FEL Simulations – A Black Box Problem

Users are expecting that the code is producing correct results.

They are applying the code to more complex problems of advanced FEL operation mode.

It is hard to judge for the users if the output of the code is physically reasonable.

Even with source code available, codes are too complex to understand the functionality and the limitations

Outline

Background Info

- Modeling of FEL Process

Challenges

- Harmonic Generation
- Partial Coherence
- Coherence at high photon energies
- Red shift for off-axis emission
- Space Charge

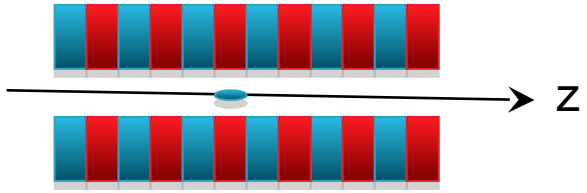
Technical Choices

- CPU vs GPU
- HPC Protocols

Summary

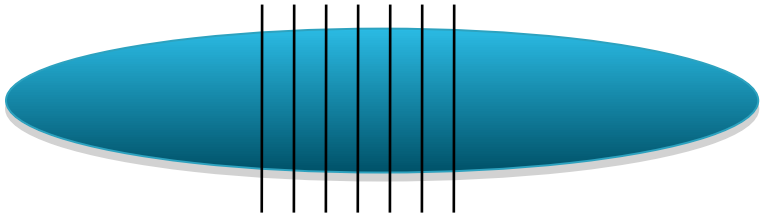
The Modeling Frame

Undulator (~100 m)

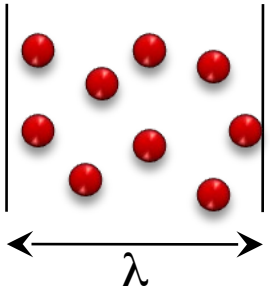


Longitudinal position is independent variable. Undulator field and focusing become “time-dependent”

Electron Beam (~20 microns)



Electron Slice (~ 0.1 nm)



Calculate particle motion in co-moving frame:

$$s = z - \beta_0 t$$

Slice width defines reference wavelength, which is not necessarily the resonant wavelength. Though both should be close to avoid strong drifts in slice

$$\beta_0 = \frac{k}{k + k_u}$$

The FEL Equations (as implemented in Genesis 1.3)

PSI

Long. Position

$$\frac{d}{dz}\theta = (k + k_u)\beta_z - k$$

Energy

$$\frac{d}{dz}\gamma = -k\frac{f_c K}{\gamma}\boxed{\left(ue^{i\theta} + c.c.\right)} + \frac{e}{imc^2}\boxed{\left(Ee^{i\theta} - c.c.\right)}$$

Interaction with radiation field

Interaction with space charge field

Trans. Position

$$\frac{d}{dz}r_\perp = x'_\perp$$

Trans. Divergence

$$\frac{d}{dz}x'_\perp = f(z)x_\perp + \Delta x'(z)$$

Particle tracking in 6 dimensions

e.g. Runge-Kutta

Radiation Field

$$\left[\nabla_\perp^2 + 2ik\left(\frac{\partial}{\partial z} + \frac{\partial}{c\partial t}\right)\right]\boxed{u} = i\frac{e^2\mu_0}{m}\boxed{\sum_j \frac{f_c K}{\gamma_j}e^{-i\theta_j}}$$

(almost) bunching factor

Space Charge Field

$$\left[\nabla_\perp^2 + \frac{k^2(1 + K^2/2)}{\gamma^2}\right]\boxed{E} = i\frac{e}{\epsilon_0 m}2\frac{k(1 + K^2/2)}{\gamma^2}\boxed{\sum_j e^{-i\theta_j}}$$

bunching factor

3D/2D Grid Solver

e.g. ADI

nteraction with

Possible Approximation and Assumption

Resonant Behavior

- Only a narrow bandwidth around a resonant frequency matters. The field can be expanded around a "carrier wave" $E \cdot exp(ikz - i\omega t)$

➡ **Slowly-varying amplitude Approximation (SVEA)**

$$\frac{\partial}{\partial z} E \ll kE$$

Directional Radiation Field

- Main carrier wave propagates in forward direction. Transverse mode size is large compared to radiation wavenumber

➡ **Paraxial Field Equation**

Quasi-Periodic Boundaries

- Due to the resonance condition the field amplitudes cannot differ that much. The time derivative in field equation is negligible

$$\frac{\partial}{c\partial t} E \ll \frac{\partial}{\partial z} E$$

➡ **Steady-State Model**

Collective Variables (not in Genesis)

- Particle distribution can be replaced by bunching factor and energy modulation

$$B = \langle e^{i\theta} \rangle$$
$$\Gamma = \langle \gamma e^{i\theta} \rangle$$

➡ **Linear Model**

Time-Dependent Simulation: Memory Management

Solving the time-dependent field equation is split in two-steps:

1. Solve steady-state solution with source term
2. Solve homogeneous without diffraction

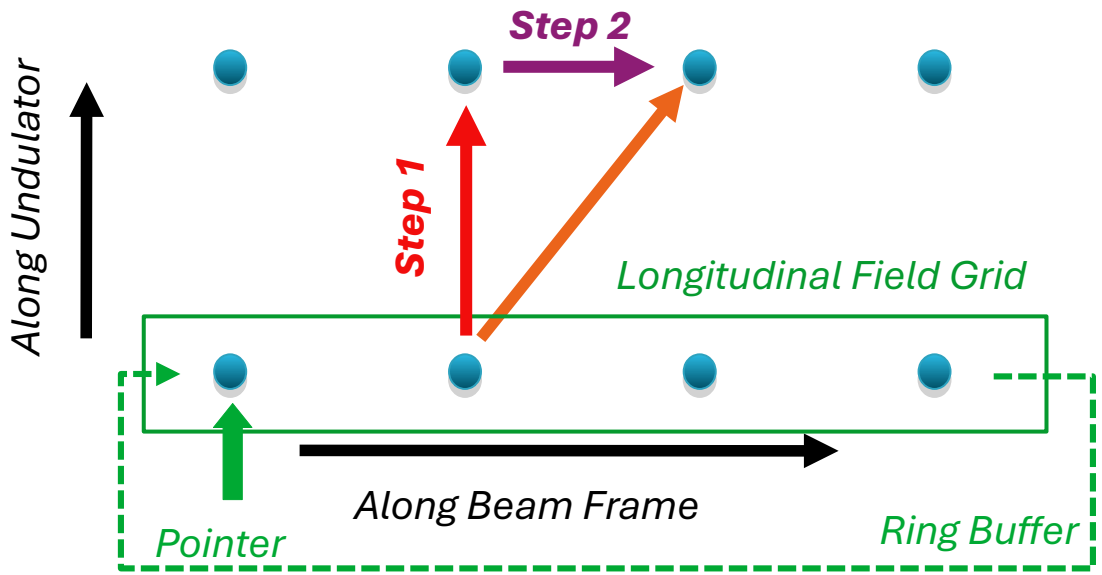
$$\left[\nabla_\perp^2 + 2ik\frac{\partial}{\partial z}\right] E = s$$

$$\left[\frac{\partial}{\partial z} + \frac{\partial}{c\partial t}\right] E = 0$$

Numerical Advantages:

- Step 1 can be vectorized when iterating through grid points
- Step 2 is just a shift in of start point in ring buffer backwards

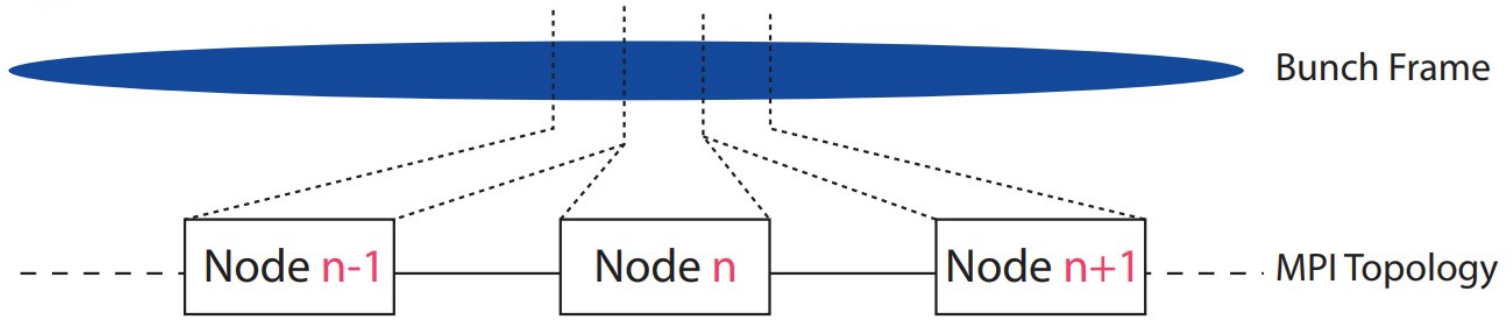
Representation in Memory:



Time-Dependent Simulation: Memory Management in Genesis

Genesis utilize the Message Passing Interface (MPI) to distribute the calculation onto multiple computers. It follows a 1D topology with minimized information exchange among processes.

Memory Management



Entire bunch and field is kept in memory.

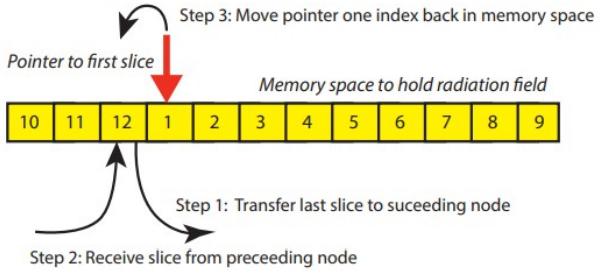
Each node holds multiple adjacent slices.

Resolving each electron (if needed)

200 pC	56 GByte
1 Å (50 fs, 151^2 grid)	51 GBye
Total	**107 GByte**

Field slippage:

Inter-Node (MPI):
one wavefront per node
Intra-Node (C++-Vector):
Increment of Index pointer



Sample Rate and Harmonics

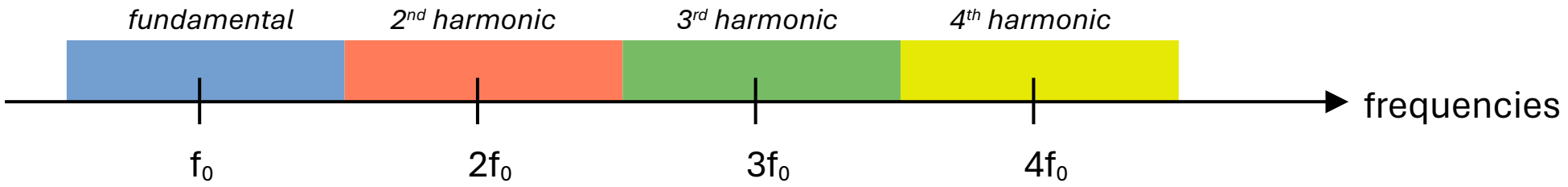
Since a “carrier-wave” is separated from the modeling, the radiation amplitude can only be sampled once per wavelength.

Sample rates lower than once per wavelength are possible but narrows the spectral bandwidth.

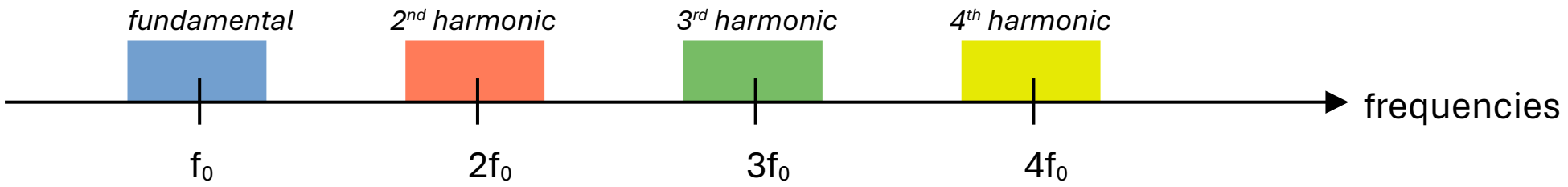
The frequency range of the simulations is between 50% and 150% at maximum

Harmonics must be modeled with their own “carrier-wave” on separated grids.

Sample: Every Wavelength



Sample: Every Two Wavelengths



Sample Rate and Harmonics – Numerical Challenge

- The integration step is coupled to the sample wavelength and should never be larger (but can be smaller).
- The allows to relate both values to the expected gain length, e.g. 10 steps per gain length to speed up hard X-ray simulations.
- This fails if the bunch is chirped, jumps between sub-harmonics for harmonic lasing, two color etc.
- The alternative approach is to have a single grid with a sample rate of multiple samples per wavelength with a non SVEA solver (e.g. **Puffin**)
- However it increases execution time and memory demand significantly

Exampled: Hard X-ray with 3rd harmonics

	SVEA	non-SVEA	Scaling
Sample	$5 \cdot \lambda$	$0.2 \cdot \lambda$	25
Grids	2	1	0.5
Step	$5 \cdot \lambda_u$	$0.2 \cdot \lambda_u$	25

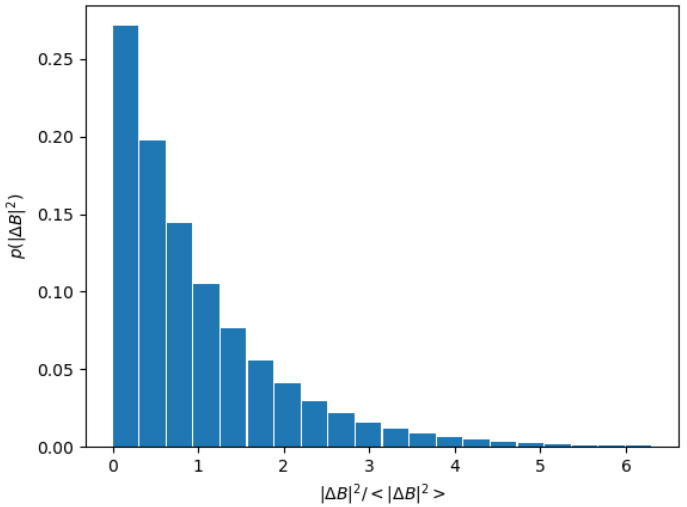
Total Penalty of Execution: **312.5**
Total Penalty in Memory: **12.5**

Shot Noise Algorithm

For faster execution it is highly desirable to run with less particles than electrons.

However, it must produce the correct shot noise statistic.

$$\langle |\Delta B|^2 \rangle = \frac{1}{N_e} \qquad p(|\Delta B|^2) = e^{-N_e |\Delta B|^2}$$



It is the same statistic of a random walk of N_e steps in a 2D plane

Numerical Implementation:

- Use bunching as an internal degree of freedom for each macro particle (linear regime only)
- Use beamlets (a group of macro particles with identical variable values but equally distributed in longitudinal space over one wavelength. Add a small sinusoidal modulation in its position to control the bunching value) – most codes are using this approach

The beamlets can never be broken up without destroying the correct bunching statistic

Harmonic Generation Schemes and Beamlets

Harmonic generation requires an upsampling in the electron slices and radiation field grid.

- Beamlets cannot be split. Therefore the longitudinal position is scaled and the distribution is copied to all slices. (Copying is not really needed but can cause bandwidth problem)

Method fails when cooperation length is shorter than subharmonics, since superradiant mechanism are numerically suppressed.

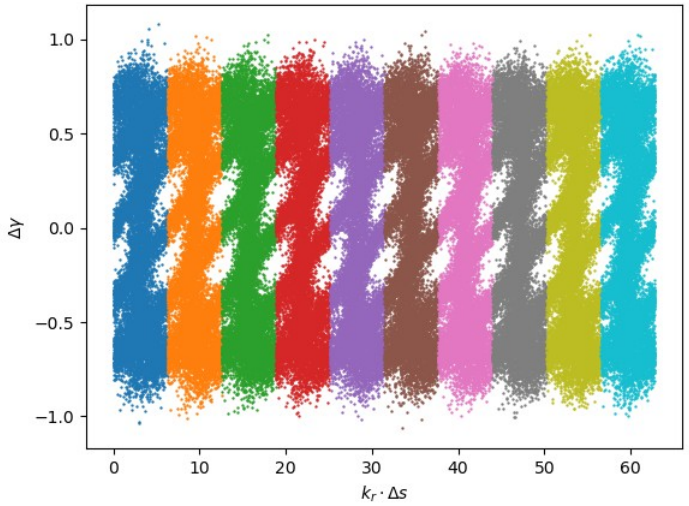
- In one-4-one simulations shotnoise is fulfilled trivially. The distribution can be sliced.

It opens up the box of pandora

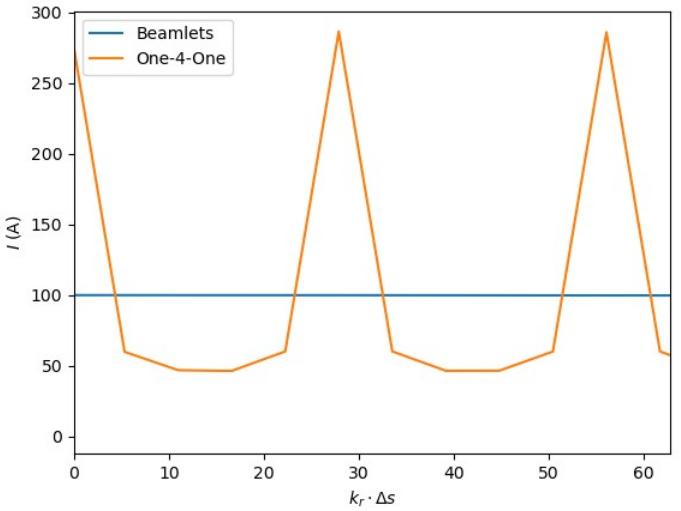
(see next slides)

Example: HGHG for 5th harmonics

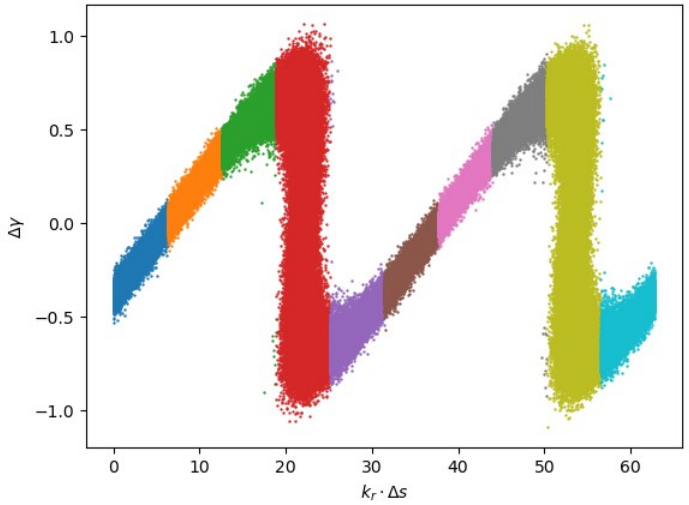
Beamlet Method



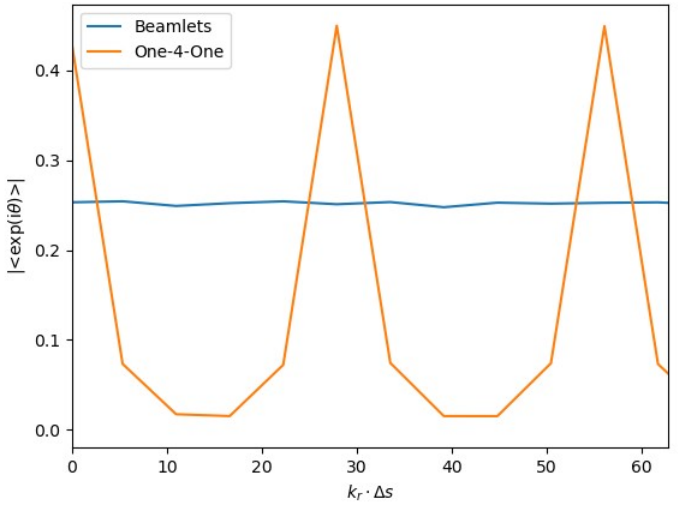
Sampled Current



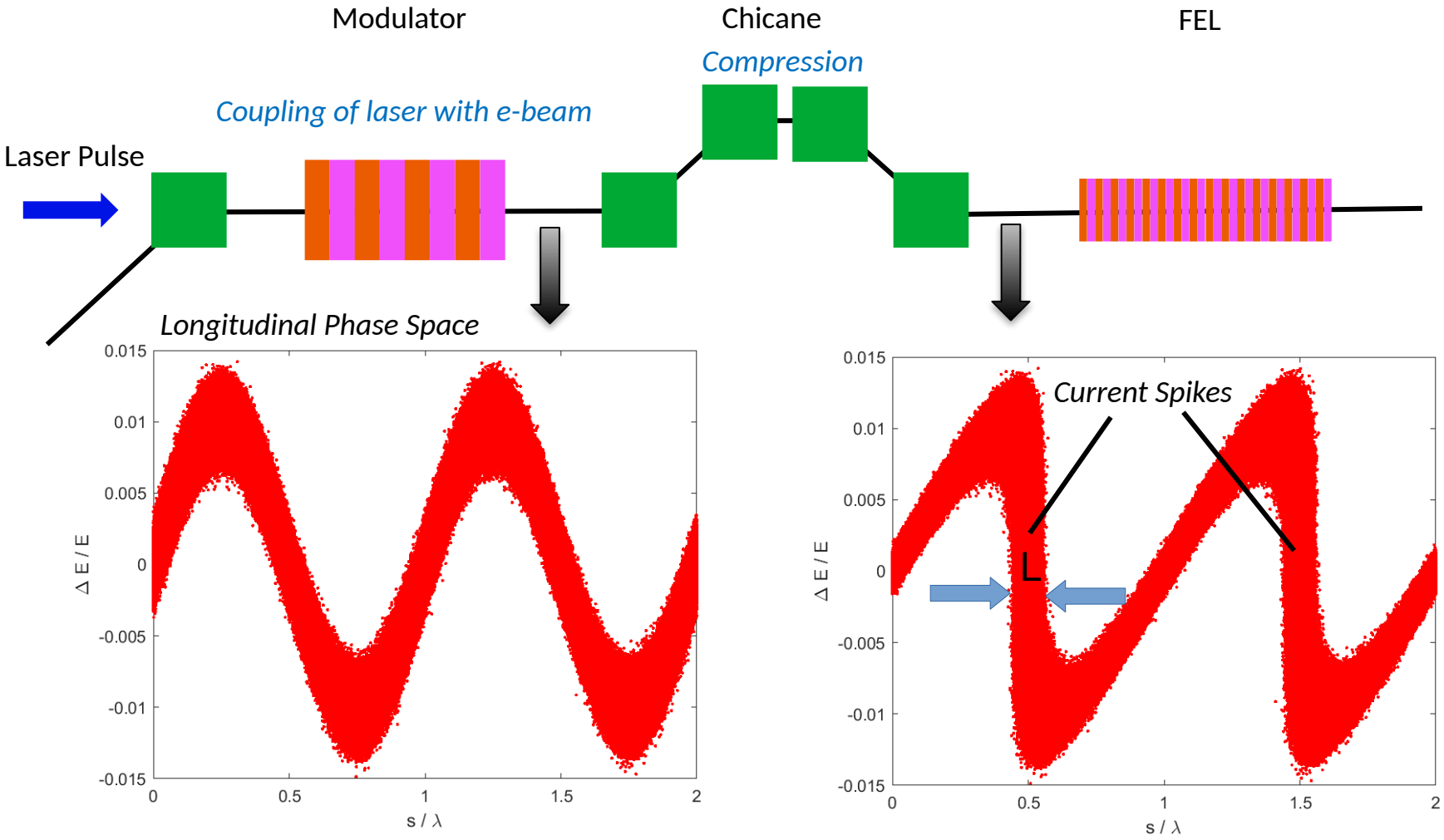
One-4-One Method

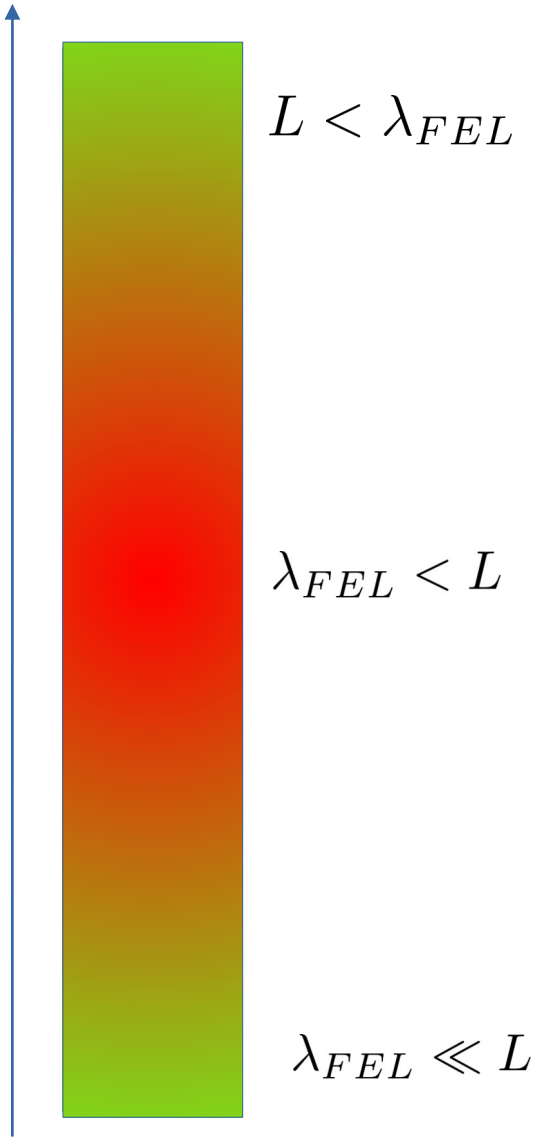


Sampled Bunching



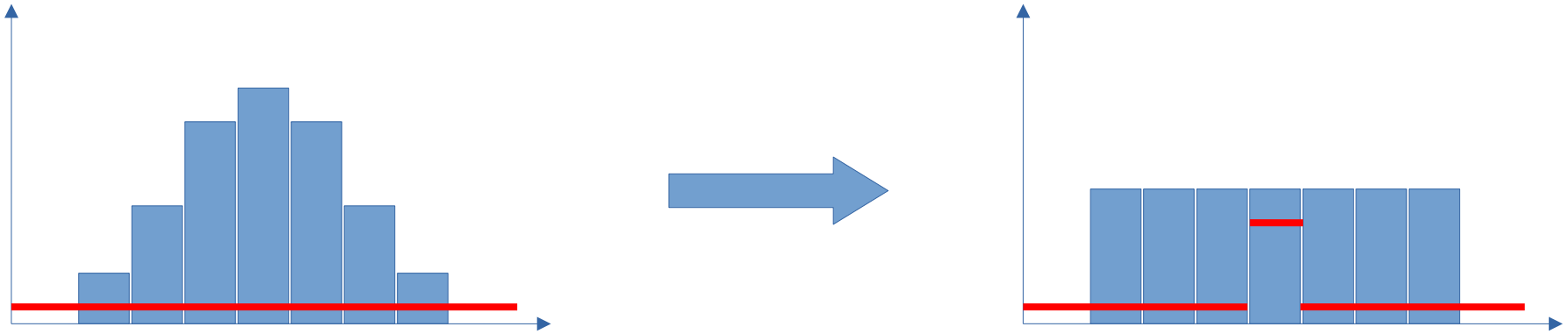
Between ESASE and HGHG - Physics





HGHG and ESASE share the same set-up. The transition (partial coherence) should be smooth

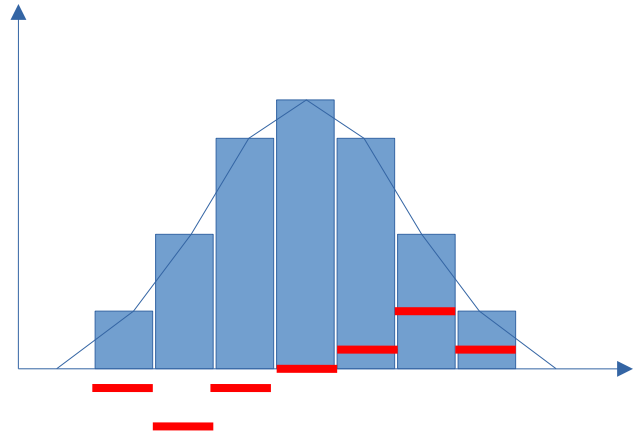
Between ESASE and HGHG - Numerics



To prepare for a smooth transition from ESASE to HGHG allow for uneven distribution per radiation slice, which should be easy to do with.

But the calculation of the bunching is done per grid point, resulting in an artificial bunching, locked to the grid:

$$b_j = \frac{1}{\pi I} \left. \frac{dI}{ds} \right|_{s_j}$$



It looks like a coherent signal, but it is a numerical artifact and not the growing HGHG seed!

Between ESASE and HGHG – Numerics - Solutions

There are three possible solutions:

1. Dual grid: Calculate the source term on two grids. Grid 1 is the regular grid, Grid 2 is shifted by half wavelength. The effective source term is:

$$\tilde{b}_j = \frac{1}{4}\left[2b_{1,j} - b_{2,j} - b_{2,j+1}\right]$$

   Allows to explore shorter current spikes but will eventually break down again.

2. Field Discretization in Frequency: Evaluate the bunching factor of the entire beam.

   Strong penalty in execution time since vectorization of the problem is lost.

3. Use a non period-average solver: Consistent methods since it avoids that the paraxial approximation is broken for short current spikes.

   Unfavorable scaling since multiple steps per wavelength and sub-wavelength grid resolution is needed.

Coherence Degradation - Physics

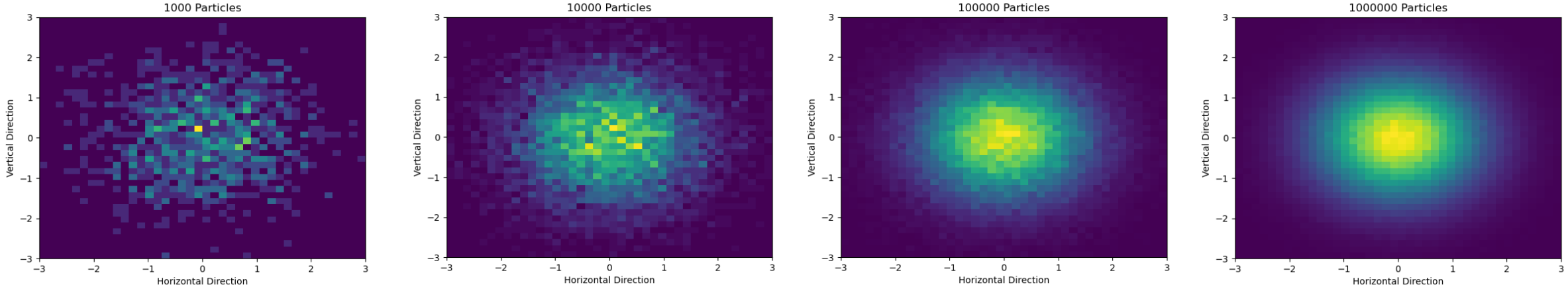
Pushing the photon energy barrier into the several tens keV photon energies with low emittances and peak currents of few kA, the resulting number of electrons per wavelengths drops to

The discreteness in the transverse distribution becomes noisy:

3 kA at 36 keV FEL photon energies ➡ about 2100 electrons

While the fundamental mode bunches all electrons (is insensitive to the transverse position because it averages over 1000 slices), the noise in the distribution allows coherently under a larger angle.

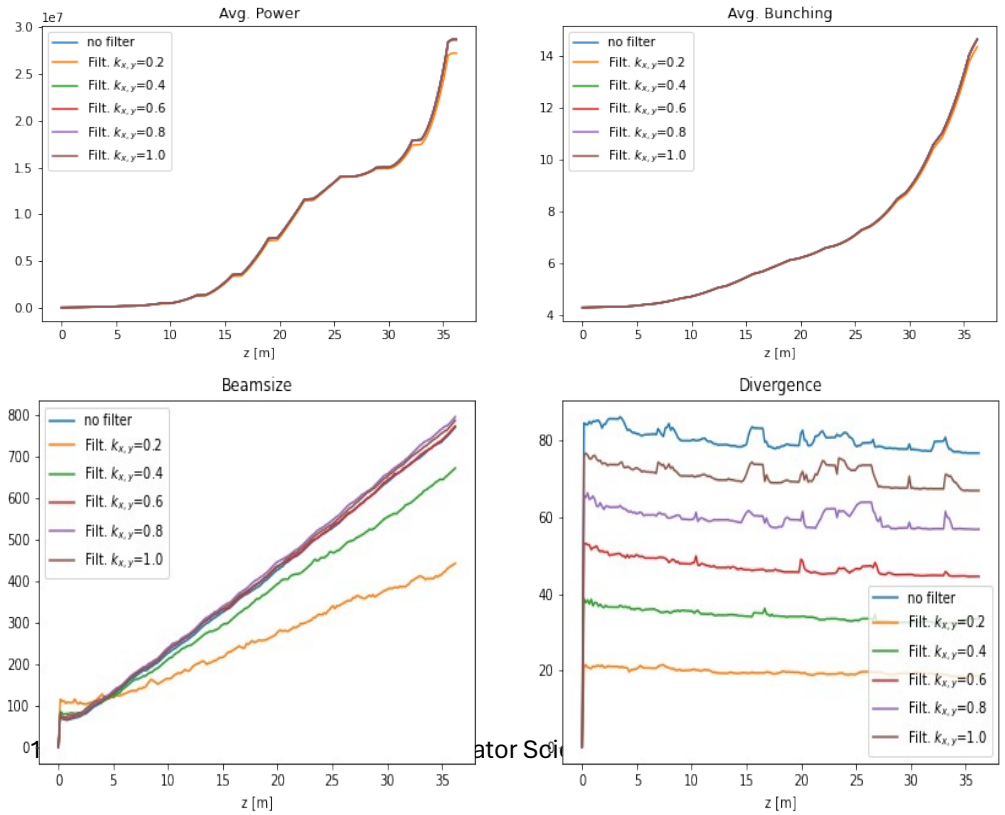
This „extra“ emission has poor coherence, since the emission under an angle can be random even within the coherence length of a SASE spike.



Coherence Degradation - Numerics

For noisy transverse distribution in the source term (“pixel noise” on the grid), divergent modes are reflected by the boundaries forming standing pattern. These couple weakly to the FEL process but obscures the beam size and divergences.

Numerical counter measure is a filter in the 2D Fourier space of the source term in the field equation

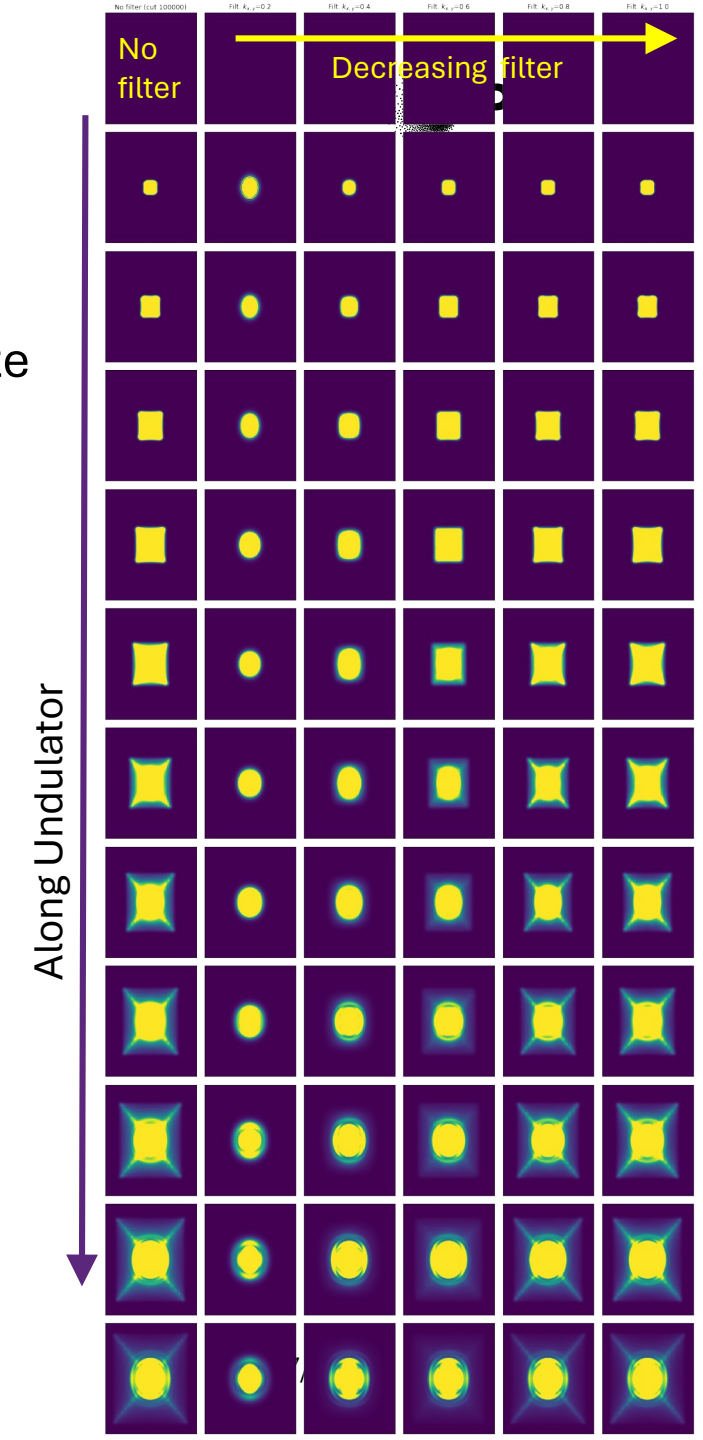


Alternative solution:

- Absorbing boundary condition of the field grid
- Discretization of the field by set of orthonormal 2D function (e.g. Gauss-Hermite) instead finite difference of a grid.

All are more CPU intensive and can cut-out physics.

Simulations done by E. Ferrari



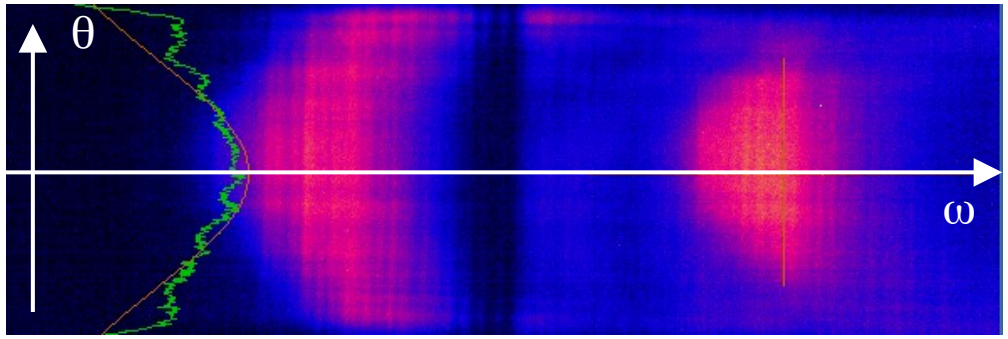
Emission under an Angle - Physics

The emission under an angle has a red shift:

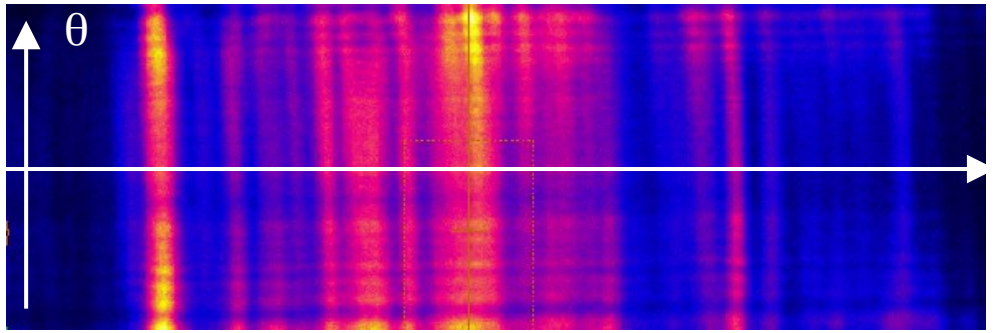
$$\lambda = \frac{\lambda_u}{2\gamma^2}\left(1 + K^2/2 + \theta^2\gamma^2\right)$$

Spontaneous radiation shows this quadratic dependence, while SASE shows a stable line up to the coherence angle.

Spontaneous radiation



SASE radiation



It would also allow that a bunched beam would emit under an angle (hollow mode) if:

- The frequency band of the off-axis spontaneous radiation overlaps with the frequency of the bunch signal
- The angle is still smaller than the coherence angle of the FEL mode

Emission under an Angle - Numerics

Current codes are using the paraxial approximation. While the coherent FEL radiation is correct, the emitted spontaneous radiation is not. However for SASE FEL the forward emitted radiation matters.

To include this effect, the code must calculate the retardation time of the field emission.

This is quite challenging and needs a lot of book keeping (CSR codes)

Another approach would to transfer to the rest frame of the electron beam, as it is done for space charge calculation of photo-electro RF guns.

However, the electron motion is still relativistic (K>1) and the paraxial approximation is lost.

Space Charge and collective effects

Similar to bunching (ESASE problematic), space charge is split into two domains

1. Sub-wavelength scale: Opposing force, counteracting the bunching by the FEL process

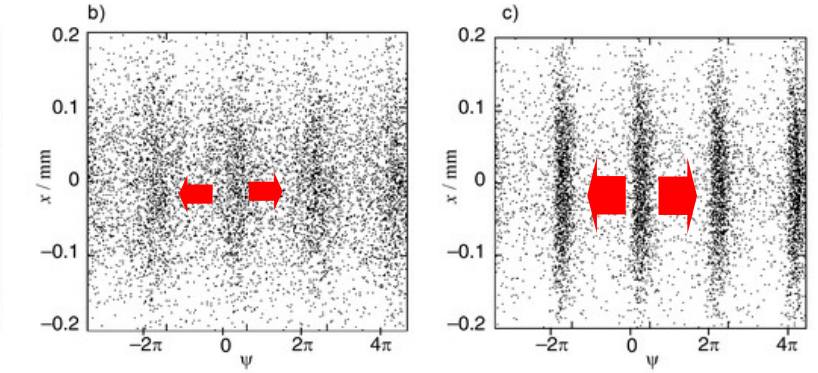
2. Multiple wavelengths scale: Shifting the energy of a slice.

Most common approach is to solve the sub-wavelength self-consistently but calculate the long range with Green's function, similar to wakefields.

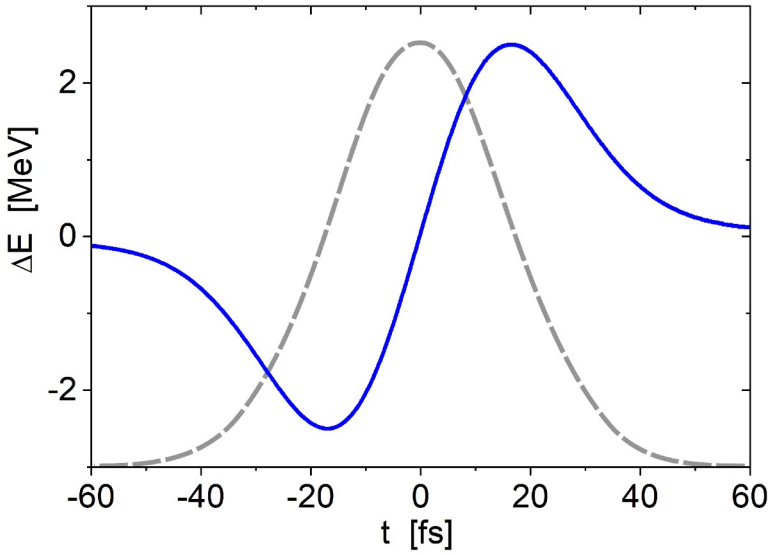
Similar to partial coherence a gradient of the space charge field over one slide causes problems with numerical stability.

Sampling the space charge field on a high resolution grid, but has the same problem of scaling up the memory requirement

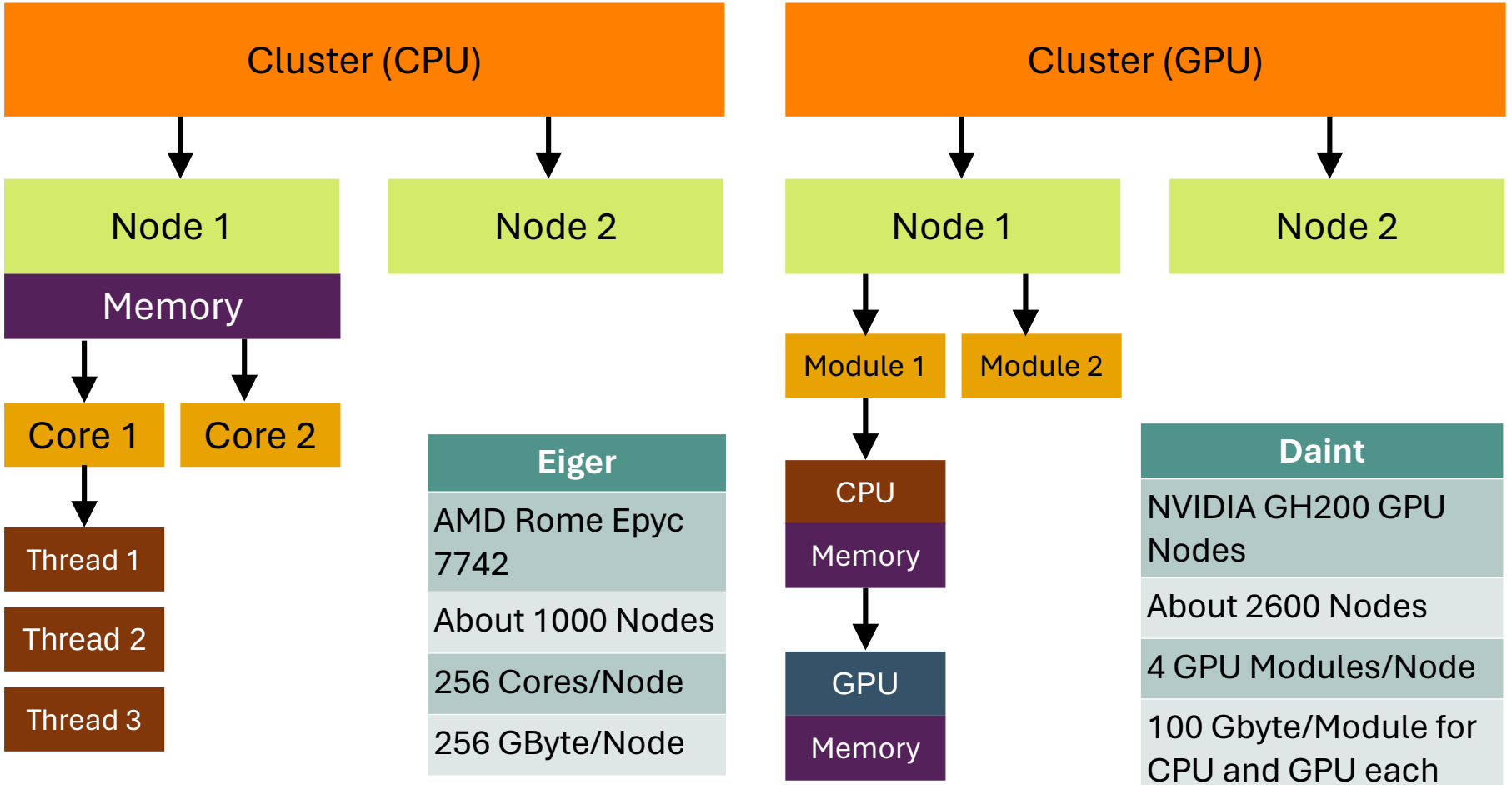
1) Self-consistent sub-wavelength SC



2) External Space-charge Potential



Modern Cluster Topology (Example: Alps @ CSCS)



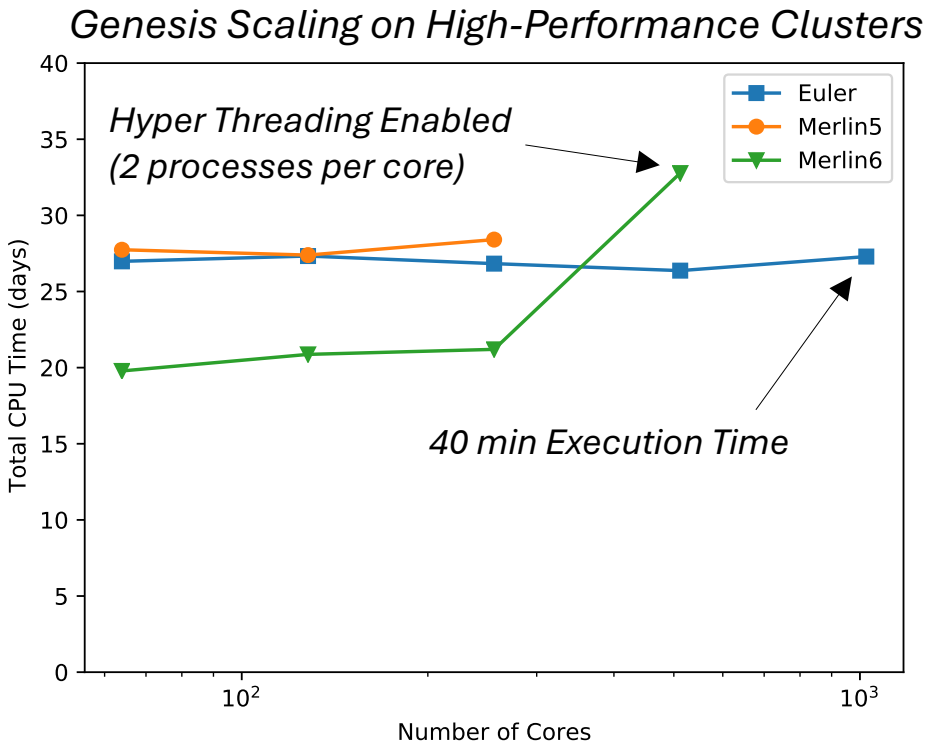
Eiger
AMD Rome Epyc 7742
About 1000 Nodes
256 Cores/Node
256 GByte/Node

Daint
NVIDIA GH200 GPU Nodes
About 2600 Nodes
4 GPU Modules/Node
100 Gbyte/Module for CPU and GPU each

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

High-Performance Cluster – Programming Interface for C++/C

- Historically no computer could hold the entire beam in its memory. Therefore the best choice was **MPI** (Massage Passing Protocol) to interface several computers over the ethernet. It is very robust, scalable and has an abstraction layer to the underlying infrastructure. It is an parallelization between nodes
- With multiple core per chip (=node), shared memory access becomes feasible with **openMP**. It benefits highly from vectorization of programming loops. It is a parallelization within a node.
- Recently the internal memory of GPU reached 100 Gbyte. The processor and the memory access is highly optimized for vectorization of linear transformations, suitable for most particle tracking and field solvers. The programming API is **Cuda**
- Cuda and openMP can be combined with MPI for hybrid codes, but not recommended



While almost all case could be carried out on a single node for all cases, significant speed-up is possible with more nodes, in particular for MPI

Summary

- FEL Codes have the tendency to be treated as a black box. It is difficult to estimate if a results makes sense or is prone to numerical errors.
- A single code cannot cope with all FEL mode proposed and studied.
- There are general utility codes (e.g. Ginger or Genesis), which can do most by compromises at many corners
- There are specialized code to solve a specific problem (e.g. Puffin) but have the penalty for reduced utility or longer run time
- Recent development in HPC capabilities can benefit FEL codes if adapted to the new protocol (in particular GPU driven by AI applications)
- Incentive to rewrite old code for new HPC platforms is low. Better develop a new code or a robust kernel to be called as a library, e.g. core particle and field tracking (in Genesis it is 10% of code, using 90% of the execution time)