Puffin Manual

June 7, 2017

Contents

1 Introduction

Puffin (Parallel Unaveraged Fel INtegrator) simulates a Free Electron Laser (FEL). Puffin is a massively parallel numerical solver for an unaveraged, 3D FEL system of equations, and is written mostly in Fortran 90, using MPI and OpenMP.

The initial publication describing the first version of the code is in [1]. The code has undergone many improvements and extended its functionality since then. It no longer uses an external linear solver package, and the only external package now required is FFTW v3.3.x onwards.

Puffin is a so-called 'unaveraged' FEL code - meaning it is absent of the slowly varying envelope approximation (SVEA) and wiggler period averaging approximations. It does not utilize a 'slicing' model of the beam phase space and radiation field, and instead utilizes an algorithm which is much more similar to a Particle-In-Cell (PIC) code methodology.

In addition, some accelerator components are included for simulation of the 'realistic' undulator line, and together with the lack of restrictions, means it may model:

- The full, broad bandwidth frequency spectrum, limited only by the Niquist frequency of the mesh
- Full electron beam transport
- Transport of large energy spread beams through the undulator, and the radiation emitted from these beams
- Tapered undulators
- Fully 3D undulators, including modelling of the wiggler entries/exits and natural focusing
- Interleaved undulator-chicane lattices

- Variably polarized undulators
- Tuning of each undulator module

It presently does not include the effects of space charge, and ignores emission of the backwards wave from the e-beam.

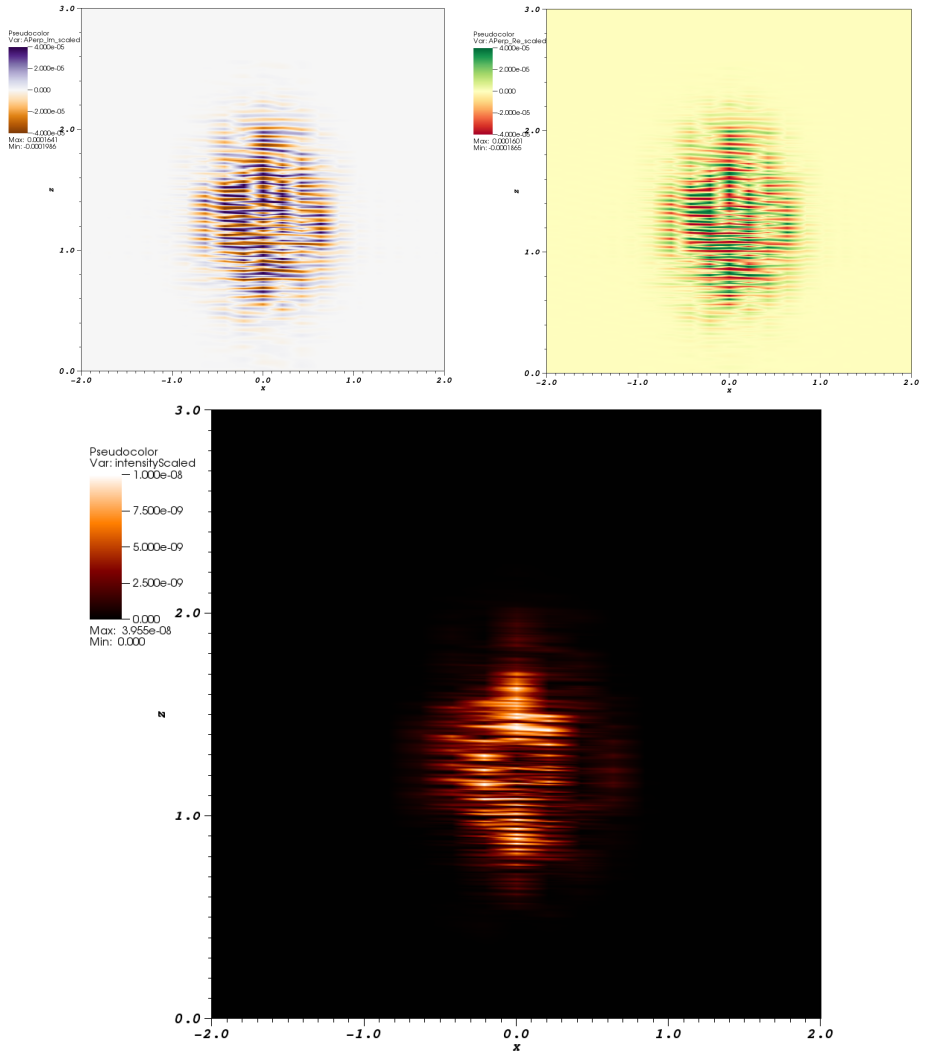

Figure 1: Radiated spontaneous field from Puffin in the first few undulator periods - x (top left) and y (top right) polarized fields, and instantaneous intensity (bottom), at $y = 0$. Radiation is propagating in the negative z direction (the vertical axis). One can see the noisy phase of the radiation in both transverse (x) and temporal (z) directions, which is due to the shot-noise of the electron beam.

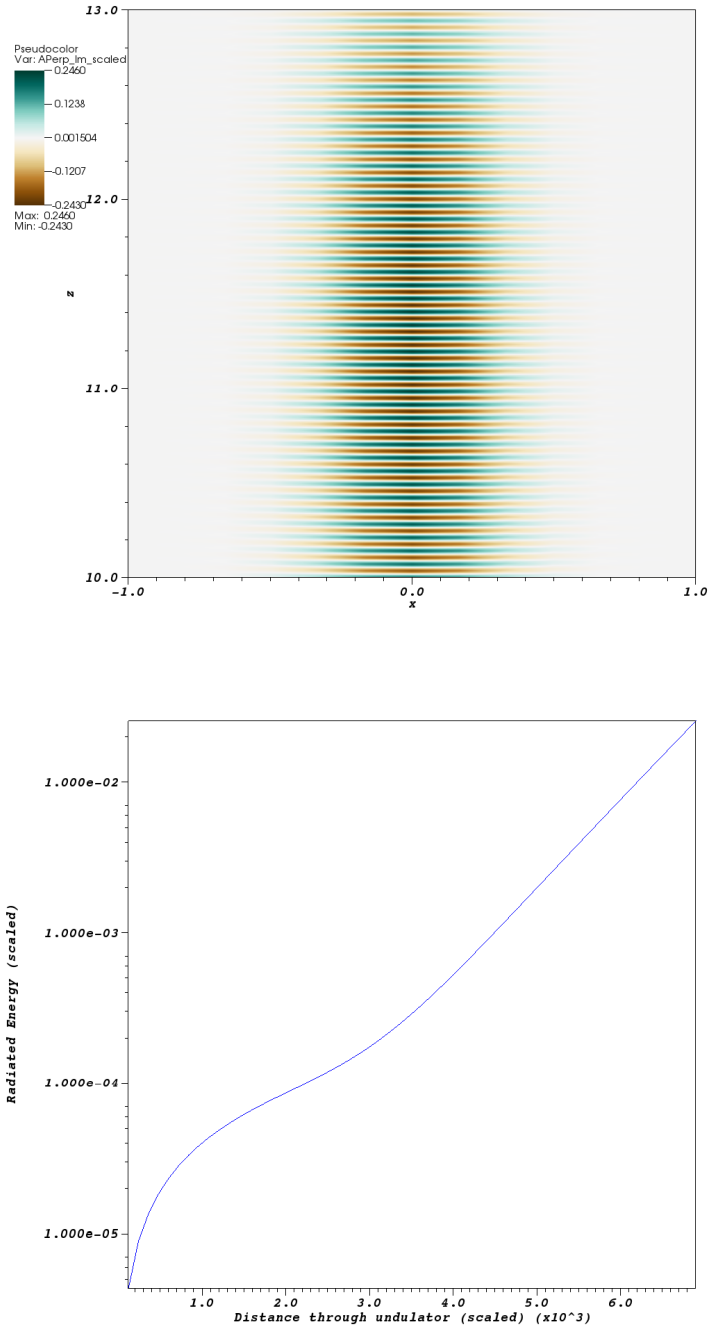

Figure 2: Radiated field from Puffin after amplification in the undulator - x polarized field (top). Radiated energy vs scaled distance through the undulator is on the bottom. Observe the noise from the previous figure has disappeared; the amplification process cause the electrons to align and radiate in phase.

2 Overview

Puffin, being an unaveraged FEL code, is quite distinct from other FEL codes in it's lack of restrictions on the radiation and electron bandwidth being modelled. Due to the nature of the code, and the likely use cases for such a code, we have attempted to create a resource which is a flexible tool for numerical scientific research into some more esoteric FEL physics and regimes, while still containing enough 'realistic' components to enable modelling of a current or future facility.

The main Puffin algorithm solves the FEL system in a scaled reference frame - see section 5 for a description. Data I/O may be in either SI or the Puffin scaled variables.

2.1 Input Files

At a minimum, one must specify 2 files for use in Puffin - a main input file, and a beam input file. The main file sets up the system scaling, the field mesh, the integration step sizes, and other simulation options. A simple, single undulator system can be setup here. It also specifies the names of the other input files, including the beam file.

The beam file specifies the electron beam parameters - there are 3 different types of beam input, detailed later in section 2.4. Multiple beams can be specified, each with different characteristics, such as energies, lengths, charges, current profiles, etc.

Optionally, one may also supply a radiation seed file, to describe an injected radiation seed into the FEL with the electron beam. Similarly to the beam file, many radiation seeds can be specified here, at different frequencies, intensity profiles, *etc.*

In addition, one may also supply a lattice file, which describes the layout of undulator modules, focusing quadrupoles, chicanes, drifts *etc.* If this is

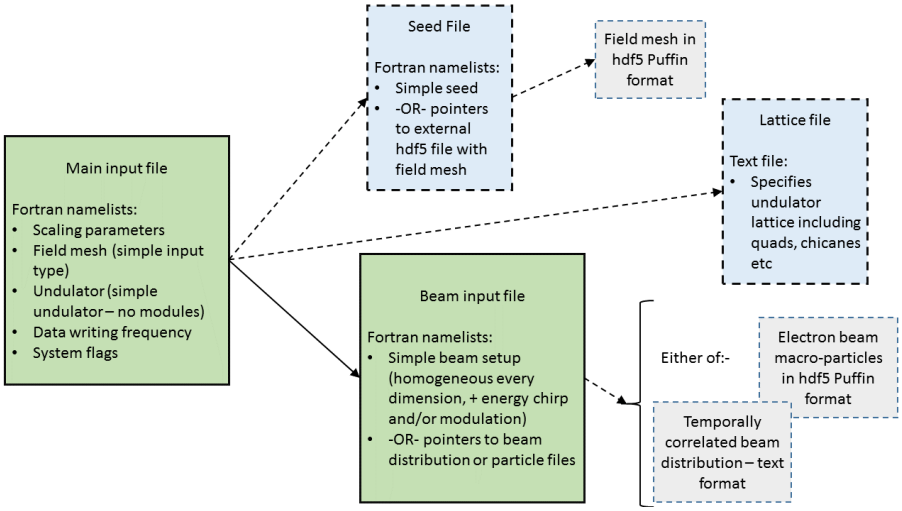

Figure 3: Schematic of input files - dashed lines indicate optional files.

not supplied, then a single undulator is set up as detailed in the main input file. If it is supplied, the wiggler in the main input file is ignored. Undulators can have independently variable polarizations, magnetic field strengths, etc. A description of available 3D undulator types is available in section 2.3.

The main input file sets the frequency of data dumps from Puffin. Additionally, a 'write file' can be supplied to give more flexibility and specify exactly when the data is written.

Puffin can be resumed from a previous run - if your cluster crashes, for instance. You could even resume from a previous run to extend or change undulator parameters, for instance.

Figure 3 shows a schematic of the Puffin input decks. Puffin comes with a few example input decks for reference.

2.2 To scale or not to scale?

Although the Puffin algorithm uses scaled variables, the beam and radiation field parameters may be input in either SI units or the scaled units. Use the

variable **qscaled** in the main input file to specify whether scaled or SI units are being used. If = **.true.**, then Puffin expects scaled variables are being used.

The main algorithm in Puffin utilizes the scaled variables above to make the numbers 'nicer' to work with. The system saturates with scaled intensity ~ 1, the scaled perpindicular momentum of the beam in the wiggler oscillates between -1 and 1, and so forth. The distance through the wiggler is given in units of a (1D) gain length, and the beam coordinate in the radiation frame is in units of the cooperation length, so that a reference electron slips $1l_c$ through the radiation field in 1 gain length. The scaling defines a frame of reference which normalises the quantities not only to numerically more manageable numbers, but to *characteristic* variables.

The two dominant variables for the scaling are ρ and η. η is a function of the wiggler parameters a_u and λ_u, and reference beam energy γ_r, and is calculated in Puffin from the input. It defines the velocity of the 'reference' electron in the scaled radiation frame. The $\bar{z}_2$ coordinate is in the scaled frame travelling with the field (at velocity c) - η determines how quickly the electrons slip back through this frame, such that an electron with energy γ_r, travelling in a wiggler with parameters a_u and λ_u, slips back with scaled velocity $p_2 \equiv \dfrac{d\bar{z}_2}{d\bar{z}} = 1$.

The ρ parameter is dependent on the peak electron beam number density, amongst other things, and as far as the discussion of the scaling parameters in Puffin goes, defines a reference for the amplification characteristics. A beam, or indeed, sections of an electron beam with peak number density giving a smaller 'localised' ρ will amplify the beam more slowly with respect to the reference beam specified by ρ.

That is to say, that they are only *scaling* parameters. The FEL parameter as input by the user does not have to correspond to the beam or wiggler being input for the results to be correct. The scaled parameters will behave 'nicely'

Table 1: Input Dimensions and Units for Beam and Radiation Field

Unscaled	units	unscaled equivalent (dimensionless)
x	metres (m)	$\bar{x}$
y	metres (m)	$\bar{y}$
dx/dz	dimensionless	$\bar{p}_x$
dy/dz	dimensionless	$\bar{p}_y$
t	seconds (s)	$\bar{z}_2$

if it does, but it is not necessary for the result to be physically correct. When the result is unscaled, the result will still be correct. The user may input in SI units, and get output in SI units. If using scaled units as input, the user must scale the lengths appropriately, but still, no beam input needs to satisfy the conditions to actually give the ρ input by the user.

2.3 3D Magnetic Fields

2.3.1 Main Undulator Models

The undulator is modelled analytically, and the model must include the fast wiggle motion. Puffin may be modified in the future to allow a map of the undulator field to be input. For now, there are a few generic undulator models employed. The undulator magnetic fields are chosen with the **zundType** string in the main input file.

Available options for use in Puffin are 'helical', 'planepole' - corresponding to a planar wiggler with flat pole faces with natural focusing only in one direction, 'curved' - correspnding to a planar undulator with curved, or canted, pole faces providing natural focusing in both transverse directions equally, and the default Puffin undulator, chosen with a blank string (''), where the transverse polarization is chosen with the **sux, suy** inputs, controlling the relative magnitudes of the peak magnetic fields in x and y, respectively. These may vary between 0 and 1, allowing a general elliptic

field to be described.

All of the undulator fields have an associated 'natural' focusing channel, which arises from from the off-axis variation in the magnetic fields. This motion arises naturally when numerically solving the equations, and is not super-imposed artificially upon the electron motion.

2.3.2 Undulator Ends

The undulators also include entry and exit tapers, and they may be switched on or off in the input file with the flag **qUndEnds**. Setting this to true will model a smooth taper up and down of the undulator magnetic fields in the first and last 2 periods of the undulator, taking the form of a $\cos^2$. If they are switched off, the beam is artificially initialized with an 'expected' initial condition in the transverse coordinates for that undulator. Including these ends will model a more realistic and natural entry and exit from the undulator, and will reduce CSE effects from the shape of the wiggler.

2.3.3 Natural Undulator Focusing

Each undulator type has an associated natural focusing wavenumber. In the helical case, the natural betatron wavenumber is

$$\bar{k}_{\beta nx} = \bar{k}_{\beta ny} = \frac{a_w}{2\sqrt{2}\rho\gamma_0}, \tag{1}$$

with γ_0 being the average energy of the electron beam (and not necessarily $= \gamma_r$, which only sets the scaling of the system.)

In the planar case,

$$\bar{k}_{\beta ny} = 0, \tag{2}$$

$$\bar{k}_{\beta ny} = \frac{a_w}{2\sqrt{2}\rho\gamma_0}. \tag{3}$$

In the canted pole case,

$$\bar{k}_{\beta nx,y} = \frac{a_w \bar{k}_{x,y}}{\sqrt{2\eta}\gamma_0}, \tag{4}$$

where $\bar{k}_{x,y}$ describe the hyperbolic variation in the transverse directions (see eqns (32 - 34)), and must obey

$$\bar{k}_x^2 + \bar{k}_y^2 = \frac{\eta}{4\rho^2} \tag{5}$$

to be physically valid. They determine the focusing strength in the $\bar{x}$ and $\bar{y}$ dimensions. For the case of equal focusing, then,

$$\bar{k}_{\beta nx} = \bar{k}_{\beta ny} = \frac{a_w}{4\rho\gamma_0}. \tag{6}$$

2.3.4 Strong Beam Focusing

In addition to the natural focusing channel, a constant, 'strong' focussing channel may be utilized, to focus the beam to a smaller transverse area. This is a magnetic field super-imposed upon the wiggler. It may be switched on or off with the flag **qFocussing** in the main input file, and is specified through the use of the variables **sKBetaXSF** and **sKBetaYSF**. It is probably highly artificial - it may be thought of as physically similar to an ion channel. Nevertheless it allows one to obtain strong focusing without using a lattice. It is defined very simply as

$$b_x = \sqrt{\eta}\frac{\bar{k}_{\beta y}^2}{\kappa}\bar{y}_j, \tag{7}$$

$$b_y = -\sqrt{\eta}\frac{\bar{k}_{\beta x}^2}{\kappa}\bar{x}_j \tag{8}$$

If either **sKBetaXSF** or **sKBetaYSF** are not specified, then no focusing channel will be added for that dimension, even if the **qFocussing** flag is true.

Magnetic quads between modules can be specified in the lattice file. See section 2.6.

2.3.5 Auto beam-matching

The beam, when specified by the 'simple' method (see below) may be matched to the focusing channel of the undulator automatically with the flag **qMatched_A** in the beam file - the option can be set for each beam. (Note this is for the natural or enhanced *undulator* focusing, and matching is not performed for a FODO lattice!) In the scaled notation,

$$\bar{\sigma}_{x,y} = \sqrt{\frac{\rho\bar{\epsilon}_{x,y}}{\bar{k}_{\beta x,y}}} \tag{9}$$

where $\bar{\epsilon}_{x,y} = \epsilon_{x,y}/(\lambda_r/4\pi)$ are the transverse emittances scaled to the so-called Kim criterion.

The spread in the transverse momentum directions is then given by

$$\bar{\sigma}_{px,py} = \frac{\sqrt{\eta}}{2\kappa}\Big\langle\frac{\Gamma}{1+\eta p_2}\Big\rangle\frac{\bar{\epsilon}_{x,y}}{\bar{\sigma}_{x,y}}. \tag{10}$$

where angular brackets indicate the ensemble average of the beam.

If **qFocussing** is true and the strong betatron wavenumber is given, then the strong betatron wavenumber is used to match the beam. Otherwise the beam is auto-matched to the natural focusing channel of the undulator. As this matching is done only for the in-undulator focusing, and not for the FODO lattice, the Twiss (α_x, α_y) input parameters will be ignored if **qMatched_A = .true.** in the input. If one wishes to use a FODO lattice, the matching must be done by the user - the matched Twiss parameters are not calculated by Puffin. See section 2.3.6 for how to input Twiss parameters.

2.3.6 Twiss Parameters

One can use Twiss parameters to provide a transverse phase-space 'tilt' for the electron beam. In the simple beam case, the user can specify the *r.m.s.* $\bar{\sigma}_x, \bar{\sigma}_{px}$ (or the S.I. equivalents σ_x, σ'_x) for the electron beam, along with the

usual α_x for the slope of the $x - x'$ correlation. Using this method, the supplied *r.m.s.* σ'_x should be the standard deviation at $x = 0$. If the emittance is supplied by using **emitx**, then the input $\bar{\sigma}_{px}$ is ignored, and is instead calculated from σ_x, the emittance, and α_x. Obviously, the intended method is to input the emittances, so that the transverse momenta parameters are over-written.

2.4 Beam Initialization

There are 3 different ways of defining and initializing the electron beam in Puffin:

2.4.1 Simple

The beam is described in terms of a homogeneous Gaussian function in every dimension. Some simple correlations in energy can be achieved by specifying an oscillation in the beam energy as a function of $\bar{z}_2$, or as a simple linear energy chirp in $\bar{z}_2$. Twiss parameters may be used to tilt the beam for matching to a FODO lattice (Puffin does not calculate the matched Twiss parameters - this must be done separate to Puffin). The beam is generated in Puffin according to this description.

In general, the correct noise statistics are added to the beam with the method described in [4]. This method, like other beam noise algorithms for FELs, requires a quiet beam to add the noise to. We include two methods of generating the beam - one with the correct noise in all dimensions, and one with the correct noise only in the temporal/$\bar{z}_2$ dimension. In both methods, the beam is initially quiet in the temporal dimension before adding the noise, with an equispaced layout of the particles in this dimension. The beam MUST be created to appropriately sample the wavelength of emission/amplification - so at least 10 equispaced particles per resonant wavelength. The

methods are:-

Equispaced Grid in Every Dimension

When using this option, the beam is intialized on an equispaced 6D grid. This requires very many particles to appropriately sample every dimension, and you may find it very easy to have more macroparticles than real electrons while loading the beam this way. However, we leave it up to the user to decide if this is appropriate - for some extreme dispersion situation with high charge it may be necessary to create the beam in this way. This can be activated by setting **qEquiXY = .true.** in the beam input. **qEquiXY** is actually an array of size nbeams, with a seperate value for each beam if multiple beams are desired. **qEquiXY = .false.** by default.

Equispaced grid ONLY in $\bar{z}_2$

If qEquiXY is false, then the beam is intially generated as a 1D beam, with equispaced macroparticles. Each macroparticle is then split into many particles in the other 5 dimensions, with coordinates generated by a quasi-random sequence in each dimension. The SAME SEQUENCE is used for each particle in z2 - the creates a series of beamlets in the $\bar{z}_2$ dimension, giving a quiet start in $\bar{z}_2$. Because random sequences are used, the other 5-dimensions require orders of magnitude less particles to adequately fill the phase space than in the equispaced case. This is the default option, so if not specified, **qEquiXY = .false.**.

Both of the above methods may, of course, be replicated and modified by the user to suit a particular situation. The creation of the beam with the correct statistics, while still retaining the sampling necessary for the FEL interaction, is still an area of ongoing research, and is quite controversial, especially when including large velocity disperion in the beam dynamics. We ultimately leave it to the user to ensure the beam is appropriately initialized for their situation. The methods above can probably be replicated with just a few lines of *e.g.* Python. The generated beam can then be input into

Puffin.

To use this, set the option dtype = 'simple' in the nblist namelist in the beam file. The parameters for the distribution are then set in the blist namelist in the same file.

2.4.2 Dist input

The input is composed of a description of temporal slices along the bunch, describing a Gaussian mean and standard deviation in every other dimension, along with Twiss α parameter, for each temporal slice. This information is used to generate the electron beam in Puffin. The slices must be spaced finely enough to sample the radiation field wavelength being amplified - so *e.g* at least 10 per wavelength.

2.4.3 Beam input

This method allows one to read in the 6D scaled particle coordinates from an HDF5 file or a text file. The HDF5 file should be in the same format as the Puffin output. The text input is not recommended, especially for 3D runs, since the reading in is very inefficient, and the files are usually huge for 3D data. The beam is generated externally, and the user is then responsible for ensuring the sampling and the noise in the pulse are correct - it is not added by Puffin in this method.

2.5 Field Mesh

The radiation field in Puffin is modelled by a simple cartesian mesh, with equispaced nodes in each of the 3 spatial dimensions. Each node samples a value of the x and y polarized radiation field (the z component of the electric field is not modelled). The scaled $\bar{z}_2$ coordinate frame is such that the back of the system is with increasing $\bar{z}_2$ - *i.e.* to the right in figure 4. Recall that

$\bar{z}_2$ is the stationary radiation frame, so that as the beam propagates through the undulator, the beam slips back through the radiation field, moving to the right in figure 4.

This mesh must be large enough to contain the beam through the entire propagation distance through the wiggler.

It must also be large enough to contain the beam in the transverse plane, see figure 5. The radiation field is calculated from the driving electron beam by linear interpolants, so the field must adequately sample the area which the beam occupies. The radiation also diffracts outwards from the beam, so the field mesh must extend to adequately model this diffracting radiation to avoid numerical problems at the boundaries of the grid. There are absorbing boundaries in the outer 16 nodes in the mesh in x and y to mitigate this issue, but an absorbing boundary which works well for the full range of frequencies modelled by Puffin is difficult to realise, and the absorbing boundary should not be relied upon to absorb everything which propagates to the outer reaches of the mesh.

2.5.1 Fixing the Radiation Mesh

One may use the variables **iRedNodesX, iRedNodesY** to fix the mesh around the beam when initializing a simple beam - see figure 5. These variables define an inner set ('reduced' set) of nodes which the beam width will occupy. So the mesh length in x and y will be set up such that the inner set of nodes defined by **iRedNodesX, iRedNodesY** will contain the beam. In this case, the **sFModelLengthX, sFModelLengthY** inputs will be ignored.

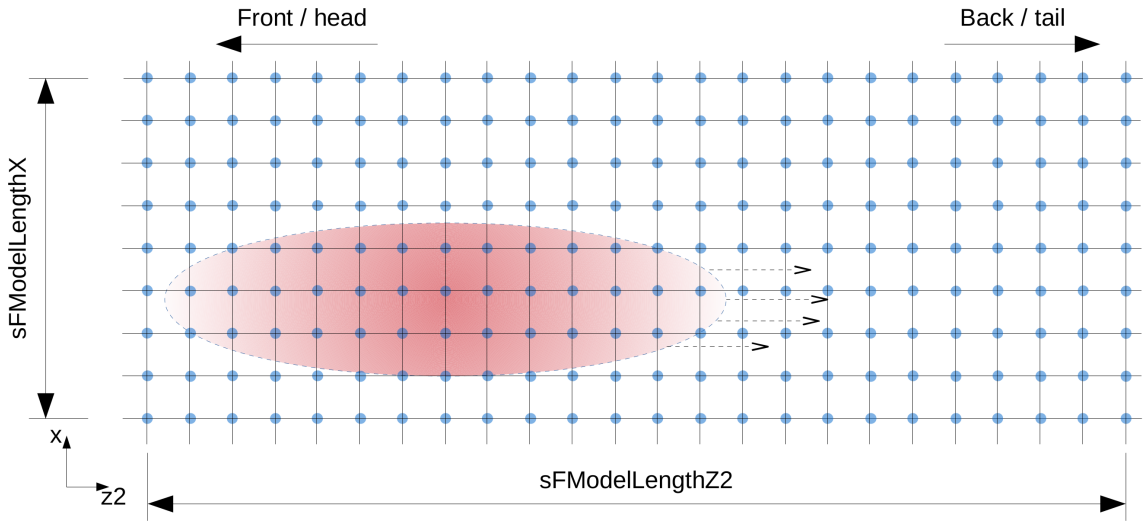

Figure 4: Showing the longitudinal setup of the radiation mesh. The beam is indicated in red. The scaled $\bar{z}_2$ coordinate is defined so that the front of the radiation and beam is to the left. This is the constant radiatioon frame, so the beam slips backwards through the field from left to right. In the lab frame, the beam and radiation propagates from right to left. The full length of the mesh must be large enough to contain the beam through propagation.

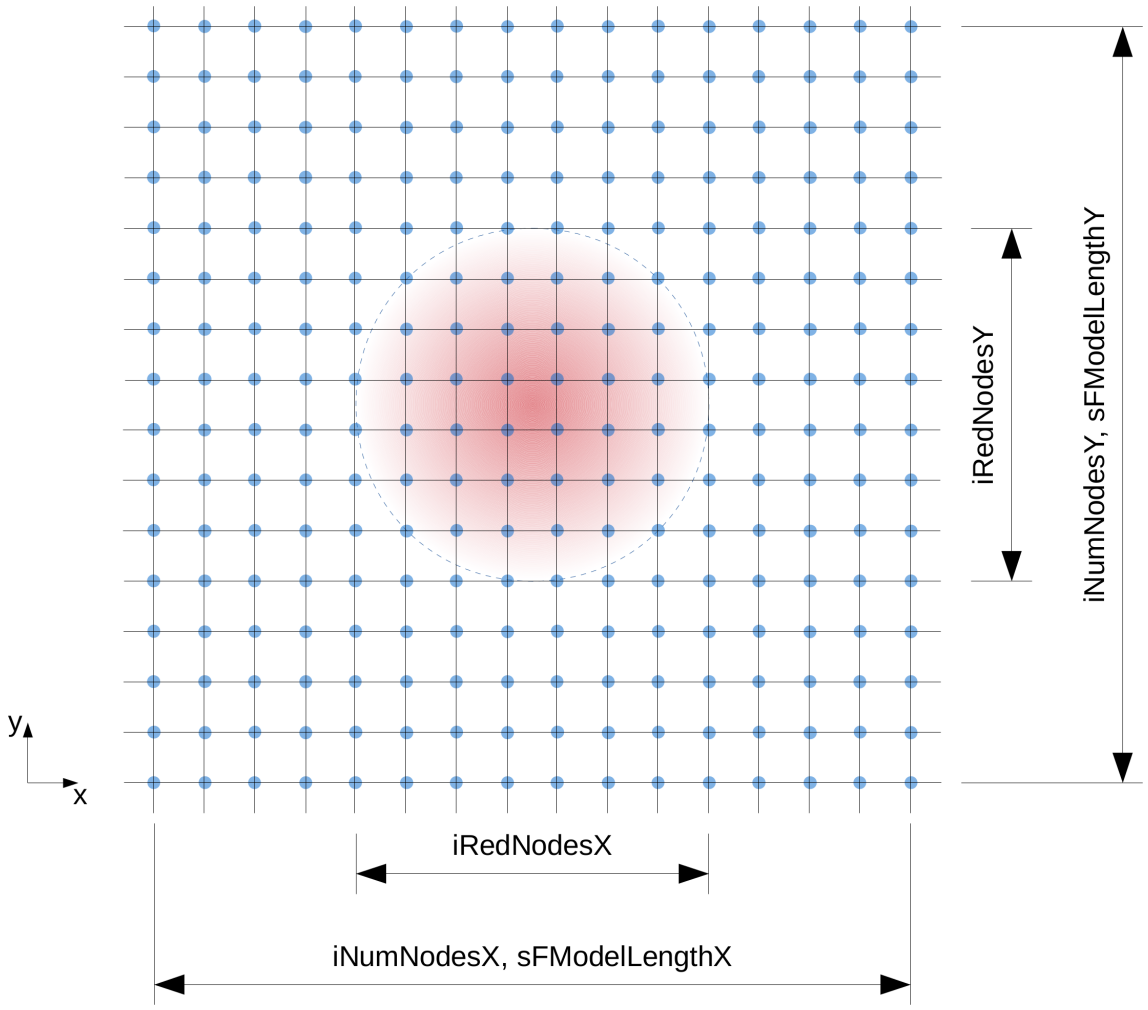

Figure 5: Transverse mesh to model the radiation field - the beam is indicated in red. The mesh may be 'matched' to the transverse beam profile initially by specifying the iRedNodes parameters - these specify an inner set of nodes such that the mesh will be setup so that the beam is contained within this number of nodes (when using the simple beam input type only). This is only an initial condition, but can aid in setting up the mesh.

Table 2: Accelerator components for use in lattice file

Component	ID Initials
Undulator	**UN**
Quadrupole	**QU**
Chicane	**CH**
Drift	**DR**
Modulation	**MO**

2.6 Lattice Input

A lattice file may be supplied in the main input file with the variable 'lattFile'. This is the name of the lattice file, containing details of the undulator module lattice layout. If blank, then a single undulator as specified in the main input file will be used.

The lattice file is a plain text file, where each line beginning with one of the two-letter identifiers below, and in table 2, represents a lattice element. There are five supported lattice element types, being undulators, quads, chicanes, drifts, and modulations. These are specified in the lattice file by the first 2 characters of each line, followed by (on the same line) the options for the element. Any line not beginning with one of these identifiers is ignored.

Each entry must be followed by the options specifying that lattice element, on the same line, delimited by spaces.

UN - Undulator Module

For the undulator, one must specify the undulator type, the number of wiggler periods N_w, the undulator parameter relative to the base undulator parameter in the main input file α, the linear taper $\frac{d\alpha}{d\bar{z}}$, the polarization u_x and u_y (ignored for the non-variable polarized undulator types), a logical specifying if undulator ends are being used for this module, another logical spcifying if a strong focusing channel is being used in the undulator, and the strong focusing strengths $\bar{k}_{\beta x}$ and $\bar{k}_{\beta y}$, and the number of integration steps

per period to be used in this module.

QU - Quadrupole

For the quad, we must specify $\bar{F}_x$ and $\bar{F}_y$. These are scaled transverse components of the transform matrix for the quad. The quad is only a point transform, currently, and does not have a length, so the length must be included in a drift section. The transforms for the quad, in S.I. units, are defined as

$$\left.\frac{dx_j}{dz}\right|_{new} = \left.\frac{dx_j}{dz}\right|_{old} + \frac{1}{F_x}x_j \tag{11}$$

$$\left.\frac{dy_j}{dz}\right|_{new} = \left.\frac{dy_j}{dz}\right|_{old} + \frac{1}{F_y}y_j \tag{12}$$

which in the scaled notation becomes:

$$\left.\frac{d\bar{p}_{xj}}{d\bar{z}}\right|_{new} = \left.\frac{d\bar{p}_{xj}}{d\bar{z}}\right|_{old} + \frac{\sqrt{\eta}}{2\rho\kappa}\frac{\bar{x}_j}{\bar{F}_x} \tag{13}$$

$$\left.\frac{d\bar{p}_{yj}}{d\bar{z}}\right|_{new} = \left.\frac{d\bar{p}_{yj}}{d\bar{z}}\right|_{old} + \frac{\sqrt{\eta}}{2\rho\kappa}\frac{\bar{y}_j}{\bar{F}_y} \tag{14}$$

where $\bar{F}_{x,y} = \dfrac{F_{x,y}}{l_g}$.

In x (unscaled) focal length is taken as

$$\frac{1}{F_x} = \frac{gL_Q}{B\rho}, \tag{15}$$

where g is the quad magnetic field gradient in Tm^{-1}, L_Q is the length of the quad, and $B\rho = \beta E_0/0.2998$ is the so-called magnetic rigidity in Tm, with the beam energy E_0 in GeV (corresponding to the scaling energy defined in the input file).

CH - Chicane

For the chicane, one must specify physical length, slippage length, and dispersion enhancement (scaled R56). The length is the physical length of the device in undulator periods, and is used to calculate the diffraction properly.

The slippage length tells how many wavelengths to delay the beam by relative to the radiation field in resonant radiation wavelengths - be careful, this should not be ≤ 0, or the beam will then be faster than light speed! Similarly, it is left up to the user to decide what is appropriate to delay the beam by with respect to the physical length of the device. The dispersive enhancement, D, is the scaled R56 of the device - using γ_r in the main input as the mean γ for the chicane, so that

$$\bar{z}_2 = \bar{z}_{20} - 2D\left(\frac{\gamma - \gamma_r}{\gamma_r}\right), \tag{16}$$

and $D = k_r \rho R_{56}$.

This approach of being able to vary the 3 parameters above is quite flexible, but at the expense of easy automatic checking of the physical relevance of the parameters. For example, one can disperse the beam 'in place', without shifting it w.r.t. the resonant wavelength, which is obviously unphysical. However, this may be useful for a quick setup of the beam for EEHG at the beginning of the undulator, for instance. If one is not sure of the physical length of the actual device, but knows the length between modules and the desired slippage (e.g. for mode locking the FEL), then one can specify the spatial drift between modules using a drift section, and then specify a chicane with zero physical length, but with the desired delay and R56. In this way the radiation diffraction will still be modelled correctly.

DR - Drift

For the drift, one must specify length in units of the number of undulator periods.

MO - Modulation

For the modulation section, one may add an energy modulation to the beam specified by $\bar{k}_M = \dfrac{2\pi}{\bar{\lambda}_M}$ and $\dfrac{\Delta\gamma}{\gamma_r}$.

3 Numerical Model

4 Data Output and Post-Processing

4.1 Data Output

Data output is classified as either 'full' or 'integrated' (reduced) data. The full data dumps consist of the x and y field values on the field mesh, and the macroparticle coordinates in 6D phase space. In a 3D run, this data can easily be $>$ 100GB's per dump, so it is advisable to limit the frequency of the dumps, which is controlled in the main input file by the variable **iWriteNthSteps**.

On the other hand, the reduced data sizes are much smaller, and typically can be written every undulator period. It can still be quite large, depending on the modelled FEL and length of the run, but it will always be significantly smaller than the full data dumps for 3D runs. Integrated data includes quantities like radiation power, bunching, energy spread, current, emittance, beam radius *etc* as functions of $\bar{z}_2$ or $ct - z$. Frequency of the integrated dumps is controlled with **iWriteIntNthSteps** in the main input file.

4.2 Output Format

When running Puffin, a few different files will be produced besides the data output files. A *.log file provides any information on errors. A rec.out file tracks the step number the simulation is at in the main integration loop (this is also printed to standard out, but on some job submission systems the stdout will not be visible till AFTER the job has finished. rec.out should be available as the job is running to track the job progress).

The data output is usually in the HDF5 file format, or optionally in the SDDS format. Currently, SDDS support is quite limited, and full data dumps and only the radiation power of the integrated data is output. Much more integrated data is output in the HDF5 files.

4.2.1 HDF5

The HDF5 file naming convention in Puffin is: <basename> _ <type> _ <step>.h5, where basename is the input file basename (i.e.the input file name without the postfix '.in')

Type can be integrated, aperp, or electrons, corresponding to reduced or integrated data, a full dump of the field mesh (x and y polarised fields), and a full dump of the electron macroparticle coordinates + charge weight. These are described individually in more detail below. Due to the possible extremely large size of the full dumps, one should be careful how often the data is being written.

One may simply choose which steps to write out through use of the **iWriteNthSteps** and **iWriteIntNthSteps** parameters in the main input file. The former controls the frequency of the full dumps, and the latter specifies the frequency of the integrated data writes.

In addition, one may activate the **qDumpEnd** flag to write a full dump at the end of the simulation - this is **.true.** by defualt, but may be switched off.

More useful is the write file. This allows one to specify the exact step numbers at which to write out the full dumps. This will be done in addition to the regular write intervals specified above. The file is pointed to from the main input file by the string **wr_file**. In this plain text file, specify each step to write at with a line beginning with **WR**, followed by a space (or multiple spaces) and then the step number. Any line not beginning with **WR** will be ignored.

4.3 Output Format Specification

Puffin uses primarily hdf5 for output, although sdds dumps can be made if desired. The sdds outputs do not contain the integrated data, except power.

For input filename = **basename.in**, then outputs are as described:

4.3.1 Full Dumps

Two files for each step written out - one for the field mesh, and one for the macroparticle positions.

Field Mesh

Named ***basename*_aperp_*stepnum*.h5**, where **stepnum** is the integration step number the data was written out at. The file has internal hdf5 structure:

```
/ (RootGroup)
/aperp (Array(2, nz2, ny, nx))
/globalLimits (Group)
/intensityScaled (Group)
/meshScaled (Group)
/runInfo (Group)
/time (Group)
```

The **aperp** dataset contains the values of the field at each field node. **nz2**, **ny** and **nx** are the number of nodes in the equispaced field mesh in $\bar{z}_2$, $\bar{y}$, and $\bar{x}$, respectively. The data **aperp(1,:,:,:)** contains the values of the x-polarized field, and **aperp(2,:,:,:)** is the *negative* y-polarized field. It also contains some data about the run itself (some of this data is also in the runInfo group in the file). An example of the data is:-

```
iChic_cr := 1,
iCsteps := 0,
iDrift_cr := 1,
iL := 0,
iModulation_cr := 1,
iQuad_cr := 1,
```

```
iUnd_cr := 1,
istep := 0,
numSpatialDims := 3,
time := 0.0,
vsAxisLabels := 'xbar,ybar,z2bar',
vsCentering := 'nodal',
vsIndexOrder := 'compMajorF',
vsLabels := 'aperp_real,␣aperp_imaginary',
vsLimits := 'globalLimits',
vsMesh := 'meshScaled',
vsNumSpatialDims := 3,
vsTimeGroup := 'time',
vsType := 'variable',
zInter := 0.0,
zLocal := 0.0,
zTotal := 0.0,
zbarInter := 0.0,
zbarLocal := 0.0,
zbarTotal := 0.0
```

The integers ending in 'cr' are counters for the lattice elements which may be present - **iChic** is for the chicane, **iQuad** is for the quads, **iUnd** is for the undulator modules, **iModulation** is for the beam modulator sections, and **iDrift** is for the drift sections. The counter number indicates that the beam has been been completely through the counter number -1 (so in the example shown, it has been through no components yet.)

istep is the number of completed integration steps through the current undulator module. It is local to the module being integrated through when the data was written.

iCsteps is the number of *total* (cumulative) integration steps completed

through the entire undulator line, including all previous undulators.

The **zInter, zLocal** and **zTotal** values are distances through the machine, in metres. **Inter** indicates the *interaction* distance through the machine - that is, the distance of undulators only, discounting the drifts, chicanes, etc. **Local** is the distance through the local undulator module. **Total** is the total distance through the machine when the data was written.

zbarInter, zbarLocal and **zbarTotal** correspond to **zInter, zLocal** and **zTotal**, scaled to L_g, the 1D gain length.

The **intensityScaled** group is a VizSchema derived variable, which tells Visit how to calculate the intensity from the scaled field data.

The **meshScaled**, **globalLimits** and **time** groups are VizSchema metadata for *e.g.* Visit.

runInfo is a group containing data about the run the data is from. It includes data about the physical parameters, the scaling and the mesh sizes, amongst other things. A listing of the metadata in this group is here:

```
Lc := 1.5887624436132764e-06,
Lg := 0.43767608132326558,
aw := 1.0121808999999999,
eta := 3.6299960436719031e-06,
gamma_r := 456.39999999999998,
iChic_cr := 1,
iDrift_cr := 1,
iModulation_cr := 1,
iQuad_cr := 1,
iUnd_cr := 1,
kappa := 0.22177495617879053,
lambda_r := 9.9824891200977331e-08,
lambda_w := 0.0275,
nSteps := 870,
```

```
nX := 120,
nY := 120,
nZ2 := 161939,
npk_bar := 302255429.89145231,
rho := 0.0050000000000000001,
sLengthOfElmX := 0.033613445378151259,
sLengthOfElmY := 0.033613445378151259,
sLengthOfElmZ2 := 0.0033069325981203339,
sStepSize := 0.002094395160675049,
vsBeamFile := 'beam_file',
vsBuildConfigDate := '2017-02-07 20:04',
vsBuildHost := 'crb101',
vsBuildHostType := 'Linux-2.6.32-642.4.2.el6.x86_64',
vsCommandLine := '/gpfs/stfc/local/HCP084/bwm06/ltc84-bwm06/bin/puf
vsFCompiler := 'ifort',
vsFCompilerFlags := '  -open',
vsFCompilerVersion := '15.0.2',
vsInputFile := 'test1',
vsRunDate := '2017-02-12  18:29:48.648 -0000',
vsRunHost := 'nxb3b04',
vsSeedFile := 'seed_file',
vsSoftware := 'PUFFIN',
vsSwRevision := 'dev-twiss : 0e2c73483f1dd246a26363292f932946cd1917c
vsSwVersion := '1.8.0',
vsType := 'runInfo',
vsUser := 'root',
vsVsVersion := '3.0'
```

sLengthOfElmX, **sLengthOfElmY** and **sLengthOfElmZ2** are the distances between field nodes in the respective directions in the field mesh,

in scaled notation. The other data should be self-explanatory.

Beam Dump

The electron beam files are named ***basename*_electrons_*stepnum*.h5**. The structure of the file is

```
/ (RootGroup)
/electrons (Array(numMacroParticles, 7))
/electrons_chargeSI (Group)
/electrons_gammaSI (Group)
/electrons_numPhysicalParticles (Group)
/electrons_pxSI (Group)
/electrons_pySI (Group)
/electrons_xSI (Group)
/electrons_ySI (Group)
/electrons_zSI (Group)
/globalLimits (Group)
/phi_lamda (Group)
/runInfo (Group)
/slice_nom_lamda (Group)
/time (Group)
```

The main data is the **electrons** dataset, where the each column corresponds to the macroparticle $\bar{x}$, $\bar{y}$, $\bar{z}_2$, $\bar{p}_x$, $-\bar{p}_y$, and Γ coordinates, and the macroparticle weight, respectively (note the negative sign in the $\bar{p}_y$ data). The weight $= N_e/\bar{n}_p$, where N_e is the number of real electrons the macroparticle represents, and $\bar{n}_p$ is the peak number density of the electron beam required to get the input value of ρ. $\bar{n}_p$ is in the **runInfo** group in the file.

The **...SI** groups are VizSchema derived datasets, allowing visit to calulate the SI quantities from the scaled data in the file.

The **runInfo** group is as specified in the field dump file.

4.3.2 Integrated Data Files

The integrated data files are named ***basename*_integrated_*stepnum*.h5**. They contain the following data:

```
/ (RootGroup) ''
/beamCurrent (Array(8525,)) ''
/beamCurrentSI (Array(8525,)) ''
/bunching2ndHarmonic (Array(8524,)) ''
/bunching2ndHarmonicSI (Array(8524,)) ''
/bunching3rdHarmonic (Array(8524,)) ''
/bunching3rdHarmonicSI (Array(8524,)) ''
/bunching4thHarmonic (Array(8524,)) ''
/bunching4thHarmonicSI (Array(8524,)) ''
/bunching5thHarmonic (Array(8524,)) ''
/bunching5thHarmonicSI (Array(8524,)) ''
/bunchingFundamental (Array(8524,)) ''
/bunchingFundamentalSI (Array(8524,)) ''
/emittanceX (Array(8524,)) ''
/emittanceXSI (Array(8524,)) ''
/emittanceY (Array(8524,)) ''
/emittanceYSI (Array(8524,)) ''
/meanDeltaGamma (Array(8524,)) ''
/meanDeltaGammaSI (Array(8524,)) ''
/meanGamma (Array(8524,)) ''
/meanGammaSI (Array(8524,)) ''
/meanMomentumX (Array(8524,)) ''
/meanMomentumXSI (Array(8524,)) ''
/meanMomentumY (Array(8524,)) ''
/meanMomentumYSI (Array(8524,)) ''
```

```
/meanPosX (Array(8524,)) ''
/meanPosXSI (Array(8524,)) ''
/meanPosY (Array(8524,)) ''
/meanPosYSI (Array(8524,)) ''
/power (Array(161939,)) ''
/powerSI (Array(161939,)) ''
/sigmaX (Array(8524,)) ''
/sigmaXSI (Array(8524,)) ''
/sigmaY (Array(8524,)) ''
/sigmaYSI (Array(8524,)) ''
/sigmapX (Array(8524,)) ''
/sigmapXSI (Array(8524,)) ''
/sigmapY (Array(8524,)) ''
/sigmapYSI (Array(8524,)) ''
/sliceCharge (Array(8524,)) ''
/sliceChargeSI (Array(8524,)) ''
/globalLimits (Group) ''
/globalLimitsSI (Group) ''
/intFieldMeshSI (Group) ''
/intFieldMeshSc (Group) ''
/intPtclMeshSI (Group) ''
/intPtclMeshSc (Group) ''
/runInfo (Group) ''
/time (Group) ''
```

Data with an appended **SI** indicate the measurement is in SI units. Otherwise, the data is in the scaled notation.

4.4 Post-processing

The powPrep.py script will process the power data in the integrated data files to produce a file named basename-integrated-all.vsh5, which contains the power data normalised and formatted for use as a surface plot in Visit, to show the evolution of the temporal power profile as a function of distance through the undulator, see figure. It also contains the energy and peak power data for plotting.

Before viewing in VisIt, if desired (see below), the field dumps need to be processed by the 'ReorderColMajorFtoColMinorC.py' script, since there is currently a bug in VisIt which prevents the data being opened in the native fortran ordering. The field files can easily be 10's to 100's of GB's (or even larger) - so you may wish to process them anyway before attempting to view them.

4.5 Viewing the Data

Users are free to choose their favourite visualisation method, and none is 'officially' supported by Puffin. Nevertheless, the data output itself has vizSchema metadata to tell *e.g.* Visit how to display the data. VisIt allows the ability to plot the data in client-server mode while the data is on a cluster, which is useful when doing large runs.

There are example Python scripts to subselect portions of the field mesh. For example, you may wish to select only a central portion of the field mesh in the transverse or temporal directions, or you may wish to select portions at a a time to average over, etc.

In addition, there are some scripts supplied to give examples of how to access, manipulate and plot the data in Matlab and Python/matplotlib. Feel free to modify these for your own purposes - we expect each project to require its own bespoke processing just due to the nature of the code.

5 Analytic Equations Solved by Puffin

The system of equations solved by Puffin have been altered from those described in [1]. They are now:

$$\Big[\frac{1}{2}\Big(\frac{\partial^2}{\partial\bar{x}^2}+\frac{\partial^2}{\partial\bar{y}^2}\Big)-\frac{\partial^2}{\partial\bar{z}\partial\bar{z}_2}\Big]A_\perp=-\frac{1}{\bar{n}_p}\frac{\partial}{\partial\bar{z}_2}\sum_{j=1}^{N}\frac{\bar{p}_{\perp j}}{\Gamma_j}(1+\eta p_{2j})\delta^3(\bar{x}_j,\bar{y}_j,\bar{z}_{2j}) \tag{17}$$

$$\frac{d\bar{p}_{\perp j}}{d\bar{z}}=\frac{1}{2\rho}\Big[i\alpha b_\perp-\frac{\eta p_{2j}}{\kappa^2}A_\perp\Big]-i\kappa\frac{\bar{p}_{\perp j}}{\Gamma_j}(1+\eta p_{2j})\alpha b_z \tag{18}$$

$$\frac{d\Gamma_j}{d\bar{z}}=-\rho\frac{(1+\eta p_{2j})}{\Gamma_j}(\bar{p}_{\perp j}A^*_{\perp j}+c.c.) \tag{19}$$

$$\frac{d\bar{z}_{2j}}{d\bar{z}}=p_{2j} \tag{20}$$

$$\frac{d\bar{x}_j}{d\bar{z}}=\frac{2\rho\kappa}{\sqrt{\eta}\Gamma_j}(1+\eta p_{2j})\Re(\bar{p}_{\perp j}) \tag{21}$$

$$\frac{d\bar{y}_j}{d\bar{z}}=-\frac{2\rho\kappa}{\sqrt{\eta}\Gamma_j}(1+\eta p_{2j})\Im(\bar{p}_{\perp j}). \tag{22}$$

Scaled parameters are:-

$$\bar{z}_{2j}=\frac{ct_j-z}{l_c},\qquad \bar{z}=\frac{z}{l_g},$$

$$\bar{p}_\perp=\frac{p_\perp}{mca_u},\qquad A_\perp=\frac{e\kappa l_g}{\gamma_0 mc^2}E_\perp,$$

$$(\bar{x},\bar{y})=\frac{(x,y)}{\sqrt{l_g l_c}},\qquad l_g=\frac{\lambda_w}{4\pi\rho},$$

$$l_c=\frac{\lambda_r}{4\pi\rho},\qquad \Gamma_j=\frac{\gamma_j}{\gamma_0},$$

$$\rho=\frac{1}{\gamma_0}\left(\frac{a_u\omega_p}{4ck_u}\right)^{2/3},\qquad a_u=\frac{eB_0}{mck_u},$$

$$\kappa=\frac{a_u}{2\rho\gamma_0},\qquad b_\perp=b_x-ib_y,$$

B_0 is the peak magnetic field in the wiggler. $\omega_p = \sqrt{e^2 n_p/\epsilon_0 m}$ is the (non-relativistic) plasma frequency, and n_p is the peak spatial number density of the electron beam ($N_e/\delta_x\delta_y\delta_z$). $E_\perp = E_x - iE_y$ are the x and y radiation electric field vectors. γ_0 is the reference energy (Lorentz factor), which is usually taken as the mean beam energy.

The scaled reference velocity,

$$\eta = \frac{1-\beta_{zr}}{\beta_{zr}} = \frac{\lambda_r}{\lambda_u} = \frac{l_c}{l_g}, \tag{23}$$

where β_{zr} is some reference velocity scaled to c, which is sensible (but not, strictly speaking, necessary) to take as the mean longitudinal electron velocity in the wiggler, so that

$$\beta_{zr} = \sqrt{1 - \frac{1}{\gamma_0^2}\Big(1 + \bar{a}_u^2\Big)}, \tag{24}$$

where $\bar{a}_u$ is the rms undulator parameter. This defines the velocity at which the electrons travel in the scaled $\bar{z}_2$ frame. More generally, η describes an *ideal* resonance condition - electrons resonant with wavelength λ_r will travel with velocity $p_2 = 1$ through the $\bar{z}_2$ frame.

p_{2j} may be worked out analytically from

$$p_{2j} = \frac{1}{\eta}\Big[\Big(1 - \frac{(1 + a_u^2|\bar{p}_{\perp j}|^2)}{\gamma_r^2\Gamma_j^2}\Big)^{-1/2} - 1\Big] \tag{25}$$

at each step. It is NOT output from Puffin.

Outputs from Puffin are classed as either 'integrated' data or a full 'dump.' The integrated data includes things like the power, beam radius, current, bunching, emittance, etc, as a function of temporal coordinate. The full data dump is the raw data, including the values of the scaled radiation field $A_\perp$ mesh, and the 6D electron macroparticle phase space coords $(\bar{x}, \bar{y}, \bar{z}_2, \bar{p}_\perp, \Gamma)$ and the macroparticle charge weights. Data is in hdf5 format by default, with vizschema metadata for viewing in Visit.

Compared to the original Puffin paper [1], now the peak magnetic fields are used to scale the system of eqns (as opposed to the rms values used previously). Previously (*e*.g. as described in [1]), the energy exchange was modelled through the scaled longitudinal velocity p_2. This was problematic in multiple ways from a computational point of view, since p_2 is a function of the energy, p_x and p_y, so that the same quantity was essentially being calculated twice, leading to large errors in some cases. The energy exchange is instead now modelled directly through the scaled variable $\Gamma_j = \gamma_j/\gamma_r$, meaning smaller numerical errors and enabling a significantly larger step size.

The formalism for including a variation in the magnetic undulator field, so that $B_0(\bar{z})$ now varies, was included in [2], and is varied using the parameter $\alpha(\bar{z}) = \dfrac{B_0(\bar{z})}{B_0(\bar{z}=0)}$.

5.1 Full Magnetic Undulator Field Analytic Description

The analytic description of the undulator fields are based on those discussed in [3], which in the scaled notation here are:-

helical

$$b_x = \cos(\bar{z}/2\rho) \tag{26}$$

$$b_y = \sin(\bar{z}/2\rho) \tag{27}$$

$$b_z = \frac{\sqrt{\eta}}{2\rho}(-\bar{x}\sin(\bar{z}/(2\rho)) + \bar{y}\cos(\bar{z}/(2\rho))) \tag{28}$$

plane-pole

$$b_x = 0 \tag{29}$$

$$b_y = \cosh((\sqrt{\eta}/2\rho)\bar{y})\sin(\bar{z}/2\rho) \tag{30}$$

$$b_z = \sinh((\sqrt{\eta}/2\rho)\bar{y})\cos(\bar{z}/(2\rho) \tag{31}$$

canted-pole

$$b_x = \frac{\bar{k}_{\beta x}}{\bar{k}_{\beta y}} \sinh(\bar{k}_{\beta x}\bar{x}) \sinh(\bar{k}_{\beta y}\bar{y}) \sin(\bar{z}/2\rho) \tag{32}$$

$$b_y = \cosh(\bar{k}_{\beta x}\bar{x}) \cosh(\bar{k}_{\beta y}\bar{y}) \sin(\bar{z}/2\rho) \tag{33}$$

$$b_z = \frac{\sqrt{\eta}}{2\rho\bar{k}_{\beta x}} \cosh(\bar{k}_{\beta x}\bar{x}) \sinh(\bar{k}_{\beta y}\bar{y}) \cos(\bar{z}/2\rho) \tag{34}$$

variably polarized elliptical

$$b_x = u_x \cos(\bar{z}/2\rho) \tag{35}$$

$$b_y = u_y \sin(\bar{z}/2\rho) \tag{36}$$

$$b_z = \frac{\sqrt{\eta}}{2\rho}(u_x\bar{x} \sin(\bar{z}/(2\rho)) + u_y\bar{y} \cos(\bar{z}/(2\rho))) \tag{37}$$

In the 1D approximation, $b_z = 0$.

All of the above have an associated 'natural' focusing channel, which arises from from the off-axis variation in the magnetic fields. This motion arises naturally when numerically solving the equations, and is not superimposed upon the electron motion.

5.1.1 Magnetic Undulator Field Ends

Figure 6 shows the entry tapers. The x field envelope inceases more rapidly so that the first 'pole' is at full strength. The x field is symmetric, and the y field is antisymmetric. The ends of the x field shift the beam off-center by a small amount in the y plane. The x plane motion is not off-set by the y field. Neither causes a velocity offset (or spatial drift) along the undulator. Note that the symmetry of the x field technically results in an extra half-period of full-strength field oscillation in that direction.

These same entry/exit tapers are used for all of the undulator types in Puffin.

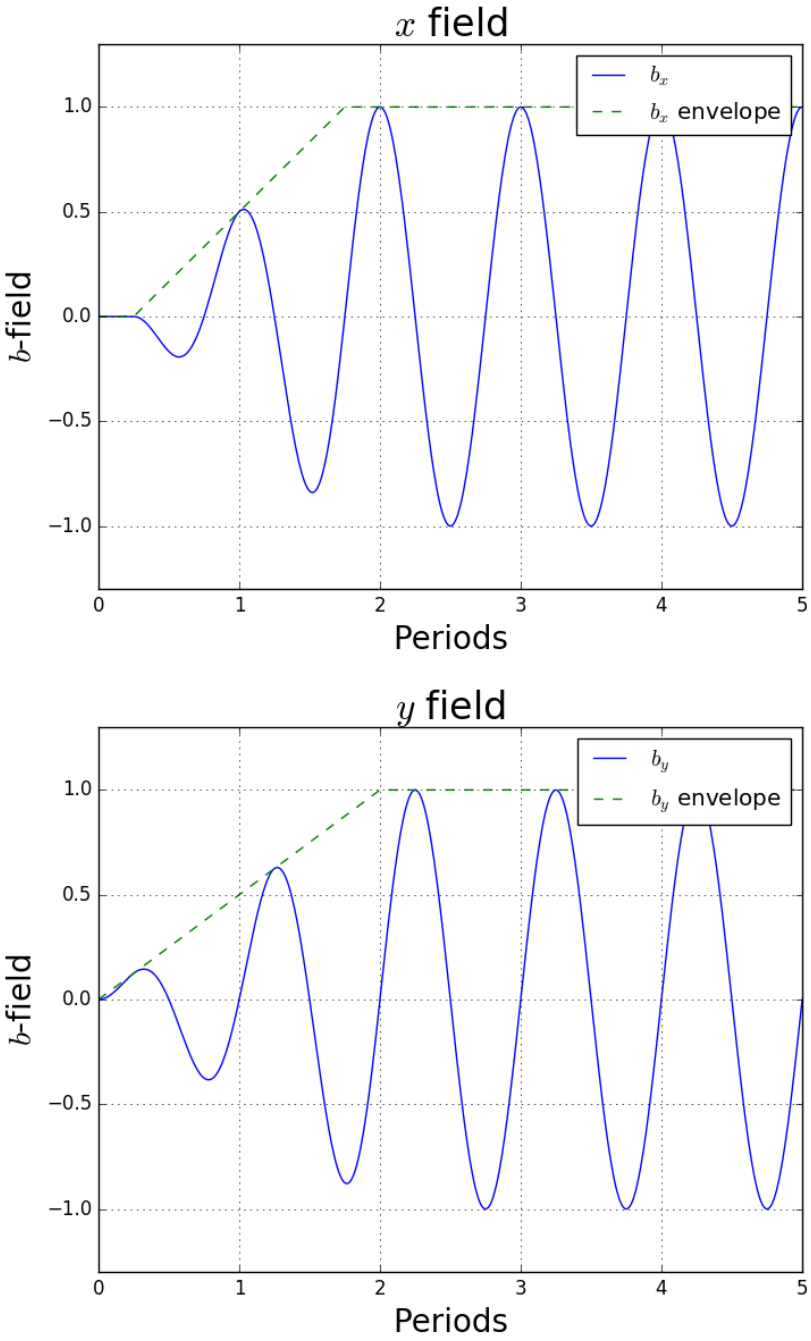

Figure 6: Schematic of undulator field ends. The magnetic poles are increased linearly over the first 2 periods as shown.

6 Parameters

Below is an exhaustive list of parameters, for each input file type. Some options are only really useful for investigating runtime errors or strange behaviour. The parameter list for each file is loosely sorted from most to least useful to the average user (default values are in square brackets after the description).

6.1 Main Input File

sRho

ρ, the FEL parameter. This specifies the scaling of the system. It does **not** have to be strictly correct - it only descibes the scaling. The simulation can be performed, and the system may be scaled back to SI units, and the result should be correct. If the supplied ρ is correct, however (meaning, it has been calculated from the beam and undulator parameters input), then the scaled notation becomes physically relevant. For an ideal, 1D system, the system should saturate at intensity $|A|^2 \approx 1$, and the system should be firmly in the high or exponential gain regime at $\bar{z} \approx 2 - 3$. This allows one to quickly see how efficiently the system is lasing w.r.t its ideal case. [0.01]

saw

a_u, the undulator parameter defined with the peak on-axis magnetic field. [1.0]

sgamma_r

γ_0, the reference (usually the mean) beam energy. [100]

lambda_w

λ_u, the undulator, or wiggler, period. [0.04]

zundType

Allows one to select from a choice of undulators. Choices are 'curved', 'planepole', and 'helical', described analytically below. If neither of these

are chosen, then the default elliptical undulator is chosen, with polarization specified by u_x and u_y. [‘’ (blank - corresponding to the generic variably polarized undulator]

sux, suy

u_x and u_y - the relative peak magnetic field of the undulator in each transverse polarization. Usually, at least one should be = 1, and the other between 0 (planar) and 1 (helical). If they are not defined, the default is helical, $u_x = u_y = 1$. Ignored if the undulator type is specified as either ‘helical’, ‘planepole’ or ‘curvedpole’. [$(u_x = u_y = 1.0)$]

taper

Taper in the single, simple undulator case, $\frac{d\alpha}{d\bar{z}}$. Ignored if a lattice file is supplied. [0.0]

iWriteNthSteps

Steps at which to write the full data dumps at. This sets the periodicity of the data dumps, in integration steps. [30]

iWriteIntNthSteps

Steps at which to write the integrated data at. This sets the periodicity of the writes, in integration steps. [30]

sFiltFrac

During the field diffraction step, the lower frequencies are filtered out. This sets the cutoff for the high-pass filter, in units of the reference resonant frequency - (so should usually be ≤ 0.3). [0.3]

sDiffFrac

Length of the diffraction step, in units of the undulator period. Usually = 1. [1.0]

sBeta

Absorption coefficient for the absorbing boundaries on the transverse grid. [1.0]

qOneD

Logical. If **.true.**, Puffin will be run in 1D mode. Default is **.true.**, since this is the least computationally expensive option. [.true.]

qFieldEvolve

Logical. If = **.false.**, then the radiation field evolution will be switched off. This is obviously artificial and very rarely used, but can be useful for debugging in some instances. Default is = **.true**. [.true.]

qElectronsEvolve

Logical. If = **.false.**, then the electron equations are not solved in the integration steps. Like the case when qFieldEvolve= **.false.**, this is obviously artificial and very rarely used, but can be useful for debugging in some instances. Default is = **.true.**. [.true.]

qElectronFieldCoupling

Logical. If = **.false.**, then the radiation field feedback onto the electrons is switched off by setting the field terms in the electron evolution equations = 0, and so switches off the amplification of the radiation from the electrons. This is unphysical, but is used more often than when qFieldEvolve and qElectronsEvolve =**.false.** . It can often be used to check for gain in the FEL in low gain situations, or to check whether amplification has occurred rather than coherent emission (CSE) effects due to *e.g.* fine structure in the beam. When =**.false.**, this will give the purely spontaneous emission (coherent or incoherent) from the electrons in the wiggler. If there is no appreciable difference between the qElectronFieldCoupling=true and false cases, then there has been negligible FEL amplification. [.true.]

qSeparateFiles

If each rank is writing its own file for the hdf5 files. If = true, then the files will need to be gathered together in post for analysis. Depending on the file system, this may be faster or slower for you. There are some post-processing python scripts to show how you might stitch the files together in post. [.false.]

qFocussing

Allows one to switch the strong focussing channel on or off. If =bf .true., the strong focusing is switched on with calues specified by sKBetaXSF and sKBetaYSF. If qFocussing=**.true.**, yet no focussing strength is specified (sKBetaXSF or sKBetaYSF), then strong focusing is effectively switched off in that dimension. Default = **.false.**. [.false.]

qDiffraction

Switches modelling of diffraction of the radiation field on or off. If =**.true.**, field diffraction is modelled. This can be useful to switch off only really for debugging purposes. Default =**.true.**. [.true.]

qFMesh_G

Whether fixing the field mesh in the transverse plane to contain the beam within the inner **iRedNodesX** $\times$ **iRedNodesY** nodes. [.true.]

sKBetaXSF, sKBetaYSF

Betatron wavenumber for energy γ_r of the applied strong focussing channel over the wiggler. Ignored if **qFocussing** is **.false.**. Both = 0 if not specified. Remember this will be applied on top of wiggler field, which has it's own natural focussing. [0.0, 0.0]

nodesPerLambdar

Number of nodes in the field mesh to use in the longitudinal dimension per reference resonant wavelength. The reference resonant wavelength is defined from the reference energy **sgamma_r**, the undulator period **lambda_w** and the base or reference undulator parameter **saw** in the usual way. [17]

sFModelLengthZ2

The full length of the field mesh in the longitudinal or temporal dimension. This must be long enough to contain the entire beam as it propagates through the undulator. It is auto-checked at the start of the simulation for the simple beam case only, and the check is only an estimate. [4.0]

sFModelLengthX, **sFModelLengthY**

The full length of the field mesh in the x and y (transverse) dimensions. This must be long enough to contain the entire beam as it propagates through the undulator. It is auto-checked at the start of the simulation for the simple beam case only, and the check is only an estimate. [1.0, 1.0]

sElectronThreshold

When using the simple beam generation, any macroparticles generated with weight below the threshold will be removed. The threshold is defined using this variable, which expresses the threshold as a percentage of the mean particle weight. [0.05]

lattFile

The name of the lattice file, if required. If not required, supply a blank string (enter **''**). ['' (blank string)]

nPeriods

Number of undulator periods in the single (non-lattice) undulator case. If a lattice file is suppiled, then this is ignored. [8]

stepsPerPeriod

Number of integration steps for the electron beam and field source term per undulator period. Should usually be $\gtrsim$ **nodesPerLambdar** for a beam with energy and wiggler parameter close to the reference parameters. [30]

qWriteA, qWriteZ2, qWritePperp, qWriteP2, qWriteX, qWriteY, qWriteZ

Logicals to switch on/off the writing of field, electron phase space coordinates ($\bar{z}_2$, $\bar{p}_\perp$, Γ, $\bar{x}$, $\bar{y}$), and propagation distance, respectively, in the sdds full data dumps. (Only used when **qsdds = .true.**)

beam_file

String - name of the electron beam input file. ['beam_file.in']

seed_file

String - name of the radiation seed input file. [‘’ (blank string)]

iNumNodesX, iNumNodesY

Number of nodes in the field mesh in the x and y transverse dimensions, respectively. [129,129]

iRedNodesX, iRedNodesY

If **qFixMesh** = **.true.**, then the mesh size in x and y will be altered in the initialization stage so that the inner **iRedNodesX** $\times$ **iRedNodesY** nodes in the transverse plane contain the beam. Only works for the simple beam input type. [20,20 (won't be fixed)]

nspinDX, nspinDY

Inner set of transverse nodes to be used in the MPI process updates, which should contain the full transverse beam - auto-calculated in Puffin if wrong. [20,20]

qsdds

Switch on/off ouput of particle and field mesh dumps in sdds format. [.false.]

qhdf5

Switch on/off output of particle and field mesh dumps in vizSchema compliant hdf5 format. [.true.]

qScaled

Logical turning on/off the input of scaled units into Puffin. If **.true.**, input is assumed to be in the scaled units. [.true.]

qFilter

Logical turning on/off the filtering of low frequency content during the siffraction step. The high-pass filter cut-off is defined by **sFiltFrac**. If true, the low frequency content is set = 0. If false, the low frequency content is not manipulated/diffracted. [.true.]

qInitWrLat

Logical. If true, there will be a full particle and field mesh dump at the end of each undulator module in the lattice. [.false.]

qUndEnds

Switch on/off proper modelling of the wiggler module ends. If true, then the first and last 2 periods of the wiggler will be tapered up/down with a $\sin^2$ function. [.false.]

q_Noise

If adding the statistical shot-noise to the beam in the simple beam case. [.true.]

sRedistLen, iRedistStp

Parallel tuning parameters, controlling the buffer length needed by each MPI process, and the number of steps per redistribution stage, respectively. [4.0, 60]

6.2 Beam Input File

As noted in section 2.4, the electron beam file can take 3 different forms.

References

[1] L.T. Campbell and B.W.J. McNeil, Physics of Plasmas **19**, 093119 (2012)

[2] L T Campbell, B.W.J. McNeil and S. Reiche, New J. Phys. **16** (2014) 103019

[3] E.T. Scharlemann, in High Gain, High Power FELs (edited by R. Bonifacio *et al*) (1989)

[4] B. W. J. McNeil, M. W. Poole, and G. R. M. Robb, Phys. Rev. ST Accel. Beams **6**, 070701 (2003)