Puffin Manual

September 19, 2016

Contents

1 Introduction **3**

2 Analytic Equations Solved by Puffin **7**

2.1 3D Magnetic Fields 9

2.2 Undulator Ends 10

2.3 Natural Undulator Focusing 13

2.4 Strong Beam Focusing 14

2.5 Auto beam-matching 15

2.6 To scale or not to scale? 15

2.7 Beam Initialization 16

2.7.1 Simple 17

2.7.2 Dist input 18

2.7.3 Beam input 18

2.8 Field Mesh 19

2.9 Fixing the Radiation Mesh 20

2.10 Lattice Input 22

3 Parameters **24**

3.1 Main Input File 24

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 v2.1.5.

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
- Undulator tuning of each module

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

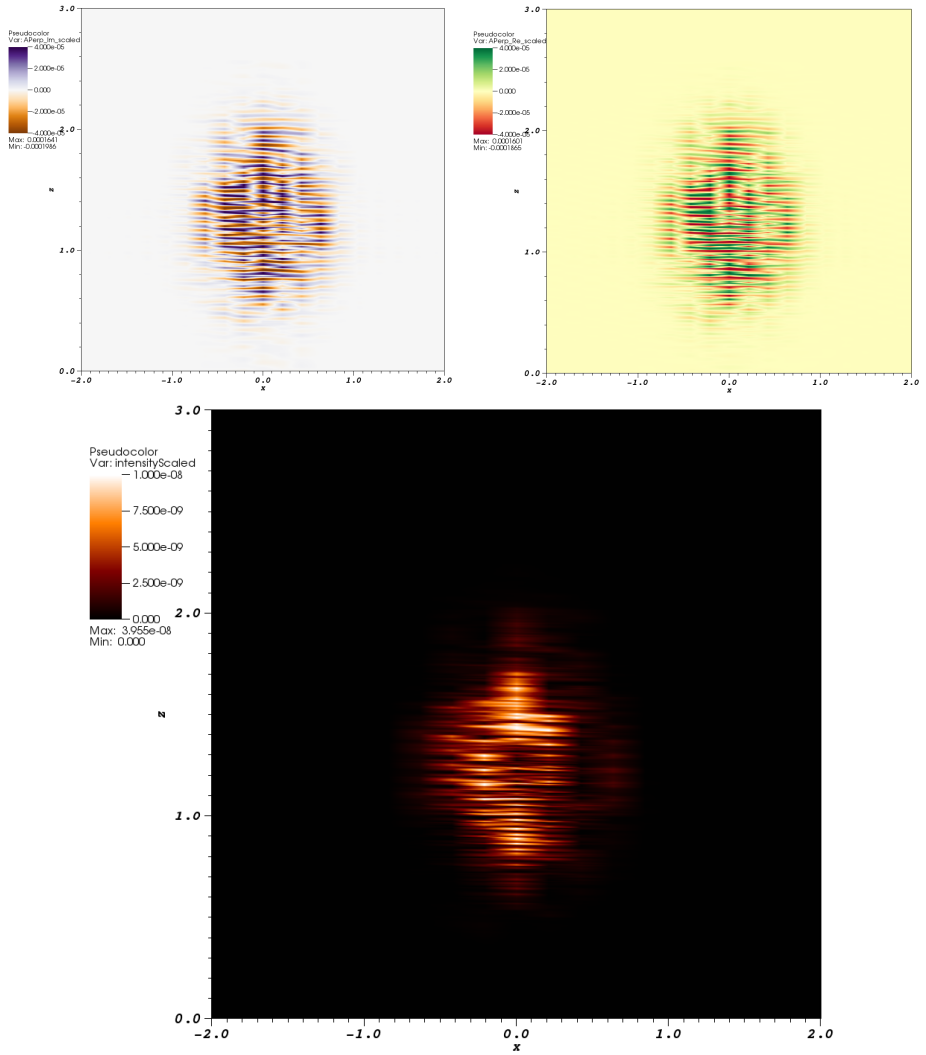

Figure 1: Radiated spontaneous field from Puffin in the first few undulator periods - x (top left) and y (top right) polarized fields, and instantaneous intensity (bottom), at $y = 0$. Radiation is propagating in the negative z direction (the vertical axis). One can see the noisy phase of the radiation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

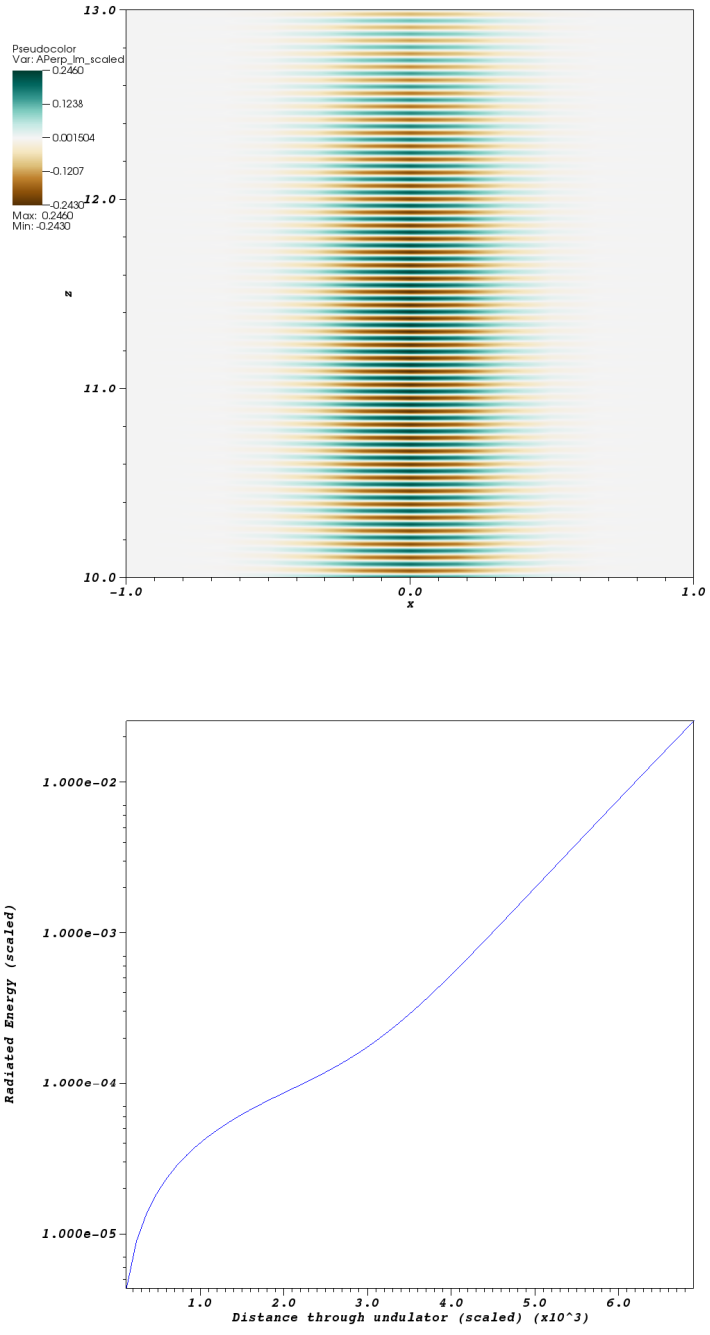

Figure 2: Radiated field from Puffin after amplification in the undulator - x polarized field (top). Radiated energy vs scaled distance through the undulator is on the bottom. Observe the noise from the previous figure has disappeared; the amplification process cause the electrons to align and radiate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

2 Analytic Equations Solved by Puffin

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

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

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

$$\frac{d\Gamma_j}{d\bar{z}}=-\rho\frac{(1+\eta p_{2j})}{\Gamma_j}(\bar{p}_{\perp j}A^*_{\perp j}+c.c.) \tag{3}$$

$$\frac{d\bar{z}_{2j}}{d\bar{z}}=p_{2j} \tag{4}$$

$$\frac{d\bar{x}_j}{d\bar{z}}=\frac{2\rho\kappa}{\sqrt{\eta}\Gamma_j}(1+\eta p_{2j})\Re(\bar{p}_{\perp j}) \tag{5}$$

$$\frac{d\bar{y}_j}{d\bar{z}}=-\frac{2\rho\kappa}{\sqrt{\eta}\Gamma_j}(1+\eta p_{2j})\Im(\bar{p}_{\perp j}). \tag{6}$$

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,

$$\eta = \frac{1-\beta_{zr}}{\beta_{zr}} = \frac{\lambda_r}{\lambda_u} = \frac{l_c}{l_g}, \tag{7}$$

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

$$\beta_{zr} = \sqrt{1 - \frac{1}{\gamma_0^2}\Big(1 + \bar{a}_u^2\Big)}, \tag{8}$$

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

$$p_{2j} = \frac{1}{\eta}\Big[\Big(1 - \frac{(1 + a_u^2|\bar{p}_{\perp j}|^2)}{\gamma_r^2\Gamma_j^2}\Big)^{-1/2} - 1\Big] \tag{9}$$

at each step. It is NOT output from Puffin.

Outputs from Puffin are the scaled radiation field $A_\perp$, electron phase space coords $\bar{x}, \bar{y}, \bar{z}_2, \bar{p}_\perp, \Gamma$, and scaled distance through the undulator $\bar{z}$. The normalized electron weights are in the file NormChiDataFile.dat (normalised to the peak *spatial* (3D) density).

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 $\alpha(\bar{z}) = \dfrac{B_0(\bar{z})}{B_0(\bar{z}=0)}$.

In the most basic form, the input requires 3 files - a 'main' input file for parameters, numerical integration, and description of the field mesh, a 'beam' input file descibing the electron beam, and a 'seed' input file describing the radiation seed. Optionally, a lattice file can be used to setup an undulator lattice, including quads, chicanes, and drifts.

2.1 3D Magnetic Fields

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 magnetic fields available for use in Puffin are the helical, plane-pole, and canted-pole undulator fields discussed in [3], which in the scaled notation here are:-

helical

$$b_x = \cos(\bar{z}/2\rho) \tag{10}$$

$$b_y = \sin(\bar{z}/2\rho) \tag{11}$$

$$b_z = \frac{\sqrt{\eta}}{2\rho}(-\bar{x}\sin(\bar{z}/(2\rho)) + \bar{y}\cos(\bar{z}/(2\rho))) \tag{12}$$

plane-pole

$$b_x = 0 \tag{13}$$

$$b_y = \cosh((\sqrt{\eta}/2\rho)\bar{y})\sin(\bar{z}/2\rho) \tag{14}$$

$$b_z = \sinh((\sqrt{\eta}/2\rho)\bar{y})\cos(\bar{z}/(2\rho) \tag{15}$$

canted-pole

$$b_x = \frac{\bar{k}_{\beta x}}{\bar{k}_{\beta y}}\sinh(\bar{k}_{\beta x}\bar{x})\sinh(\bar{k}_{\beta y}\bar{y})\sin(\bar{z}/2\rho) \tag{16}$$

$$b_y = \cosh(\bar{k}_{\beta x}\bar{x})\cosh(\bar{k}_{\beta y}\bar{y})\sin(\bar{z}/2\rho) \tag{17}$$

$$b_z = \frac{\sqrt{\eta}}{2\rho\bar{k}_{\beta x}}\cosh(\bar{k}_{\beta x}\bar{x})\sinh(\bar{k}_{\beta y}\bar{y})\cos(\bar{z}/2\rho) \tag{18}$$

variably polarized elliptical

$$b_x = u_x\cos(\bar{z}/2\rho) \tag{19}$$

$$b_y = u_y\sin(\bar{z}/2\rho) \tag{20}$$

$$b_z = \frac{\sqrt{\eta}}{2\rho}(u_x\bar{x}\sin(\bar{z}/(2\rho)) + u_y\bar{y}\cos(\bar{z}/(2\rho))) \tag{21}$$

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.

2.2 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.

The precise description of the undulator ends are as follows. Note the presence of corrective terms in addition to the main $\cos^2$ term, which ensure the beam oscillates close to the axis. (Note also the very small non-zero initial $z-$magnetic field in each case, which we find has no noticable deleterious effects in practice.)

Helical Front

$$b_x = \frac{1}{4}\sin(\bar{z}/16\rho)\cos(\bar{z}/16\rho)\sin(\bar{z}/2\rho) + \sin^2(\bar{z}/16\rho)\cos(\bar{z}/2\rho) \tag{22}$$

$$b_y = -\frac{1}{4}\sin(\bar{z}/16\rho)\cos(\bar{z}/16\rho)\cos(\bar{z}/2\rho) + \sin^2(\bar{z}/16\rho)\sin(\bar{z}/2\rho) \tag{23}$$

$$\begin{aligned} b_z = \frac{\sqrt{\eta}}{2\rho}\Big[\bar{x}\Big(&\frac{1}{32}\cos(\bar{z}/8\rho)\sin(\bar{z}/2\rho) + \frac{1}{4}\sin(\bar{z}/8\rho)\cos(\bar{z}/2\rho) \\ &- \sin^2(\bar{z}/16\rho)\sin(\bar{z}/2\rho)\Big) \\ \bar{y}\Big(&-\frac{1}{32}\cos(\bar{z}/8\rho)\cos(\bar{z}/2\rho) + \frac{1}{4}\sin(\bar{z}/8\rho)\sin(\bar{z}/2\rho) \\ &+ \sin^2(\bar{z}/16\rho)\cos(\bar{z}/2\rho)\Big)\Big] \end{aligned} \tag{24}$$

Helical Back

$$b_x = -\frac{1}{4}\cos(\bar{z}/16\rho)\sin(\bar{z}/16\rho)\sin(\bar{z}/2\rho) + \cos^2(\bar{z}/16\rho)\cos(\bar{z}/2\rho) \tag{25}$$

$$b_y = \frac{1}{4}\cos(\bar{z}/16\rho)\sin(\bar{z}/16\rho)\cos(\bar{z}/2\rho) + \cos^2(\bar{z}/16\rho)\sin(\bar{z}/2\rho) \tag{26}$$

$$\begin{aligned} b_z = \frac{\sqrt{\eta}}{2\rho}\Big[\bar{x}\Big(&-\frac{1}{32}\cos(\bar{z}/8\rho)\sin(\bar{z}/2\rho) - \frac{1}{4}\sin(\bar{z}/8\rho)\cos(\bar{z}/2\rho) \\ &- \cos^2(\bar{z}/16\rho)\sin(\bar{z}/2\rho)\Big) \\ \bar{y}\Big(&\frac{1}{32}\cos(\bar{z}/8\rho)\cos(\bar{z}/2\rho) - \frac{1}{4}\sin(\bar{z}/8\rho)\sin(\bar{z}/2\rho) \\ &+ \cos^2(\bar{z}/16\rho)\cos(\bar{z}/2\rho)\Big)\Big] \end{aligned} \tag{27}$$

Plane-Pole Front

$$b_x = 0, \tag{28}$$

$$b_y = \cosh(\sqrt{\eta}\bar{y}/2\rho)\Big[-\frac{1}{4}\sin(\bar{z}/16\rho)\cos(\bar{z}/16\rho)\cos(\bar{z}/2\rho) + \sin^2(\bar{z}/16\rho)\sin(\bar{z}/2\rho)\Big], \tag{29}$$

$$\begin{aligned} b_z = \sinh(\sqrt{\eta}\bar{y}/2\rho)\Big[&-\frac{1}{32}\cos(\bar{z}/8\rho)\cos(\bar{z}/2\rho) + \\ &\frac{1}{4}\sin(\bar{z}/8\rho)\sin(\bar{z}/2\rho) + \sin^2(\bar{z}/16\rho)\cos(\bar{z}/2\rho)\Big], \end{aligned} \tag{30}$$

Plane-Pole Back

$$b_x = 0, \tag{31}$$

$$b_y = \cosh(\sqrt{\eta}\bar{y}/2\rho)\Big[\frac{1}{4}\sin(\bar{z}/16\rho)\cos(\bar{z}/16\rho)\cos(\bar{z}/2\rho) + \cos^2(\bar{z}/16\rho)\sin(\bar{z}/2\rho)\Big], \tag{32}$$

$$\begin{aligned} b_z = \sinh(\sqrt{\eta}\bar{y}/2\rho)\Big[&\frac{1}{32}\cos(\bar{z}/8\rho)\cos(\bar{z}/2\rho) - \\ &\frac{1}{4}\sin(\bar{z}/8\rho)\sin(\bar{z}/2\rho) + \cos^2(\bar{z}/16\rho)\cos(\bar{z}/2\rho)\Big]. \end{aligned} \tag{33}$$

Curved-Pole Front

$$b_x = \frac{\bar{k}_{\beta x}}{\bar{k}_{\beta y}} \sinh(\bar{k}_{\beta x}\bar{x}) \sinh(\bar{k}_{\beta y}\bar{y}) \Big[-\frac{1}{8} \sin(\bar{z}/8\rho) \cos(\bar{z}/2\rho) + \sin^2(\bar{z}/16\rho) \sin(\bar{z}/2\rho) \Big], \tag{34}$$

$$b_y = \cosh(\bar{k}_{\beta x}\bar{x}) \cosh(\bar{k}_{\beta y}\bar{y}) \Big[-\frac{1}{8} \sin(\bar{z}/8\rho) \cos(\bar{z}/2\rho) + \sin^2(\bar{z}/16\rho) \sin(\bar{z}/2\rho) \Big], \tag{35}$$

$$b_z = \frac{\sqrt{\eta}}{2\rho \bar{k}_{\beta y}} \cosh(\bar{k}_{\beta x}\bar{x}) \sinh(\bar{k}_{\beta y}\bar{y}) \Big[-\frac{1}{32} \cos(\bar{z}/8\rho) \cos(\bar{z}/2\rho) + \frac{1}{4} \sin(\bar{z}/8\rho) \sin(\bar{z}/2\rho) + \sin^2(\bar{z}/16\rho) \cos(\bar{z}/2\rho) \Big]. \tag{36}$$

Curved-Pole Back

$$b_x = \frac{\bar{k}_{\beta x}}{\bar{k}_{\beta y}} \sinh(\bar{k}_{\beta x}\bar{x}) \sinh(\bar{k}_{\beta y}\bar{y}) \Big[\frac{1}{8} \sin(\bar{z}/8\rho) \cos(\bar{z}/2\rho) + \cos^2(\bar{z}/16\rho) \sin(\bar{z}/2\rho) \Big], \tag{37}$$

$$b_y = \cosh(\bar{k}_{\beta x}\bar{x}) \cosh(\bar{k}_{\beta y}\bar{y}) \Big[\frac{1}{8} \sin(\bar{z}/8\rho) \cos(\bar{z}/2\rho) + \cos^2(\bar{z}/16\rho) \sin(\bar{z}/2\rho) \Big], \tag{38}$$

$$b_z = \frac{\sqrt{\eta}}{2\rho \bar{k}_{\beta y}} \cosh(\bar{k}_{\beta x}\bar{x}) \sinh(\bar{k}_{\beta y}\bar{y}) \Big[\frac{1}{32} \cos(\bar{z}/8\rho) \cos(\bar{z}/2\rho) - \frac{1}{4} \sin(\bar{z}/8\rho) \sin(\bar{z}/2\rho) + \cos^2(\bar{z}/16\rho) \cos(\bar{z}/2\rho) \Big]. \tag{39}$$

2.3 Natural Undulator Focusing

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

$$\bar{k}_{\beta nx} = \bar{k}_{\beta ny} = \frac{a_w}{2\sqrt{2}\rho\gamma_0}, \tag{40}$$

with γ_0 being the average beam energy (and not necessarily = γ_r)

In the planar case,

$$\bar{k}_{\beta ny} = \frac{a_w}{2\sqrt{2}\rho\gamma_0}. \tag{41}$$

In the canted pole case,

$$\bar{k}_{\beta nx,y} = \frac{a_w \bar{k}_{x,y}}{\sqrt{2\eta}\gamma_0}, \tag{42}$$

where $\bar{k}_{x,y}$ describe the hyperbolic variation in the transverse directions (see eqns (16 - 18)), and must obey

$$\bar{k}_x^2 + \bar{k}_y^2 = \frac{\eta}{4\rho^2} \tag{43}$$

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

$$\bar{k}_{\beta nx} = \bar{k}_{\beta ny} = \frac{a_w}{4\rho\gamma_0}. \tag{44}$$

2.4 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

$$b_x = \sqrt{\eta}\frac{\bar{k}_{\beta y}^2}{\kappa}\bar{y}_j, \tag{45}$$

$$b_y = -\sqrt{\eta}\frac{\bar{k}_{\beta x}^2}{\kappa}\bar{x}_j \tag{46}$$

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 will be added soon.

2.5 Auto beam-matching

The beam, when specified by the 'simple' method (see below) may be matched to the focusing channel automatically with the flag **qMatched_A** in the beam file - the option can be set for each beam. In the scaled notation,

$$\bar{\sigma}_{x,y} = \sqrt{\frac{\rho\bar{\epsilon}_{x,y}}{\bar{k}_{\beta x,y}}} \tag{47}$$

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

$$\bar{\sigma}_{px,py} = \frac{\sqrt{\eta}}{2\kappa}\Big\langle\frac{\Gamma}{1+\eta p_2}\Big\rangle\frac{\bar{\epsilon}_{x,y}}{\bar{\sigma}_{x,y}}. \tag{48}$$

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.

2.6 To scale or not to scale?

The main algorithm in Puffin utilizes scaled variables 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

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

Unscaled	units	unscaled equivalent
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)	$\bar{z}_2$

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. The ρ parameter is dependent on the peak electron beam number density, amongst other things.

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 do this. If = **.true.**, then Puffin expects scaled variables are being used. If SI units are used, then the input parameters will be scaled according to the scaling values specified in the main input file.

Note that they are only *scaling* parameters. The FEL parameter as input does not have to correspond to the beam being input for the results to be correct. The scaled parameters will behave 'nicely' if it does, but it is not necessary. When the result is unscaled, the result will still be correct.

2.7 Beam Initialization

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

2.7.1 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 $\bar{z}_2$. 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] (see figure REF). This method, like other noise algorithms, 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. 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.

2.7.2 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 for each temporal slice. This information is used to generate the electron beam in Puffin.

2.7.3 Beam input

This method allows one to read in the 6D scaled particle coordinates from a text file. The beam is generated externally. (Arrangement?) The user is

then responsible for the noise in the pulse - it is not added by Puffin in this method.

2.8 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 3. 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 3.

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 4. 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.

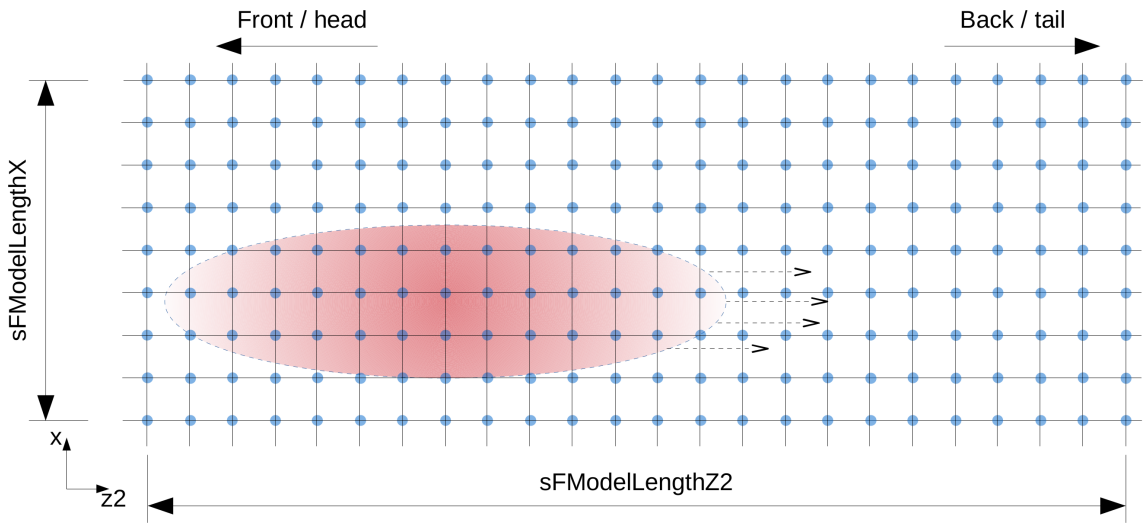

Figure 3: Showing the longitudinal setup of the radiation mesh. The beam is indicated in red. The scaled $\bar{z}_2$ coordinate is defined so that the front of the radiation and beam is to the left. This is the constant radiatioon frame, so the beam slips backwards through the field from left to right. 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.

2.9 Fixing the Radiation Mesh

One may use the variables **iRedNodesX, iRedNodesY** to fix the mesh around the beam - see figure 4. 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, sFModelLengthX** inputs will be ignored.

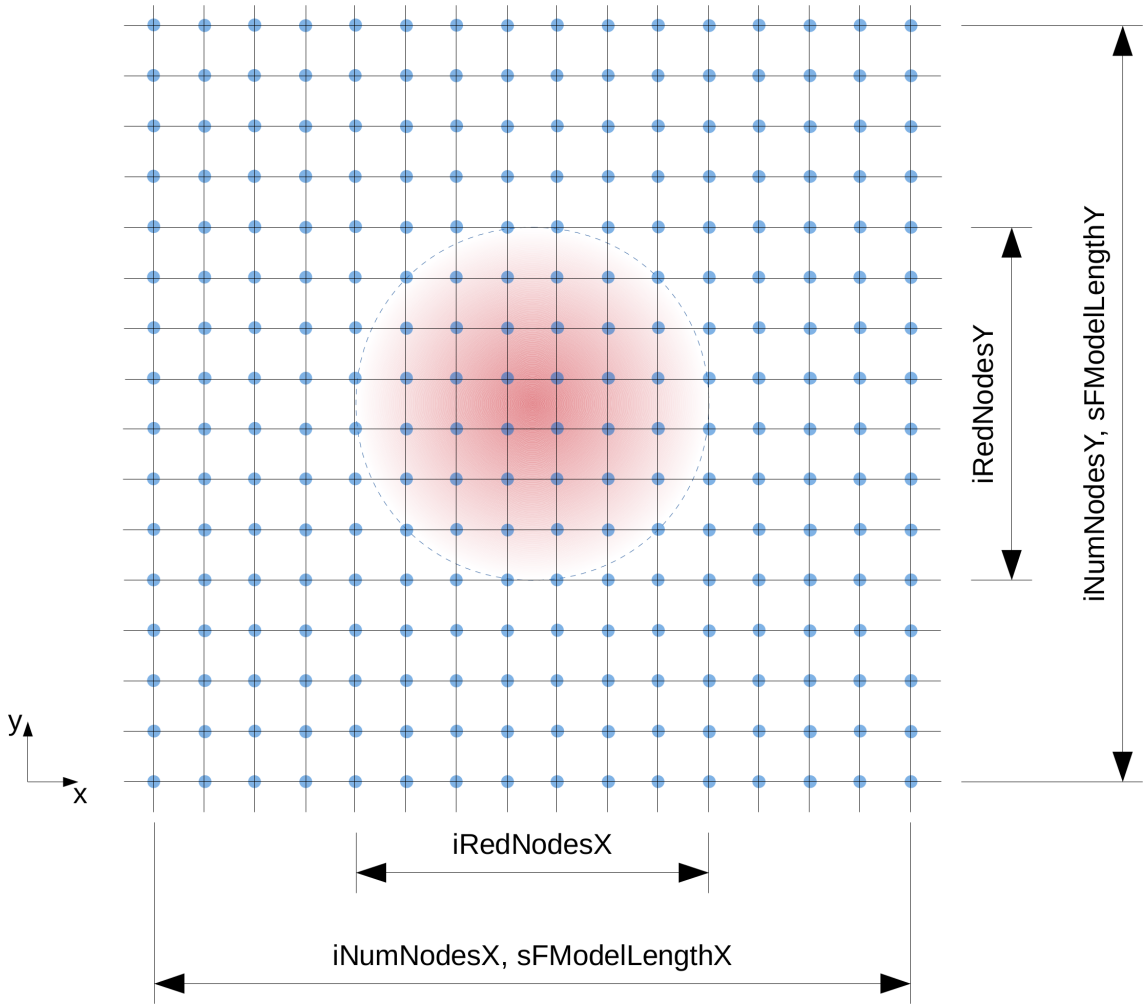

Figure 4: Transverse mesh to model the radiation field - the beam is indicated in red. The mesh may be 'matched' to the mesh initially by specifying the iRedNodes parameters - these specify an inner set of nodes such that the mesh will be setup so that the beam is contained within this number of nodes. This is only an initial condition, but can aid in setting up the mesh.

2.10 Lattice Input

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.

Undulator - UN

Quadrupole - QU

Chicane - CH

Drift - DR

Modulation - MO

For the undulator, we must specify type, Nw, alpha, taper, ux, uy, qends, qfocus, kbx, kby, and steps per period.

For the quad, we must specify k and L.

For the chicane, one must specify physical length, slippage length, and dispersion enhancement (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 is the R56 of the device - using γ_r in the main input as the mean *gamma* for the chicane.

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.

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

For the modulation section, one may add an energy modulation to the beam specified by k_M and $\Delta\gamma$.

3 Parameters

Below is an exhaustive list of parameters, for each filetype. Some options are only really useful for investigating runitme errors or strange behaviour. I general, the parameter list for each file is sorted from most to least useful or relevant to the average user.

3.1 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 using the supplied scripts (NOT SUPPLIED YET!!). If it 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 see how efficiently the system is lasing w.r.t its ideal case.

saw

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

sgamma_r

γ_0, the reference (usually the mean) beam energy.

lambda_w

λ_u, the undulator, or wiggler, period.

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.

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, $u_x = u_y = 1$.

taper

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

iWriteNthSteps

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

iWriteIntNthSteps

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

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).

sDiffFrac

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

sBeta

Absorption coefficient for the absorbing boundaries on the transverse grid.

qOneD

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

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**.

qElectronsEvolve

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

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.**.

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.**.

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.

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.

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.

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.

lattFile

The name of the lattice file, if required. If not required, supply a blank string (enter ‘’).

nPeriods

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

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.

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)