Cylindrical EPOCH

Stuart Morris

December 6, 2023

Abstract

A report on the cylindrical PIC implementation in EPOCH, featuring theory, input deck syntax and benchmarks. For code users, the introduction should be enough to understand the code, and the Test decks section at the end provides basic examples to get simulations up and running. For developers, the rest of the document discusses the exact conversion process from EPOCH2D into cylindrical-EPOCH.

1 Introduction

EPOCH is the **E**xtendible **P**article-in-cell **O**pen **C**ollaboration code (with a silent **H**) maintained at the University of Warwick. The code framework was originally based on the Plasma Simulation Code (PSC) [1], in which weighted macro-particles move under the influence of fields calculated on a 1D, 2D or 3D Cartesian grid. It is also possible to design a particle-in-cell (PIC) code to use fields evaluated in a cylindrical (x, r, θ) co-ordinate system, as shown by Lifschitz *et al* [2].

In certain circumstances, cylindrical PIC codes can have advantages over the Cartesian equivalents. Field points may be uniformly spaced on a 2D (x, r) grid, but these cells may describe a θ range from 0 to 2π, such that the 2D grid describes a full 3D space. A visualisation of this grid and the 3D volume described by a cell is present in Figure 1. Simulated fields may include a θ variation by combining complex field modes evaluated on the grid $\hat{\boldsymbol{F}}^m(x, r)$, which would allow a generic field $\boldsymbol{F}(x, r, \theta)$ to be written as

$$\boldsymbol{F}(x, r, \theta) = \mathbb{R}\left\{\sum_{m=0} \hat{\boldsymbol{F}}^m(x, r) e^{-im\theta}\right\} \tag{1}$$

$$= \mathbb{R}\left\{\hat{\boldsymbol{F}}^0(x, r)\right\} + \mathbb{R}\left\{\hat{\boldsymbol{F}}^1(x, r)\right\}\cos(\theta) + \mathbb{I}\left\{\hat{\boldsymbol{F}}^1(x, r)\right\}\sin(\theta) + \mathbb{R}\left\{\hat{\boldsymbol{F}}^2(x, r)\right\}\cos(2\theta) + ... \tag{2}$$

Hence, a cylindrical PIC code stores and updates field modes on the grid, and a sum of modes is performed to evaluate an EM field or current density at a particular θ. In a general case, this new treatment would provide no benefit as we have swapped one form of complexity for another - going from 2D to 3D on a Cartesian grid requires many new particles, and this quasi-3D cylindrical grid would require many new modes. However, as demonstrated by Lifschitz *et al* [2], some problems with cylindrical symmetry can be described using only a few modes, like laser-wakefield acceleration. In these cases, a cylindrical PIC code could provide 3D results in simulations only slightly more computationally expensive than traditional Cartesian 2D codes.

A new version of EPOCH has been created for running in a cylindrical co-ordinate system, following the methods outlined by Lifschitz *et al* [2]. Where implementation details were unclear, we have adapted algorithms from the SMILEI PIC code [3], or derived our own. In this report, we describe the basic structures of the code, derive the theory behind the cylindrical-PIC method used, and provide benchmark simulations.

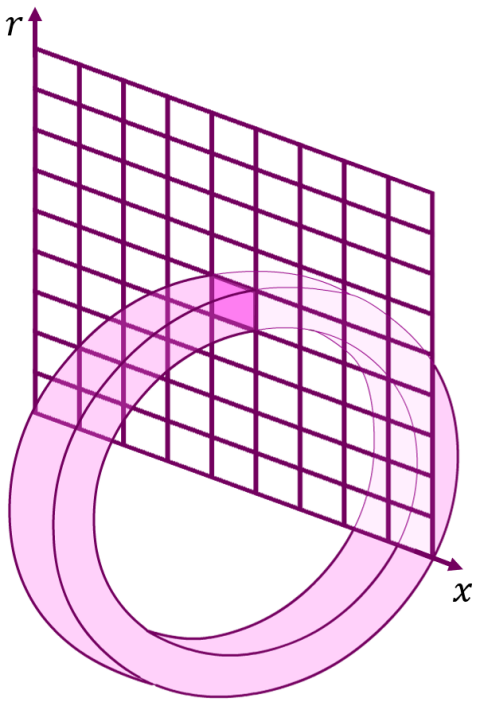

Figure 1: A schematic diagram showing the cylindrical EPOCH (x, r) grid, and the 3D ring of volume represented by a given, highlighted cell.

2 Basic structures

The cylindrical EPOCH code was adapted from EPOCH2D, and uses many existing features for I/O and domain decomposition. The main differences between cylindrical EPOCH and EPOCH2D are in the field solver and the particle push.

On a given MPI rank in EPOCH2D, the local EM fields were stored in the 2D real arrays `ex`, `ey`, `ez`, `bx`, `by`, and `bz`, which had dimensions `(1-ng:nx+ng,1-ng:ny+ng)`. Here, `nx` and `ny` described the number of cells in the x and y directions for the current MPI rank, and `ng` described the number of ghost cells which surrounded the rank. In cylindrical EPOCH, we instead track the field modes in the 3D complex arrays `exm`, `erm`, `etm`, `bxm`, `brm` and `btm`, with dimensions `(1-ng:nx+ng,1-ng:ny+ng,0:n_mode-1)` where `n_mode` is the number of modes present in the simulation. As SDF does not currently have a complex variable type, the real and imaginary components of these fields are output to different arrays when dumping to file. Also, due to constraints imposed by the methods of Lifschitz *et al* [2], we require $\hat{B}_r^m$, $\hat{E}_x^m$ and $\hat{E}_\theta^m$ to be evaluated on the $r = 0$ axis. Consequently, the grid-stagger used in cylindrical EPOCH is different to that in EPOCH2D, as seen in Figure 2.

Despite using cylindrical components for $\boldsymbol{E}$, $\boldsymbol{B}$ and $\boldsymbol{J}$, we have chosen to use (x, y, z) and (p_x, p_y, p_z) for the positions and momenta of macro-particles. The equivalence between these two co-ordinate systems is demonstrated in Figure 3. A quick conversion between co-ordinate system positions can therefore be achieved using

$$y + iz = re^{i\theta} \tag{3}$$

and vectors can transform according to

$$F_y = F_r \cos\theta - F_\theta \sin\theta \tag{4}$$
$$F_z = F_r \sin\theta + F_\theta \cos\theta \tag{5}$$

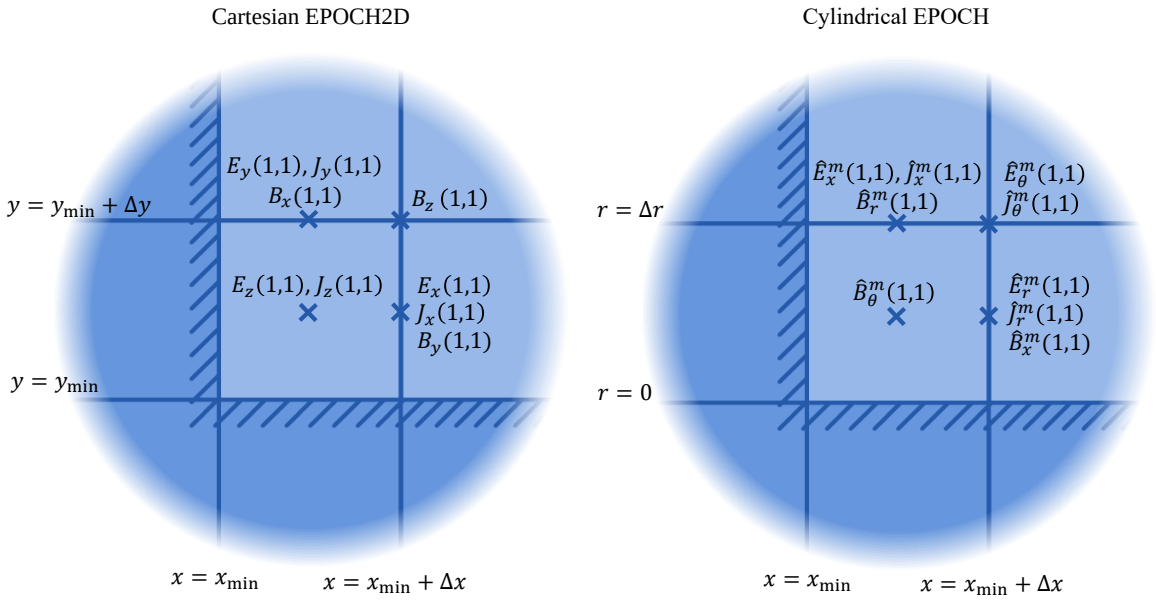

Figure 2: The positions of field evaluation points on the EPOCH2D and cylindrical EPOCH grids. The central cell describes the low-x, low-y corner of the simulation window for EPOCH2D, and the low-x, $r = 0$ corner for cylindrical EPOCH. Ghost cells are coloured by darker cells. The evaluation points included on these figures describe points with (x, y) or (x, r) indices $(1, 1)$, as seen by the MPI rank which contains this region of the simulation window.

with reverse transform

$$F_r = F_y \cos\theta + F_z \sin\theta \tag{6}$$
$$F_\theta = -F_y \sin\theta + F_z \cos\theta \tag{7}$$

For computational efficiency, is often convenient to express these trigonometric terms as

$$\cos\theta = \mathbb{R}\left\{e^{i\theta}\right\} \tag{8}$$
$$\sin\theta = \mathbb{I}\left\{e^{i\theta}\right\} \tag{9}$$

When macro-particle shapes on the (x, r) grid dip into ghost cells below $r = 0$, this is equivalent to having a shape extending away from $r = 0$ with $\theta \to \theta + \pi$, as shown in Figure 4. This has implications which are addressed in later sections.

3 Macro-particle loading

In cylindrical EPOCH, the volume, V, of a cell with central radius r_0 is

$$V = 2\pi r_0 \Delta x \Delta r \tag{10}$$

which means that for the first two cells with $r_0 = \Delta r/2$ and $r_0 = 3\Delta r/2$, the volume difference is a factor of 3. Hence, if we chose macro-particle weights based on cell-volumes, there would be sharp weight discontinuities near the $r = 0$ axis, which would lead to noise. Instead, we have modified the macro-particle loading behaviour for cases both with and without variable macro-particle weights.

When the code is compiled without `-DPER_SPECIES_WEIGHT`, we calculate the number of macro-particles per cell N_{ppc}, and randomly sample the positions of N_{ppc} macro-particles within each cell from uniform distributions between the x and r cell limits. Then for each macro-particle, we interpolate

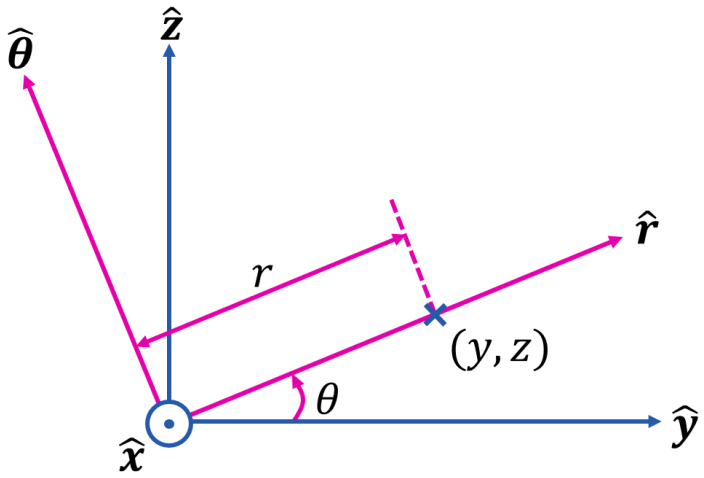

Figure 3: Unit vectors in Cartesian and cylindrical space, when evaluated about the point marked (y, z).

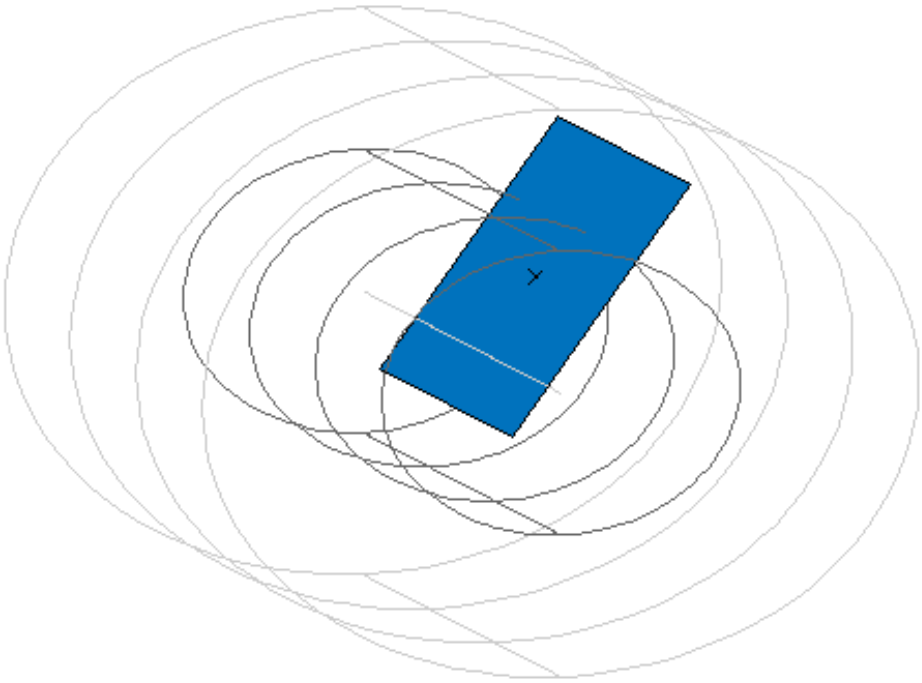

Figure 4: Particle shape on a cylindrical grid when the radial position of the particle satisfies $r < \Delta r$, where Δr is the radial cell-size. The macro-particle position is marked with an X. Particle shape spans over x and r, but has constant θ. The first low-r ghost cell on the cylindrical grid corresponds to the opposite side of $r = 0$ in the first physical cell.

the target number densities in local cells over the macro-particle shape to get n_p, and set the weight to $W = n_p V_p / N_{ppc}$, where V_p is the volume of the macro-particle

$$V_p = 2\pi r_p \Delta x \Delta r \tag{11}$$

and r_p is the radial macro-particle position.

When using `-DPER_SPECIES_WEIGHT`, the code multiplies the cell volumes by their number densities to calculate the total number of real particles to model N_{total}, then equally shares that weight between all macro-particles in the species. A number of macro-particles are then loaded into each cell based on the ratio of real particles in that cell to the total number of real particles.

$$r_p = \sqrt{(r_0 - \Delta r/2)^2 + x_r \left((r_0 + \Delta r/2)^2 - (r_0 - \Delta r/2)^2\right)} \tag{12}$$

where x_r is a uniformly distributed random number between 0 and 1. Using these methods, EPOCH is able to load a uniform distribution of macro-particles, which remains approximately uniform even when macro-particle shapes are considered.

4 Field Solver

Cylindrical EPOCH models complex field modes instead of the full EM fields, so the EPOCH2D field-solver must be re-derived to update these new fields. This will be done following the approach of Lifschitz *et al* [2] and Smilei [3]. As with the Cartesian case, we begin with the time-dependent Maxwell equations

$$\frac{\partial \boldsymbol{B}}{\partial t} = -\nabla \times \boldsymbol{E} \tag{13}$$

$$\frac{\partial \boldsymbol{E}}{\partial t} = c^2 \nabla \times \boldsymbol{B} - \frac{1}{\epsilon_0}\boldsymbol{J} \tag{14}$$

where in cylindrical co-ordinates, we have

$$\nabla \times \boldsymbol{F} = \frac{1}{r}\left(\frac{\partial}{\partial r}(rF_\theta) - \frac{\partial}{\partial \theta}F_r\right)\hat{\boldsymbol{x}} + \left(\frac{1}{r}\frac{\partial}{\partial \theta}F_x - \frac{\partial}{\partial x}F_\theta\right)\hat{\boldsymbol{r}} + \left(\frac{\partial}{\partial x}F_r - \frac{\partial}{\partial r}F_x\right)\hat{\boldsymbol{\theta}} \tag{15}$$

By substituting the field mode definition (1) into Maxwell's equations (13) and (14), grouping terms of the same mode together, and using

$$\frac{\partial}{\partial \theta}\left(\hat{\boldsymbol{F}}^m(x,r)e^{-im\theta}\right) = -im\hat{\boldsymbol{F}}^m(x,r)e^{-im\theta} \tag{16}$$

we obtain SI equivalents of the field mode equations listed by Lifschitz *et al* in their equations 6-11 [2], given here as

$$\frac{\partial}{\partial t}\hat{B}_x^m = -\frac{1}{r}\frac{\partial}{\partial r}\left(r\hat{E}_\theta^m\right) - \frac{im}{r}\hat{E}_r^m \tag{17}$$

$$\frac{\partial}{\partial t}\hat{B}_r^m = \frac{\partial}{\partial x}\hat{E}_\theta^m + \frac{im}{r}\hat{E}_x^m \tag{18}$$

$$\frac{\partial}{\partial t}\hat{B}_\theta^m = \frac{\partial}{\partial r}\hat{E}_x^m - \frac{\partial}{\partial x}\hat{E}_r^m \tag{19}$$

$$\frac{\partial}{\partial t}\hat{E}_x^m = \frac{c^2}{r}\frac{\partial}{\partial r}\left(r\hat{B}_\theta^m\right) + \frac{imc^2}{r}\hat{B}_r^m - \frac{1}{\epsilon_0}\hat{J}_x^m \tag{20}$$

$$\frac{\partial}{\partial t}\hat{E}_r^m = -\frac{imc^2}{r}\hat{B}_x^m - c^2\frac{\partial}{\partial x}\hat{B}_\theta^m - \frac{1}{\epsilon_0}\hat{J}_r^m \tag{21}$$

$$\frac{\partial}{\partial t}\hat{E}_\theta^m = c^2\frac{\partial}{\partial x}\hat{B}_r^m - c^2\frac{\partial}{\partial r}\hat{B}_x^m - \frac{1}{\epsilon_0}\hat{J}_\theta^m \tag{22}$$

As in the Cartesian case, the evaluation times and positions of all terms in (17)-(22) are staggered to allow for convenient calculation. For example, (18) may be discretised according to

$$\left(\hat{B}_r^m\right)_{(i_x,i_r)}^{t=t_0+\Delta t} = \left(\hat{B}_r^m\right)_{(i_x,i_r)}^{t=t_0} + \Delta t \left(\frac{\left(\hat{E}_\theta^m\right)_{(i_x,i_r)}^{t=t_0+\Delta t/2} - \left(\hat{E}_\theta^m\right)_{(i_x-1,i_r)}^{t=t_0+\Delta t/2}}{\Delta x} + \frac{im}{\left(r_0 + (i_r - \frac{1}{2})\Delta r\right)} \left(\hat{E}_x^m\right)_{(i_x,i_r)}^{t=t_0+\Delta t/2} \right) \tag{23}$$

where r_0 denotes the cell-centred radial position of the cell with index $i_r = 1$ on the local MPI rank, and Δx, Δr describing the cell size in the x and r directions respectively.

4.1 Field boundary conditions at $r = 0$

From Figure 2, we see that some evaluation points for $\hat{E}_x^m$, $\hat{E}_\theta^m$ and $\hat{B}_r^m$ will lie on the $r = 0$ axis, which generates $1/0$ errors for (18) and (20), and would require a $\hat{B}_x^m$ value at an unphysical $r < 0$ for (22). To address this, Lifschitz *et al* [2] discuss a few methods.

Firstly, we forbid multi-value fields: a generic Cartesian field component F_i can only have a single value at any point in space. A point at cylindrical position $(r = 0, \theta)$ refers to the same point in space for all θ, and so Cartesian field components must have no θ dependence at $r = 0$, summarised by Lifschitz *et al* equation 19 [2], or here as

$$\frac{\partial}{\partial\theta} F_i \bigg|_{r=0} \tag{24}$$

for $i = x, y, z$. Thus, $F_i(x, r = 0, \theta) = F_i(x, r = 0)$, and so $\hat{F}_i^{m>0} = 0$ according to (1). As a result of this, $\hat{E}_x^m = 0$ for $m > 0$ on $r = 0$, so we only need to find a special case for $\hat{E}_x^0$ in (20). The same logic may be applied to cylindrical field components if we consider a transverse Cartesian coordinate like F_y. By substituting the F_y cylindrical transformation (4) into (24), we get an $r = 0$ condition of

$$\left(\frac{\partial}{\partial\theta} F_r - F_\theta\right) \cos\theta - \left(\frac{\partial}{\partial\theta} F_\theta + F_r\right) \sin\theta = 0 \tag{25}$$

which requires both

$$\frac{\partial}{\partial\theta} F_r - F_\theta = 0 \tag{26}$$

$$\frac{\partial}{\partial\theta} F_\theta + F_r = 0 \tag{27}$$

or, when substituting field components for field modes as in (1), evaluating the field mode gradients as in (16) and grouping terms of equal $e^{im\theta}$, we get

$$\hat{F}_r^m = \frac{i}{m} \hat{F}_\theta^m \tag{28}$$

$$\hat{F}_r^m = im \hat{F}_\theta^m \tag{29}$$

These can only both be true for $m = 1$, hence generic field mode components $\hat{F}_r^m = \hat{F}_\theta^m = 0$ on the $r = 0$ axis for $m \neq 1$, and

$$\hat{F}_r^1(x, r = 0) = i\hat{F}_\theta^1(x, r = 0) \tag{30}$$

4.1.1 $\hat{E}_x^0$ update on $r = 0$

For the $\hat{E}_x^0(x, r = 0)$ update in (20), let us rewrite the $\hat{B}_\theta^m$ term as a limit

$$\lim_{r\to 0} \left(\frac{c^2}{r} \frac{\partial}{\partial r} \left(r\hat{B}_\theta^0\right)\right) = c^2 \left(\frac{\partial}{\partial r} \hat{B}_\theta^0\right)\bigg|_{r=0} + c^2 \lim_{r\to 0} \left(\frac{\hat{B}_\theta^0(x, r) - \hat{B}_\theta^0(x, r = 0)}{r - 0}\right) \tag{31}$$

$$= 2c^2 \left(\frac{\partial}{\partial r} \hat{B}_\theta^0\right)\bigg|_{r=0} \tag{32}$$

where we have used the fact $\hat{B}^0_\theta(x, r = 0) = 0$ from (24) to add it onto the right hand side of (31). To proceed, consider the Taylor expansion

$$\hat{B}^0_\theta\left(x, r = 0 + \frac{\Delta r}{2}\right) \approx \hat{B}^0_\theta(x, r = 0) + \frac{\Delta r}{2}\left(\frac{\partial}{\partial r}\hat{B}^0_\theta\right)\Big|_{r=0} \tag{33}$$

Therefore, the limit in (31) can be written as

$$\lim_{r\to 0}\left(\frac{c^2}{r}\frac{\partial}{\partial r}\left(r\hat{B}^0_\theta\right)\right) = \frac{4c^2}{\Delta r}\hat{B}^0_\theta\left(x, r = \frac{\Delta r}{2}\right) \tag{34}$$

which is convenient as $r = \Delta r/2$ coincides with a staggered $\hat{B}^0_\theta$ evaluation point, as can be seen in Figure 2. The $\hat{B}^0_r$ term in (20) can also be evaluated using limits and derivatives, but this can be skipped due to the m factor vanishing at $m = 0$. This brings us to the $\hat{E}^0_x(x, r = 0)$ treatment of Lifschitz *et al* in their equation 20 [2], or here as

$$\left(\frac{\partial}{\partial t}\hat{E}^0_x\right)\Big|_{r=0} = \frac{4c^2}{\Delta r}\hat{B}^0_\theta(x, r = \frac{\Delta r}{2}) - \frac{1}{\epsilon_0}\hat{J}^0_x \tag{35}$$

4.1.2 $\hat{B}^1_r$ update on $r = 0$

We may proceed with the $\hat{B}^1_r(x, r = 0)$ update in (18) using a similar treatment as in Section 4.1.1 to address the $1/0$ term. By exploiting limits and the fact $\hat{E}^1_x(x, r = 0) = 0$, we may write

$$\frac{1}{r}\hat{E}^1_x(x, r = 0) = \lim_{r\to 0}\left(\frac{\hat{E}^1_x(x, r) - \hat{E}^1_x(x, r = 0)}{r - 0}\right) = \left(\frac{\partial}{\partial r}\hat{E}^1_x\right)\Big|_{r=0} \tag{36}$$

and using the expansion

$$\hat{E}^1_x\left(x, r = 0 + \Delta r\right) \approx \hat{E}^1_x(x, r = 0) + \Delta r\left(\frac{\partial}{\partial r}\hat{E}^1_x\right)\Big|_{r=0} \tag{37}$$

we obtain

$$\left(\frac{\partial}{\partial t}\hat{B}^1_r\right)\Big|_{r=0} = \frac{\partial}{\partial x}\hat{E}^m_\theta + \frac{i}{\Delta r}\hat{E}^1_x(x, r = \Delta r) \tag{38}$$

This expression is equivalent to equation 22 in Lifschitz *et al* [2].

4.1.3 $\hat{E}^1_\theta$ update on $r = 0$

Finally, for the $\hat{E}^1_\theta(x, r = 0)$ update in (22), it is first useful to derive a second-order one-sided expression for a generic gradient evaluated at r_0 using Taylor expansions

$$F\left(r_0 + \frac{\Delta r}{2}\right) \approx F(r_0) + \frac{\Delta r}{2}\left(\frac{\partial}{\partial r}F\right)\Big|_{r=r_0} + \frac{\Delta r^2}{8}\left(\frac{\partial^2}{\partial r^2}F\right)\Big|_{r=r_0} \tag{39}$$

$$F\left(r_0 + \frac{3\Delta r}{2}\right) \approx F(r_0) + \frac{3\Delta r}{2}\left(\frac{\partial}{\partial r}F\right)\Big|_{r=r_0} + \frac{9\Delta r^2}{8}\left(\frac{\partial^2}{\partial r^2}F\right)\Big|_{r=r_0} \tag{40}$$

which combine to give

$$\left(\frac{\partial}{\partial r}F\right)\Big|_{r=r_0} = \frac{9F\left(r_0 + \frac{\Delta r}{2}\right) - F\left(r_0 + \frac{3\Delta r}{2}\right) - 8F\left(r_0\right)}{3\Delta r} \tag{41}$$

Next, let us seek an equation for $\hat{E}^1_\theta(x, r = 0)$ using Gauss' law

$$\nabla \cdot \boldsymbol{E} = \frac{\rho}{\epsilon_0} \tag{42}$$

which when evaluating the scalar product in cylindrical co-ordinates, replacing $\boldsymbol{E}$ and ρ with their mode expansions (1), and considering the $m = 1$ mode of interest, we get

$$\frac{\partial}{\partial r}\hat{E}_r^1 + \frac{1}{r}\hat{E}_r^1 - \frac{i}{r}\hat{E}_\theta^1 + \frac{\partial}{\partial x}\hat{E}_x^1 = \hat{\rho}^1 \tag{43}$$

On $r = 0$, we recall that $\hat{E}_x^1$ must be zero due to the forbidden multi-value condition (24), which also applies to the scalar quantity ρ as this can only have one value at $r = 0$. By discarding these zero terms, and using the $r = 0$ relation between $\hat{E}_r^1$ and $\hat{E}_\theta^1$ in (30) to cancel the $1/r$ terms, we find

$$\left(\frac{\partial}{\partial r}\hat{E}_r^1\right)\bigg|_{r=0} = 0 \tag{44}$$

which we may use the Taylor-derived expression (41) to write as

$$8\hat{E}_r^1(x, r = 0) = 9\hat{E}_r^1\left(x, r = \frac{\Delta r}{2}\right) - \hat{E}_r^1\left(x, r = \frac{3\Delta r}{2}\right) \tag{45}$$

Due to the cell stagger in Figure 2, we see that $r = 0$ is not an evaluation point for $\hat{E}_r^1$, so we may once again use the $r = 0$ equivalence between $\hat{E}_r^m$ and $\hat{E}_\theta^m$ in (30) to write

$$\hat{E}_\theta^1(x, r = 0) = -\frac{i}{8}\left(9\hat{E}_r^1\left(x, r = \frac{\Delta r}{2}\right) - \hat{E}_r^1\left(x, r = \frac{3\Delta r}{2}\right)\right) \tag{46}$$

This equation is not present in Lifschitz *et al* [2], but may instead be found in the online SMILEI [3] documentation. For $m > 1$, we require $\hat{E}_\theta^{m>1}(x, r = 0) = 0$, which we can force by setting

$$\hat{E}_r^{m>1}\left(x, r = \frac{\Delta r}{2}\right) = \frac{1}{9}\hat{E}_r^{m>1}\left(x, r = \frac{3\Delta r}{2}\right) \tag{47}$$

4.1.4 Field boundary conditions in $r < 0$ ghost cells

On the $r = 0$ boundary, the field mode values are set using quantities found within the simulation window, and so field points in the non-physical ghost-cells with $r < 0$ are not used by the field-solver routines. However, according to Figure 4, the macro-particle shape may still pass into these ghost-cells. Hence, we should set $r < 0$ ghost cell field modes to values found at $\theta \to \theta + \pi$, for the use of subroutines which interpolate fields over macro-particle shapes.

When implementing these physical reflections for the ghost cells, we must consider the changing direction of the fields as we go from $\theta \to \theta + \pi$, as illustrated in Figure 5. The conversion between field modes to fields given in (1) includes an $e^{-im\theta}$ factor, which transforms according to

$$e^{-im\theta} \to e^{-im(\theta+\pi)} = \begin{cases} e^{-im\theta} & \text{Even } m \\ -e^{-im\theta} & \text{Odd } m \end{cases} \tag{48}$$

Hence the field modes below $r = 0$ could add to or subtract from the total field interpolated over the macro-particle shape, depending on the field component and the mode number. An example of a physical reflection in the code would be $\hat{B}_r^1(i_x, -i_r) = \hat{B}_r^1(i_x, i_r)$, as Figure 5 shows $\hat{B}_r^1$ points in the same direction above and below the $r = 0$ line.

Most ghost cells with $r < 0$ will default to the physical reflection condition, but there are a few special cases where different rules are used to satisfy conditions within the simulation window. As $\hat{B}_x^{m>0}(x, r = 0)$ must be 0 due to the forbidden multi-value condition, we choose

$$\hat{B}_x^{m>0}(i_x, 0) = -\hat{B}_x^{m>0}(i_x, 1) \tag{49}$$

If we were to use the physical reflection rule here, then we would have these two fields be equal for even-m, but this would imply a constant field-mode through $r = 0$ instead of dropping to 0 within the simulation window. This is illustrated in Figure 6, which shows that while neither proposed condition

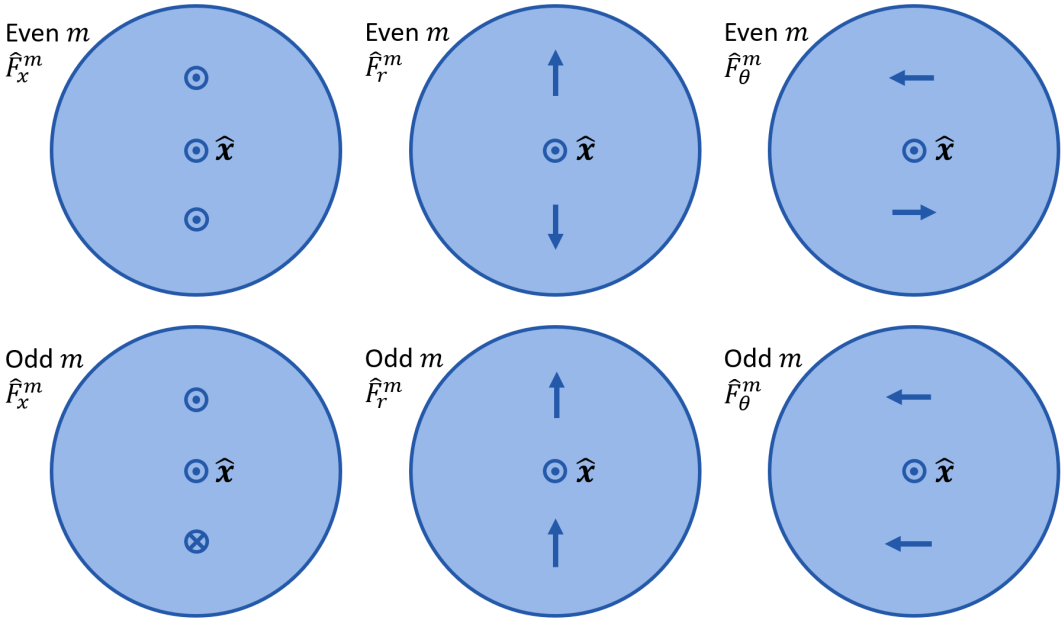

Figure 5: A positive value for a generic field mode $\hat{F}^m e^{im\theta}$ may or may not change direction when viewed at θ and $\theta + \pi$. This diagram details how different components change on opposite sides of the $r = 0$ axis, to be used in field interpolation for macro-particle shapes which dip below $r = 0$.

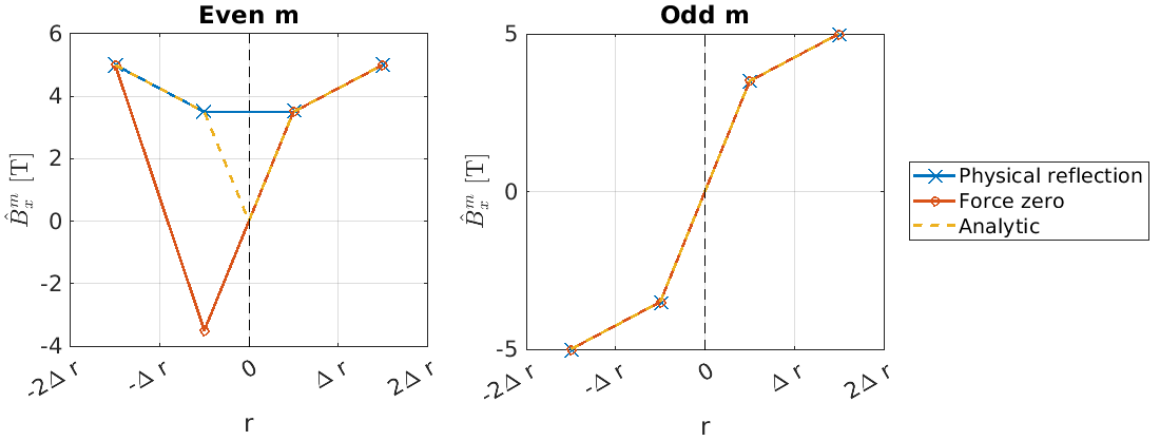

Figure 6: An illustration of different $\hat{B}_x^{m>0}$ evaluation choices for ghost cells with $r < 0$. The physical reflection condition (48) is compared to a boundary which forces the interpolation to pass through zero (49). An analytic interpolation which has the desired behaviour, but includes an additional $r = 0$ evaluation point, has also been provided.

produces the exact desired result, the method proposed in (49) is correct over more r values within the simulation window.

The remaining special cases follow a similar logic, and have been chosen to force the field interpolation to pass through some value at $r = 0$. These include

$$\hat{B}_\theta^1(i_x, 0) = -2i\hat{B}_r^1(i_x, 0) - \hat{B}_\theta^1(i_x, 1) \tag{50}$$

$$\hat{B}_\theta^{m\neq 1}(i_x, 0) = -\hat{B}_\theta^{m\neq 1}(i_x, 1) \tag{51}$$

$$\hat{E}_r^1(i_x, 0) = 2i\hat{E}_\theta^1(i_x, 0) - \hat{E}_r^1(i_x, 1) \tag{52}$$

$$\hat{E}_r^{m\neq 1}(i_x, 0) = -\hat{E}_r^{m\neq 1}(i_x, 1) \tag{53}$$

Here, (50) forces $\hat{B}_r^1(x, r = 0) = i\hat{B}_\theta^1(x, r = 0)$ as demanded by (30), with (52) providing an equivalent expression for the electric field modes. The conditions in (51) and (53) describes the same condition as their $m = 1$ counterparts, but we have forced $\hat{B}_r^{m\neq 1}(x, r = 0) = \hat{E}_\theta^{m\neq 1}(x, r = 0) = 0$.

4.2 Field boundary conditions on $r_{\max}$

Lifschitz *et al* [2] provide no description for how to implement the $r_{\max}$ boundary. The benchmarks present in their paper include moving windows with large numbers of cells in the radial direction, and so any $r_{\max}$ boundary condition would be inconsequential. When attempting to translate the existing $y_{\max}$ EPOCH boundary to cylindrical co-ordinates, I found the boundary became unstable, so alternate conditions were sought.

Cylindrical EPOCH has been fitted with two forms of boundary condition - the first is an outflow boundary adapted from the "Buneman" condition in Smilei [3], and the second is a simple zero-field condition. More background theory and justification will be given in the proceeding subsections. In both models, the aim was to set the values of $\hat{B}_x^m(i_x, n_r)$ and $\hat{B}_\theta^m(i_x, n_r)$ such that there was no dependence on any evaluation point with a higher r value.

4.2.1 Outflow boundaries

In an outflow boundary condition, we assume that any fields present must represent EM plane waves propagating through the boundary. Let us therefore assume that on the $r_{\max}$ boundary, we have waves propagating radially outwards with

$$\hat{E}_\theta^m = c\hat{B}_x^m \tag{54}$$

$$-\hat{E}_x^m = c\hat{B}_\theta^m \tag{55}$$

To obtain equations on these boundaries, we can combine the Maxwell mode equations according to (c(17) + (22)) and (c(19) - (22)), which yield

$$2c\frac{\partial}{\partial t}\hat{B}_x^m + 2c^2\frac{\partial}{\partial r}\hat{B}_x^m = -\frac{c}{r}\hat{E}_\theta^m + c^2\frac{\partial}{\partial x}\hat{B}_r^m - \frac{1}{\epsilon_0}\hat{J}_\theta^m - \frac{imc}{r}\hat{E}_r^m \tag{56}$$

$$2c\frac{\partial}{\partial t}\hat{B}_\theta^m + 2c^2\frac{\partial}{\partial r}\hat{B}_\theta^m + \frac{c^2}{r}\hat{B}_\theta^m = -c\frac{\partial}{\partial x}\hat{E}_r^m - \frac{imc^2}{r}\hat{B}_r^m + \frac{1}{\epsilon_0}\hat{J}_x^m \tag{57}$$

where we have replaced some field modes according to the plane wave equations (54) and (55). The gradients in boundary equations (56) and (57) are calculated centred on the evaluation times and positions of $\hat{E}_\theta^m(i_x, n_r - 1)$ and $\hat{E}_x^m(i_x, n_r - 1)$ respectively, where n_r is the number of cells in the radial direction on the current MPI rank. For example, some of the gradients in (56) may be discretised

according to

$$\frac{\partial}{\partial t}\hat{B}_x^m \approx \frac{1}{2\Delta t}\left(\left(\hat{B}_x^m\right)_{(i_x,n_r)}^{t=t_0+\Delta t} + \left(\hat{B}_x^m\right)_{(i_x,n_r-1)}^{t=t_0+\Delta t} - \left(\hat{B}_x^m\right)_{(i_x,n_r)}^{t=t_0} - \left(\hat{B}_x^m\right)_{(i_x,n_r-1)}^{t=t_0}\right) \tag{58}$$

$$\frac{\partial}{\partial r}\hat{B}_x^m \approx \frac{1}{2\Delta r}\left(\left(\hat{B}_x^m\right)_{(i_x,n_r)}^{t=t_0+\Delta t} - \left(\hat{B}_x^m\right)_{(i_x,n_r-1)}^{t=t_0+\Delta t} + \left(\hat{B}_x^m\right)_{(i_x,n_r)}^{t=t_0} - \left(\hat{B}_x^m\right)_{(i_x,n_r-1)}^{t=t_0}\right) \tag{59}$$

$$\frac{imc}{r}\hat{E}_r^m \approx \frac{imc}{2\left(r_0+\left(n_r-\frac{3}{2}\right)\Delta r\right)}\left(\left(\hat{E}_r^m\right)_{(i_x,n_r)}^{t=t_0+\Delta t/2} + \left(\hat{E}_r^m\right)_{(i_x,n_r-1)}^{t=t_0+\Delta t/2}\right) \tag{60}$$

$$\frac{1}{\epsilon_0}\hat{J}_\theta^m \approx \frac{1}{2\epsilon_0}\left(\left(\hat{J}_\theta^m\right)_{(i_x,n_r-1)}^{t=t_0+\Delta t} + \left(\hat{J}_\theta^m\right)_{(i_x,n_r-1)}^{t=t_0}\right) \tag{61}$$

where r_0 is the central radial position of the cell with $i_r = 1$ on the current rank. Because magnetic fields and currents are staggered in time from the electric field points, we must store the currents and magnetic fields from the previous step in order to compute time-averages to evaluate at the electric field time. The complex arrays used to save these old values are named `bxm_old`, `brm_old`, `btm_old`, `jxm_old`, `jrm_old`, and `jtm_old` in cylindrical EPOCH. Once all gradients are evaluated, we obtain the following expressions for the magnetic field modes at $i_r = n_r$

$$\begin{aligned}\left(\hat{B}_x^m\right)_{(i_x,n_r)}^{t=t_0+\Delta t} \approx \frac{1}{c+\frac{c^2\Delta t}{\Delta r}}\Bigg(&-\left(c-\frac{c^2\Delta t}{\Delta r}\right)\left(\hat{B}_x^m\right)_{(i_x,n_r-1)}^{t=t_0+\Delta t} - \left(-c+\frac{c^2\Delta t}{\Delta r}\right)\left(\hat{B}_x^m\right)_{(i_x,n_r)}^{t=t_0}\\ &-\left(-c-\frac{c^2\Delta t}{\Delta r}\right)\left(\hat{B}_x^m\right)_{(i_x,n_r-1)}^{t=t_0} - \frac{c\Delta t}{r}\left(\hat{E}_\theta^m\right)_{(i_x,n_r-1)}^{t=t_0+\Delta t/2}\\ &+\frac{c^2\Delta t}{2\Delta x}\left(\left(\hat{B}_r^m\right)_{(i_x+1,n_r-1)}^{t=t_0+\Delta t} - \left(\hat{B}_r^m\right)_{(i_x,n_r-1)}^{t=t_0+\Delta t} + \left(\hat{B}_r^m\right)_{(i_x+1,n_r-1)}^{t=t_0} - \left(\hat{B}_r^m\right)_{(i_x,n_r-1)}^{t=t_0}\right)\\ &-\frac{\Delta t}{2\epsilon_0}\left(\left(\hat{J}_\theta^m\right)_{(i_x,n_r-1)}^{t=t_0+\Delta t} + \left(\hat{J}_\theta^m\right)_{(i_x,n_r-1)}^{t=t_0}\right)\\ &-\frac{imc\Delta t}{2r}\left(\left(\hat{E}_r^m\right)_{(i_x,n_r)}^{t=t_0+\Delta t/2} + \left(\hat{E}_r^m\right)_{(i_x,n_r-1)}^{t=t_0+\Delta t/2}\right)\Bigg)\end{aligned} \tag{62}$$

$$\begin{aligned}\left(\hat{B}_\theta^m\right)_{(i_x,n_r)}^{t=t_0+\Delta t} \approx \frac{1}{c+\frac{c^2\Delta t}{\Delta r}+\frac{c^2\Delta t}{4r}}\Bigg(&-\left(c-\frac{c^2\Delta t}{\Delta r}+\frac{c^2\Delta t}{4r}\right)\left(\hat{B}_\theta^m\right)_{(i_x,n_r-1)}^{t=t_0+\Delta t}\\ &-\left(-c+\frac{c^2\Delta t}{\Delta r}+\frac{c^2\Delta t}{4r}\right)\left(\hat{B}_\theta^m\right)_{(i_x,n_r)}^{t=t_0} - \left(-c-\frac{c^2\Delta t}{\Delta r}+\frac{c^2\Delta t}{4r}\right)\left(\hat{B}_\theta^m\right)_{(i_x,n_r-1)}^{t=t_0}\\ &-\frac{c\Delta t}{2\Delta x}\left(\left(\hat{E}_r^m\right)_{(i_x,n_r)}^{t=t_0+\Delta t/2} + \left(\hat{E}_r^m\right)_{(i_x,n_r-1)}^{t=t_0+\Delta t/2} - \left(\hat{E}_r^m\right)_{(i_x-1,n_r)}^{t=t_0+\Delta t/2} - \left(\hat{E}_r^m\right)_{(i_x-1,n_r-1)}^{t=t_0+\Delta t/2}\right)\\ &-\frac{imc^2\Delta t}{2r}\left(\left(\hat{B}_r^m\right)_{(i_x,n_r-1)}^{t=t_0+\Delta t} + \left(\hat{B}_r^m\right)_{(i_x,n_r-1)}^{t=t_0}\right)\\ &+\frac{\Delta t}{2\epsilon_0}\left(\left(\hat{J}_x^m\right)_{(i_x,n_r-1)}^{t=t_0+\Delta t} + \left(\hat{J}_x^m\right)_{(i_x,n_r-1)}^{t=t_0}\right)\Bigg)\end{aligned} \tag{63}$$

where $r = r_0 + (n_r - 1.5)\Delta r$ at the evaluation point, and in this section, r_0 refers to the radius of the $i_r = 1$ cell-centre on the current MPI rank.

4.2.2 Zero field

The outflow boundary conditions were found to become unstable in the two-stream instability test. In this test, two plasma streams counter-propagate with periodic boundary conditions in the x direction, and no laser is present. As such, perhaps the plane wave approximations (54) and (55) are unsuitable in this system, leading to the non-physical boundary fields. To address this, we have implemented an additional boundary condition

$$\left(\hat{B}_x^m\right)_{(i_x,n_r)}^{t=t_0+\Delta t} = 0 \tag{64}$$

$$\left(\hat{B}_\theta^m\right)_{(i_x,n_r)}^{t=t_0+\Delta t} = 0 \tag{65}$$

for use in systems where negligible magnetic fields are expected in the high-r limit.

4.3 Fields and lasers on the x boundary

Lifschitz *et al* describe modelling of a linearly-polarised plane wave in their equation 3 [2], which is derived from the cylindrical to Cartesian component transformations in (6) and (7). A linearly-polarised wave passing through the $x_{\min}$ boundary satisfies $E_y = cB_z$ and $E_z = -cB_y$, so equation 3 in Lifschitz *et al* [2] comes from combining (6) and (7) with the field mode definition (1) to obtain

$$\hat{E}_r^1 = E_y + iE_z \tag{66}$$

$$\hat{E}_\theta^1 = -iE_y + E_z \tag{67}$$

$$\hat{B}_r^1 = \frac{1}{c}\left(-E_z + iE_y\right) \tag{68}$$

$$\hat{B}_\theta^1 = \frac{1}{c}\left(iE_z + E_y\right) \tag{69}$$

Hence, any plane-wave laser previously defined by E_y and E_z, with an axially-symmetric spatial profile, may be characterised in cylindrical EPOCH using only the $m = 1$ mode.

Boundary conditions are required to inject a laser into the simulation, but such treatment is not covered by Lifschitz *et al* [2]. Here we will derive a cylindrical form of the boundaries already present in EPOCH, which yield equations similar to the "Silver-Müller" boundary conditions present in Smilei [3]. As in the $r_{\max}$ boundary case of Section 4.2, we seek to find equations for $\hat{B}_r^m(1, i_r)$ and $\hat{B}_\theta^m(1, i_r)$ on ranks with the $x_{\min}$ boundary, and $\hat{B}_r^m(n_x, i_r)$ and $\hat{B}_\theta^m(n_x, i_r)$ on ranks with the $x_{\max}$ boundary, such that we have no dependence on evaluation points outside the simulation window.

Equations for the boundary $\hat{\boldsymbol{B}}^m$ fields may be derived from the Ampère-Maxwell law (14), using the cylindrical curl (15) and the θ derivative for field modes (16). These combine to give

$$\frac{\partial}{\partial t}\hat{E}_\theta^m + \frac{1}{\epsilon_0}\hat{J}_\theta^m = c^2\frac{\partial}{\partial x}\hat{B}_r^m - c^2\frac{\partial}{\partial r}\hat{B}_x^m \tag{70}$$

$$\frac{\partial}{\partial t}\hat{E}_r^m + \frac{1}{\epsilon_0}\hat{J}_r^m = -\frac{imc^2}{r}\hat{B}_x^m - c^2\frac{\partial}{\partial x}\hat{B}_\theta^m \tag{71}$$

To proceed, these gradients can be evaluated at a time and position centred on the evaluation points of $\hat{J}_\theta^m(1, i_r)$ and $\hat{J}_\theta^m(n_x - 1, i_r)$ for (70), and $\hat{J}_r^m(1, i_r)$ and $\hat{J}_r^m(n_x - 1, i_r)$ for (71), on the $x_{\min}$ and $x_{\max}$ boundaries respectively. The current densities are evaluated at $t = t_0$, which is half a time-step ahead of the electric field. We may write the discretisation of the electric field gradients as

$$\frac{\partial}{\partial t}\hat{E}_\theta^m \approx \frac{1}{\Delta t}\left(\left(\hat{E}_\theta^m\right)^{t=t_0+\Delta t/2} - \left(\hat{E}_\theta^m\right)^{t=t_0-\Delta t/2}\right) \tag{72}$$

$$\frac{\partial}{\partial t}\hat{E}_r^m \approx \frac{1}{\Delta t}\left(\left(\hat{E}_r^m\right)^{t=t_0+\Delta t/2} - \left(\hat{E}_r^m\right)^{t=t_0-\Delta t/2}\right) \tag{73}$$

but this cannot be evaluated in its current form as the electric field at $t = t_0 + \Delta t/2$ is not yet known. Instead, we may use the time-average of the electric field

$$\left(\hat{\boldsymbol{E}}^m\right)^{t=t_0} = \frac{1}{2}\left(\left(\hat{\boldsymbol{E}}^m\right)^{t=t_0+\Delta t/2} + \left(\hat{\boldsymbol{E}}^m\right)^{t=t_0-\Delta t/2}\right) \tag{74}$$

to write

$$\frac{\partial}{\partial t}\hat{E}_\theta^m \approx \frac{1}{\Delta t}\left(4\left(\hat{S}_\theta^m\right)^{t=t_0} \pm 2c\left(\hat{B}_r^m\right)^{t=t_0} - 2\left(\hat{E}_\theta^m\right)^{t=t_0-\Delta t/2}\right) \tag{75}$$

$$\frac{\partial}{\partial t}\hat{E}_r^m \approx \frac{1}{\Delta t}\left(4\left(\hat{S}_r^m\right)^{t=t_0} \mp 2c\left(\hat{B}_\theta^m\right)^{t=t_0} - 2\left(\hat{E}_r^m\right)^{t=t_0-\Delta t/2}\right) \tag{76}$$

where S_θ and S_r describe laser source terms, and all terms are evaluated at the relevant current density position. Here the top and bottom signs denote equations for the $x_{\min}$ and $x_{\max}$ boundaries respectively, as the source terms take different forms on these axes

$$\hat{S}_\theta^m = \hat{E}_\theta^m = \frac{1}{2}\left(\hat{E}_\theta^m \mp c\hat{B}_r^m\right) \tag{77}$$

$$\hat{S}_r^m = \hat{E}_r^m = \frac{1}{2}\left(\hat{E}_r^m \pm c\hat{B}_\theta^m\right) \tag{78}$$

since they refer to plane waves travelling into the simulation window from different directions. We note that $\hat{B}_r^m$ and $\hat{B}_\theta^m$ have evaluation points which do not coincide with the other terms of (75) and (76) respectively, so we take the averages of neighbouring evaluation points. Hence, when discretising the full boundary equations (70) and (71), we obtain $x_{\min}$ boundaries of the form

$$\begin{aligned}\left(\hat{B}_r^m\right)_{(1,i_r)}^{t=t_0} \approx &\frac{1}{c + c^2\frac{\Delta t}{\Delta x}}\left(-4\left(S_\theta^m\right)^{t=t_0} + \left(c^2\frac{\Delta t}{\Delta x} - c\right)\left(B_r^m\right)_{(2,i_r)}^{t=t_0} + 2\left(E_\theta^m\right)_{(1,i_r)}^{t=t_0-\Delta t/2}\right. \\ &\left. - c^2\frac{\Delta t}{\Delta r}\left(\left(B_x^m\right)_{(1,i_r+1)}^{t=t_0} - \left(B_x^m\right)_{(1,i_r)}^{t=t_0}\right) - \frac{\Delta t}{\epsilon_0}\left(J_\theta^m\right)_{(1,i_r)}^{t=t_0}\right)\end{aligned} \tag{79}$$

$$\begin{aligned}\left(\hat{B}_\theta^m\right)_{(1,i_r)}^{t=t_0} \approx &\frac{1}{c + c^2\frac{\Delta t}{\Delta x}}\left(4\left(S_r^m\right)^{t=t_0} + \left(c^2\frac{\Delta t}{\Delta x} - c\right)\left(B_\theta^m\right)_{(2,i_r)}^{t=t_0} - 2\left(E_r^m\right)_{(1,i_r)}^{t=t_0-\Delta t/2}\right. \\ &\left. + \frac{imc^2\Delta t}{r}\left(B_x^m\right)_{(1,i_r)}^{t=t_0} + \frac{\Delta t}{\epsilon_0}\left(J_r^m\right)_{(1,i_r)}^{t=t_0}\right)\end{aligned} \tag{80}$$

and $x_{\max}$ boundaries

$$\begin{aligned}\left(\hat{B}_r^m\right)_{(n_x,i_r)}^{t=t_0} \approx &\frac{1}{c + c^2\frac{\Delta t}{\Delta x}}\left(4\left(S_\theta^m\right)^{t=t_0} + \left(c^2\frac{\Delta t}{\Delta x} - c\right)\left(B_r^m\right)_{(n_x-1,i_r)}^{t=t_0} - 2\left(E_\theta^m\right)_{(n_x-1,i_r)}^{t=t_0-\Delta t/2}\right. \\ &\left. + c^2\frac{\Delta t}{\Delta r}\left(\left(B_x^m\right)_{(n_x-1,i_r+1)}^{t=t_0} - \left(B_x^m\right)_{(n_x-1,i_r)}^{t=t_0}\right) + \frac{\Delta t}{\epsilon_0}\left(J_\theta^m\right)_{(n_x-1,i_r)}^{t=t_0}\right)\end{aligned} \tag{81}$$

$$\begin{aligned}\left(\hat{B}_\theta^m\right)_{(n_x,i_r)}^{t=t_0} \approx &\frac{1}{c + c^2\frac{\Delta t}{\Delta x}}\left(-4\left(S_r^m\right)^{t=t_0} + \left(c^2\frac{\Delta t}{\Delta x} - c\right)\left(B_\theta^m\right)_{(n_x-1,i_r)}^{t=t_0} + 2\left(E_r^m\right)_{(n_x-1,i_r)}^{t=t_0-\Delta t/2}\right. \\ &\left. - \frac{imc^2\Delta t}{r}\left(B_x^m\right)_{(n_x-1,i_r)}^{t=t_0} - \frac{\Delta t}{\epsilon_0}\left(J_r^m\right)_{(n_x-1,i_r)}^{t=t_0}\right)\end{aligned} \tag{82}$$

5 Particle dynamics

Cylindrical EPOCH is a modified version of EPOCH2D, which already came equipped with a particle-pusher for updating all 3 macro-particle momenta components. As macro-particles have (x, y, z) coordinates in cylindrical EPOCH, the z momentum was modified to update the position in the previously missing dimension.

Modifications were also made for interpolating EM fields over the macro-particle shape. The weighting factors `gx`, `gy`, `hx` and `hy` were used as before to determine the macro-particle fraction in each cell on the standard and staggered grid, but have been applied to different fields due to the different cell-stagger shown in Figure 2. The total field strength is calculated for a given macro-particle by summing over modes as in (1), using the θ position of the particle.

Currently cylindrical EPOCH supports open, periodic and reflecting boundary conditions for $x_{\min}$ and $x_{\max}$, but only open and reflecting for $r_{\max}$. Reflection treatment is only approximate on the $r_{\max}$ boundary, as discussed in Section 5.1.

5.1 Reflecting boundary conditions

With the current design of EPOCH, only an approximate reflective boundary can be easily implemented on $r_{\max}$. For a proper treatment, the code would store the r values before and after the step,

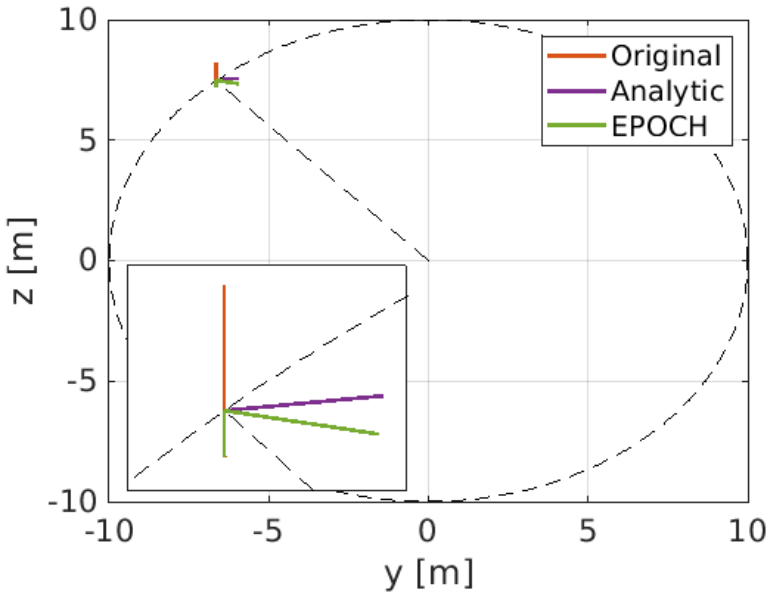

Figure 7: Particle reflection implementations on the $r_{\max}$ boundary. The original trajectory of a particle being pushed to a position outside the simulation window is shown, and two reflections at $r_{\max}$ have been calculated. The analytic implementation matches the incident and outgoing angles between the momentum and target normal, and the EPOCH implementation reflects in r only.

the macro-particle position would be deduced at $r = r_{\max}$ along with the angle between the momentum vector and the boundary normal. From here, the code would reflect the momentum vector about the normal, and determine the new end position assuming the rest of the macro-particle step was in this new direction. Such an analytic reflection has been illustrated in Figure 7 The issue with this method is that EPOCH would have to store an initial r value for all macro-particles, and reflected macro-particles may pass $r = r_{\max}$ again, potentially multiple times within a single step. Unless a user requires an accurate $r_{\max}$ boundary treatment, an approximate method should suffice.

In the current implementation of the boundary treatment, macro-particles are only reflected in the radial direction. If a macro-particle ends a step at position $(x, r_{\max} + r_p, \theta)$ in cylindrical co-ordinates, then it is reflected to position $(x, r_{\max} - r_p, \theta)$. The radial momentum at the end of the step is also flipped, and the new Cartesian momenta p'_y and p'_z become

$$p'_y = -p_r \cos\theta - p_\theta \sin\theta \tag{83}$$

$$p'_z = -p_r \sin\theta + p_\theta \cos\theta \tag{84}$$

where p_r and p_θ are cylindrical momentum components calculated using p_y and p_z before any reflection, according to (6) and (7). While similar, this will produce a different reflection to the full method, and this difference is shown in Figure 7.

6 Current density modes

In EPOCH, current density is calculated according to the Esirkepov method [4]. As macro-particles are pushed from one position to another, the weights of the macro-particle shape in the local cells change, and this weight change is made equivalent to the passing of current. While we can use this same treatment on a cylindrical grid, we must take this macro-particle current and convert it to current modes. Care must also be taken to ensure the charge-conserving principles of Esirkepov [4] are also maintained in this process.

Lifschitz *et al* provide some details on how to convert a standard current density into modes [2], and the Smilei code provides an example implementation [3]. However, certain parts of the Smilei

method are only valid in the limit where macro-particles make small angular changes over a given step, where $\Delta\theta \to 0$. This condition will be met for macro-particles at large r, but may be violated near the $r = 0$ axis.

We have developed a new current density method which will work for the general case, following the principles of Esirkepov [4]. In the following subsections, we will explain the conversion from macro-particle current density to field modes, describe a general method for calculating the current entering or leaving a cell, and demonstrate how this can be used to determine current density modes on a cylindrical grid.

6.1 Current density to field modes

Using macro-particle weights, we can determine the total current passing between cells, but we must convert this to current density modes. As is covered in the Smilei documentation [3], by integrating both sides of the field-mode definition (1) multiplied by $e^{in\theta}$, we get

$$\int_0^{2\pi} J_x(x,r,\theta)e^{in\theta}d\theta = \int_0^{2\pi} \sum_{m=0}^{\infty} \left(\mathbb{R}\left\{\hat{J}_x^m(x,r)\right\}\cos(m\theta) + \mathbb{I}\left\{\hat{J}_x^m(x,r)\right\}\sin(m\theta)\right) e^{in\theta}d\theta \tag{85}$$

where n is an integer. Using orthogonality rules, this reduces to

$$\int_0^{2\pi} J_x(x,r,\theta)e^{in\theta}d\theta = \begin{cases} 2\pi\mathbb{R}\left\{\hat{J}_x^0(x,r)\right\} & n = 0 \\ \pi\mathbb{R}\left\{\hat{J}_x^n(x,r)\right\} + i\pi\mathbb{I}\left\{\hat{J}_x^n(x,r)\right\} & n > 0 \end{cases} \tag{86}$$

The macro-particles only have a shape on the 2D grid, and so the current density from a single macro-particle may be expressed as $J_x(x,r,\theta) = J_x(x,r)\delta(\theta - \theta_p)$, where θ_p is the macro-particle θ position. This yields an expression for the current density modes including the factors quoted by Lifschitz *et al* in their equation 17 [2],

$$\hat{J}_x^m(x,r) = \begin{cases} \frac{1}{2\pi}J_x(x,r) & m = 0 \\ \frac{1}{\pi}e^{im\theta_p}J_x(x,r) & m > 0 \end{cases} \tag{87}$$

however, these are not the expressions used by Smilei [3] or cylindrical EPOCH. The Lifschitz *et al* equations do not account for the inconsistencies with treating 2D macro-particles in a 3D volume, as weights were assigned to macro-particles assuming they occupied a 3D volume in (11). If we considered the plasma as a whole instead of an individual macro-particle, and assumed an axially symmetric current density $J_x(x,r,\theta) = J_x(x,r)$, then we would have

$$\hat{J}_x^m(x,r) = \begin{cases} J_x(x,r) & m = 0 \\ 0 & m > 0 \end{cases} \tag{88}$$

and as this argument invokes no simulation features, this must represent the correct physical answer. For the $m = 0$ case, we see that the Lifschitz equation underestimates the current density mode by a factor of 2π, so we will insert a correction factor to compensate for this. Here we arrive at the current density mode equations used by Smilei [3], which in our notation reads

$$\hat{J}_x^m(x,r) = \begin{cases} J_x(x,r) & m = 0 \\ 2e^{im\theta_p}J_x(x,r) & m > 0 \end{cases} \tag{89}$$

and this is also the method used in cylindrical EPOCH.

6.2 Current modes from macro-particle motion

In both EPOCH2D and cylindrical EPOCH, macro-particle shapes are expressed as the product of two, one-dimensional shape functions. Hence, the total macro-particle weight fraction in a cylindrical cell may be written as $g_x(i_x)g_r(i_r)$, where g_x and g_r are arrays which describe the fraction of the 1D

macro-particle shapes in each cell. The changes in these weight fractions over a given time-step will therefore describe the net charge change in the cell, which provides information about the current density. For simplicity, we will only consider current modes for $\hat{I_x}^m$ and $\hat{I_r}^m$ in this section. Current densities in the cylindrical cell geometry will instead be covered in Section 6.3.

To derive an equation for current modes, let us consider the current contribution of a single macro-particle with charge Q (particle charge multiplied by macro-particle weight) during a particle-push in a cell. The macro-particle has weight factors g_x and g_r in this cell at the initial position ($t = 0$), and factors g'_x and g'_r at the final position ($t = \Delta t$). Let h_x and h_r describe the weight change over this push, such that

$$h_x = g'_x - g_x \tag{90}$$
$$h_r = g'_r - g_r \tag{91}$$

The net current flowing through a cell can be calculated using these weight functions. For example, $Qh_x(i_x)$ describes the charge change due to motion in the x-direction, summed over all cells which share an i_x index. The fraction of this charge change in a cell with index (i_x, i_r) is $Qh_x(i_x)\langle W(i_r)\rangle$, where $\langle W(i_r)\rangle$ is $(g'_r + g_r)/2$, the average perpendicular weight in the cell of interest during the particle-push. Once we know the x-directed change of charge in a cell, the net I_x flowing through the cell can be expressed as

$$I_x = \frac{Q}{2\Delta t} h_x(i_x)\left(g'_r(i_r) + g_r(i_r)\right) \tag{92}$$

We note that this expression only gives the net current entering or leaving the cell, not the current values on the cell boundaries.

This simple example describes the Cartesian case, but further analysis of the $\langle W(i_r)\rangle$ term is required for cylindrical field modes. Let us write

$$\hat{I}^m_x = \frac{Q}{\Delta t} h_x \langle W^m_{r\theta}\rangle \tag{93}$$
$$\hat{I}^m_r = \frac{Q}{\Delta t} h_r \langle W^m_{x\theta}\rangle \tag{94}$$

where $\langle W^m_{r\theta}\rangle$ and $\langle W^m_{x\theta}\rangle_m$ describe the average perpendicular weight, combined with the conversion factors in (89). To evaluate these average weights, we can produce a general expression for the weight at time t

$$W^m_{r\theta}(t) = \left(g_r + \frac{t}{\Delta t} h_r\right)\begin{cases} 1 & m = 0 \\ 2e^{im\left(\theta_0 + \Delta\theta \frac{t}{\Delta t}\right)} & m > 0 \end{cases} \tag{95}$$

$$W^m_{x\theta}(t) = \left(g_x + \frac{t}{\Delta t} h_x\right)\begin{cases} 1 & m = 0 \\ 2e^{im\left(\theta_0 + \Delta\theta \frac{t}{\Delta t}\right)} & m > 0 \end{cases} \tag{96}$$

where θ_0 is the θ position of the macro-particle at time $t = 0$, and $\theta_0 + \Delta\theta$ is that for $t = \Delta t$. The mean values of these cylindrical weights are equivalent to the expectation values, which may be calculated as

$$\langle W^m\rangle = \frac{1}{\Delta t}\int_0^{\Delta t} W^m(t)dt \tag{97}$$

which produces

$$\langle W^m_{r\theta}\rangle = \begin{cases} g_r + \frac{1}{2}h_r & m = 0 \\ \frac{2}{m\Delta\theta} e^{im\theta_0}\left(-ig_r\left(e^{im\Delta\theta} - 1\right) + \frac{h_r}{m\Delta\theta}\left(e^{im\Delta\theta}(1 - im\Delta\theta) - 1\right)\right) & m > 0 \end{cases} \tag{98}$$

$$\langle W^m_{x\theta}\rangle = \begin{cases} g_x + \frac{1}{2}h_x & m = 0 \\ \frac{2}{m\Delta\theta} e^{im\theta_0}\left(-ig_x\left(e^{im\Delta\theta} - 1\right) + \frac{h_x}{m\Delta\theta}\left(e^{im\Delta\theta}(1 - im\Delta\theta) - 1\right)\right) & m > 0 \end{cases} \tag{99}$$

A similar treatment can be applied to the current density in the θ direction. We may write

$$J_\theta^m = \frac{Q}{V} v_\theta \langle W_{xr}^m \rangle \tag{100}$$

where v_θ is the macro-particle velocity in the θ direction, which is used because there is no macro-particle shape in the θ direction to obtain a g_θ or h_θ. Here, V is the cell volume, as given by (10). For θ currents, the full macro-particle shape is transverse to the motion, so we have

$$\langle W_{xr}^m \rangle = \left(g_x + \frac{t}{\Delta t} h_x\right)\left(g_r + \frac{t}{\Delta t} h_r\right) \begin{cases} 1 & m = 0 \\ 2e^{im\left(\theta_0 + \Delta\theta \frac{t}{\Delta t}\right)} & m > 0 \end{cases} \tag{101}$$

which can be solved as

$$\langle W_{xr}^m \rangle = \begin{cases} g_x g_r + \frac{1}{2}(h_x g_r + h_r g_x) + \frac{1}{3} h_x h_r & m = 0 \\ \frac{2}{m\Delta\theta} e^{im\theta_0} \left(-i g_x g_r \left(e^{im\Delta\theta} - 1\right)\right. & \\ \quad + \frac{1}{m\Delta\theta}(g_x h_r + g_r h_x)\left(e^{im\Delta\theta}(1 - im\Delta\theta) - 1\right) & \\ \quad \left. + \frac{h_x h_r}{m^2\Delta\theta^2}\left(e^{im\Delta\theta}(-im^2\Delta\theta^2 + 2m\Delta\theta + 2i) - 2i\right)\right) & m > 0 \end{cases} \tag{102}$$

It is clear from inspection of (98), (99) and (102) that these weights become undefined at $\Delta\theta = 0$, but our issues start before this. For $\Delta\theta = 10^{-11}$ radians, it was found that the real component of $e^{i\theta}$ was evaluated as exactly 1 in EPOCH precision, perfectly cancelling the real part of $e^{im\Delta\theta} - 1$ and leading to noisy currents. Hence, for $|m\Delta\theta| < 10^{-4}$, we expand the terms about $m\Delta\theta$ to produce

$$\langle W_{r\theta}^m \rangle = 2e^{im\theta_0}\left(g_r\left(1 + \frac{1}{2}im\Delta\theta - \frac{1}{6}m^2\Delta\theta^2\right) + h_r\left(\frac{1}{2} + \frac{1}{3}im\Delta\theta - \frac{1}{8}m^2\Delta\theta^2\right)\right) \tag{103}$$

$$\langle W_{x\theta}^m \rangle = 2e^{im\theta_0}\left(g_x\left(1 + \frac{1}{2}im\Delta\theta - \frac{1}{6}m^2\Delta\theta^2\right) + h_x\left(\frac{1}{2} + \frac{1}{3}im\Delta\theta - \frac{1}{8}m^2\Delta\theta^2\right)\right) \tag{104}$$

$$\begin{aligned} \langle W_{xr}^m \rangle = 2e^{im\theta_0}\Bigg(&g_x g_r\left(1 + \frac{1}{2}im\Delta\theta - \frac{1}{6}m^2\Delta\theta^2\right) + (g_x h_r + g_r h_x)\left(\frac{1}{2} + \frac{1}{3}im\Delta\theta - \frac{1}{8}m^2\Delta\theta^2\right) \\ &+ h_x h_r\left(\frac{1}{3} + \frac{1}{4}im\Delta\theta - \frac{1}{10}m^2\Delta\theta^2\right)\Bigg) \end{aligned} \tag{105}$$

which are used for small-angle, $m > 0$ current calculation.

6.3 Current density modes on a cylindrical grid

Section 6.1 discussed the relationship between currents and current modes in cylindrical EPOCH, and this was used in Section 6.2 to calculate net currents and current densities in the cell. In this section, we seek to use these currents to determine current densities on specific boundaries, at the evaluation points in the EPOCH grid. Figure 8 shows the geometry of a single macro-particle push, and provides some example g_x, g_r, h_x and h_r values.

The simplest current density to solve is $\hat{J}_\theta^m$, as this is given by (100). As can be seen in Figure 2, a $\hat{J}_\theta^m$ evaluation point exists in the centre of the staggered cell ($i_x = 0, i_r = 0$). This point corresponds to a position of $(x_0 + 2\Delta x, r_0 + 2\Delta r)$, or an index $\hat{J}_\theta^m((x_0 - x_{\min})/\Delta x + 2, r_0/\Delta r + 2)$ if we ignore MPI domain decomposition. The volume of the staggered cell V would be $2\pi(r_0 + 2\Delta r)\Delta r \Delta x$ following (10), and $\langle W_{xr}^m \rangle$ could be determined using the g and h factors associated with the $(0, 0)$ cell. Finally, the v_θ value can be deduced using the transformation (7), where v_y and v_z are calculated during the particle push, and θ is taken to be the angular position of the macro-particle at the mid-point. A set of $\hat{J}_\theta^0$ values using the weights of Figure 8 have been calculated using some example values in Table 1.

For $i_r = 0$, there are four $\hat{J}_x^m$ evaluation points of interest, with x positions $(x_0 + \Delta x/2)$, $(x_0 + 3\Delta x/2)$, $(x_0 + 5\Delta x/2)$ and $(x_0 + 7\Delta x/2)$, and all sharing $r = r_0 + 2\Delta r$. As an example, let us first consider $\hat{J}_x^m$ at $(x_0 + \Delta x/2)$ and $(x_0 + 3\Delta x/2)$, on the boundaries of the $i_x = -1$ staggered grid. Let

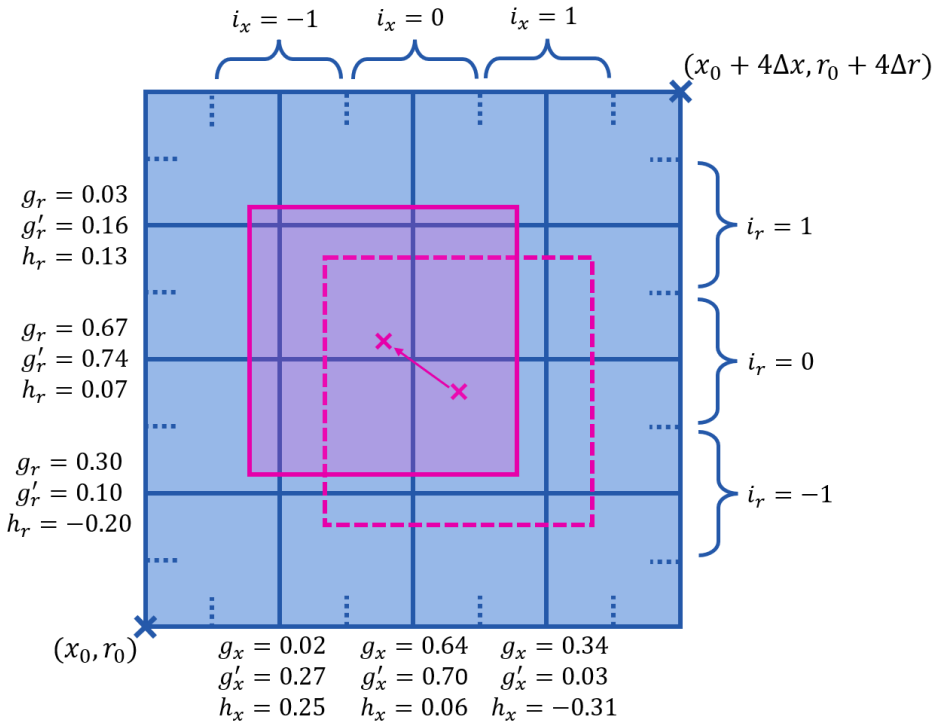

Figure 8: Schematic diagram of a particle-push on the cylindrical EPOCH grid. Solid lines denote cell-edges as in Figure 2, and the incomplete dashed lines describe the staggered cells of interest for the current solver. The regions occupied by a triangular-weighted macro-particle shape is shown before and after a particle-push using dashed and solid pink squares respectively, and staggered cell indices i_x and i_r are given where $(0, 0)$ corresponds to the indices of the staggered cell containing the macro-particle start-position. The weights and weight-changes in each staggered cell are also given. The co-ordinates of two (x, r) points are provided to aid in the discussion.

		x [μm]		
		5.05	5.10	5.15
	5.05	-15.0	-80.5	-25.5
r [μm]	5.10	-62.2	-283	-77.1
	5.15	-9.79	-38.2	-8.44

Table 1: A set of $\hat{J}_\theta^m$ values [$10^5 \mathrm{Am}^{-2}$] obtained using the weights given in Figure 8, for $m = 0$. In this example, we have taken $\Delta x = \Delta r = 50$ nm, $x_0 = r_0 = 5$ μm, $\Delta t = 0.1$ fs, $Q = -1.602 \times 10^{-19}$ C (assuming a macro-electron of weight 1), and $v_\theta = 0.1c$.

us calculate the net $m = 0$ current in this cell using (93), assuming our macro-particle represents a single electron, and a simulation time-step is 0.1 fs. By using the weights from Figure 8, we calculate $\hat{I}_x^0 \approx -2.8 \times 10^{-4}$ A. We can see from Figure 8 that no macro-particle shape passes $x = x_0 + \Delta x/2$, so the current density at this evaluation point must be zero. Therefore, the net current from this staggered cell must pass entirely through the $x = x_0 + 3\Delta x/2$ evaluation point. A drop of -2.8×10^{-4} A may be achieved within the cell if the current on this boundary is $+2.8 \times 10^{-4}$ A, and dividing by the boundary area yields the current density $\hat{J}_x^0$. Recalling the geometry of Figure 4, the boundary area at the $i_r = 0$ staggered cell is $A_{r\theta} = 2\pi(r_0 + 2\Delta r)\Delta r$. For the evaluation point at $(x_0 + 5\Delta x/2)$, we may calculate the net current in the $i_x = 0$ staggered cell, but we must also consider the $+2.8 \times 10^{-4}$ A flowing in through the $(x_0 + 3\Delta x/2)$ boundary. Values for the $m = 0$ current density have been quoted in Table 2 for a particular set of simulation parameters.

		x [μm]			
		5.025	5.075	5.125	5.175
	5.05	0.00	5.05	6.26	0.00
r [μm]	5.10	0.00	17.6	21.9	0.00
	5.15	0.00	2.35	2.92	0.00

Table 2: A set of $\hat{J}_x^m$ values [$10^7 \mathrm{Am}^{-2}$] obtained using the weights given in Figure 8, for $m = 0$. The same parameters have been used as in Table 1.

Finally, the $\hat{J}_r^m$ calculation proceeds in a similar way to $\hat{J}_x^m$. We start by considering the evaluation point with the highest r which has $\hat{J}_r^m = 0$, but which also has $r < r_p$ where r_p is the macro-particle start position. Here, this would be $r = r_0 + \Delta r/2$, such that any current in the $i_r = -1$ cells pass entirely through $\hat{J}_r^m$ evaluation points at $r = r_0 + 3\Delta r/2$. The main difference with $\hat{J}_r^m$ is that we must work with currents when moving up from $r_0 + \Delta r/2$ to $r_0 + 7\Delta r/2$, as the boundary areas $A_{x\theta}$ change with r

$$A_{x\theta} = 2\pi r \Delta x \tag{106}$$

Example values for $\hat{J}_r^0$ have been provided in Table 3.

		x [μm]		
		5.05	5.10	5.15
	5.025	0.00	0.00	0.00
	5.075	-2.91	-13.5	-3.72
r [μm]	5.125	-1.88	-8.67	-2.39
	5.175	0.00	0.00	0.00

Table 3: A set of $\hat{J}_r^m$ values [$10^7 \mathrm{Am}^{-2}$] obtained using the weights given in Figure 8, for $m = 0$. The same parameters have been used as in Table 1.

If the macro-particle central position moved to a neighbouring staggered-cell during the particle-push, then an extra set of evaluation points would be considered by the current solver as more boundaries would have been passed. Macro-particle shapes also have different sizes when considering the B-spline and top-hat interpolation methods. In all cases, all current density evaluation points which

the macro-particle shape passes through are considered in the current density calculation. This includes some neighbouring zero-current cells to deduce current flow on specific boundaries when the net current is used.

7 Test decks

Several practice problems have been designed to test the new features of the cylindrical PIC code, which are covered in the following subsections. As of the time of writing, it should be assumed that if any EPOCH functionality is not listed in these tests, then it will not work in cylindrical mode. Cylindrical EPOCH is modelled after EPOCH2D, and the source code contains a mix of cylindrical-PIC code and unmodified EPOCH2D code. This was done such that extending cylindrical EPOCH with physics packages from the Cartesian codes could be done quickly, as the coding infrastructure is already present. Unfortunately this makes reading the source code a difficult task for new developers, as large portions of the code are not currently used by the cylindrical algorithms.

As cylindrical EPOCH must still support legacy EPOCH2D code in order to compile, there are some quirks in the code. The cylindrical code uses a grid in (x, r) space, but many variables in the code will still refer to a y direction on the grid. Anything referring to y on a grid should be read as a variable in the radial direction, except the cartesian positions and momenta of macro-particles. The input deck will also require users to enter `y_max` and `ny` values, and if setting parameters which vary with radial position (like number density), the user should still refer to `y` instead of `r`.

The cylindrical code will always set `y_min=0`, and will use the special $r = 0$ treatment of fields and particles outlined in previous sections. As such, the input decks will not accept `y_min` or `bc_y_min` parameters. A new `n_mode` parameter must be included in all input decks, which asks for the *number* of modes present. For example, `n_mode = 2` will create mode arrays for $m = 0$ and $m = 1$.

Finally we will note the new output variables available to the output block. To print the complex field modes to the SDF files, we can use `exm`, `erm`, `etm`, `bxm`, `brm`, `btm`, `jxm`, `jrm` and `jtm`.

As the code was formerly EPOCH2D, all these tests may be run as if the user was running an EPOCH2D simulation. The user must compile the code using a terminal command like

```
make COMPILER=gfortran -j4
```

while in the directory `epoch/epoch_axial`. Then an input deck may be run using

```
mpirun -np 4 ./bin/epoch2d <<< /absolute/path
```

where `/absolute/path` refers to the absolute path to the directory containing the `input.deck` file.

7.1 Uniform macro-particle load

The input deck given in Figure 9 runs a lightweight simulation to check if the code can load a uniform distribution of macro-particles. As mentioned in Section 3, the code assigns weights based on the volume of the macro-particle shape and not the cell volume, which prevents a sharp discontinuity in the first two cells in r.

```
begin:control
  nx = 50
  ny = 10
  t_end = 1 * femto
  x_min = 0
  x_max = 20.0e-6
  y_max = 5.0e-6
  stdout_frequency = 10
  n_mode = 2
  npart = 100000 * nx * ny
end:control

begin:boundaries
  bc_x_min = simple_laser
  bc_x_max = open
  bc_y_max = open
end:boundaries

begin:species
  name = Electron
  density = 1.0e28
  frac = 1.0
  identify:electron
end:species

begin:output
  name = normal
  dt_snapshot = 25 * femto
  grid = always
  number_density = always + species
end:output
```

Figure 9: Input deck for the uniform load test.

The input deck in Figure 9 was run six times with different compilation flags: all three macro-particle shapes were tested in simulations with and without `-DPER_SPECIES_WEIGHT` activated. The results from running these decks have been plotted in Figure 10.

In these simulations, a uniform density of macro-particles is loaded for most cells. On $x_{\min}$, $x_{\max}$ and $r_{\max}$, the weights drop as macro-particle weight is projected outside the simulation window, with nothing outside the window to replace the weights with. This results in the sharp dip observed at high r in Figure 10, and the dip in boundary x-cells causes the x-averaged densities to be slightly less than the 10^{28} m^{-3} target elsewhere. A dip is also present at $r = 0$ for the `-DPER_SPECIES_WEIGHT` due to inconsistencies between the volume calculated from (11), and the true volume due to $r = 0$ reflection and the macro-particle shape. Such a dip is not present in the particle-weighting case, as the volumes occupied by each macro-particle can be calculated before weights are assigned, so weight assignment can be performed to account for this.

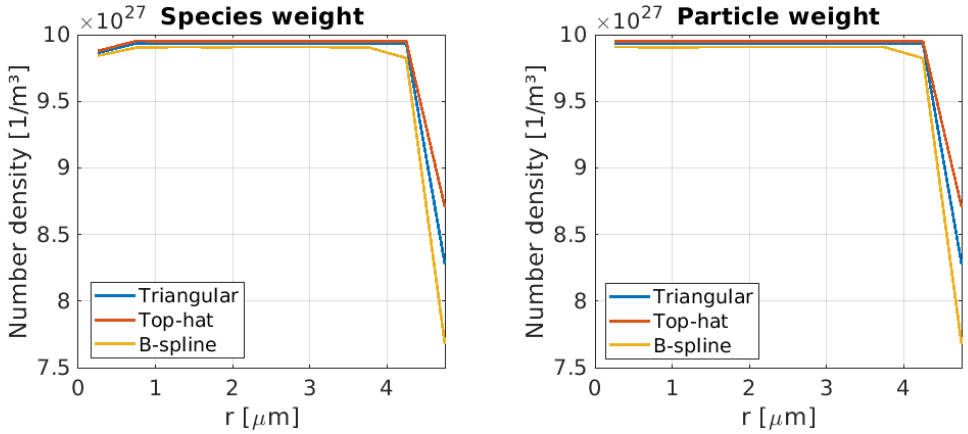

Figure 10: x-averaged number densities loaded from the Figure 9 input deck, when the code was compiled with different compilation flags. These include the three particle shapes, and in simulations where all macro-particles have the same weight (species-weight), or different weights (particle-weight).

7.2 Current density modes

In Section 7.1, we demonstrate how cylindrical EPOCH can be used to model a uniform density of macro-particles. If we also impose a drift to this system, we may test the current density solver. A test input deck has been given in Figure 11. Here, a uniform low-density plasma of electrons is initialised, with an extremely high drift momentum in x and y. We note that as drift refers to a momentum, this y describes the Cartesian y, and is not a stand-in for the radial direction. Macro-electrons with over 10^{20} kgms^{-1} momentum will be unaffected by any fields generated in this simulation, so these drifts should be maintained for the few simulation steps modelled here.

The analytic current density in this system is given by $-en_e v$. Hence, we expect $J_x \approx 3.40 \times 10^{-6}$ Am^{-2}, and $J_y \approx -3.40 \times 10^{-6}$ Am^{-2}. However, we note that cylindrical EPOCH current density is stored using cylindrical components, so we must find appropriate expressions here. Firstly, we may recall the conversion diagram in Figure 3, which shows that when $\theta = 0$, $\hat{\boldsymbol{r}}$ points along $\hat{\boldsymbol{y}}$, and $\hat{\boldsymbol{\theta}}$ points along $\hat{\boldsymbol{z}}$. Converting from current density modes to physical current densities is done using (1), which includes this θ factor. Hence, when combining modes with $\theta = \pi$, we expect $J_r \approx -3.40 \times 10^{-6}$ Am^{-2}, and $J_\theta \approx 0$, but when using $\theta = \pi/2$, we expect $J_r \approx 0$ and $J_\theta \approx -3.40 \times 10^{-6}$ Am^{-2}.

We can also deduce which modes will be important here. In Figure 5, we see the positive direction for modes above and below the $r = 0$ line. If we have uniform plasma motion, then modes which switch sign below $r = 0$ would be expected to cancel out when adding contributions across all θ for a given r. As a result, we expect $\hat{J}_x^m$ to have values for even m only, and $\hat{J}_r^m$, $\hat{J}_\theta^m$ to have values for odd m only. Hence, we need at least two modes to model the full currents in this system.

The input deck in Figure 11 has been run for triangular weighted macro-particles, without `-DPER_SPECIES_WEIGHT` active. The resulting heatmaps of current density have been plotted in Figure 12. The observed current densities largely agree with the expected values when viewed from different directions in θ, although a slight current enhancement is present near $r = 0$ for J_r. This is likely from the macro-particle shape reflection at $r = 0$.

```
begin:control
  nx = 500
  ny = 100
  t_end = 0.1 * femto
  x_min = 0
  x_max = 20.0e-6
  y_max = 5.0e-6
  stdout_frequency = 1
  n_mode = 2
  npart = 100 * nx * ny
  nprocx = 2
  nprocy = 2
end:control

begin:boundaries
  bc_x_min = simple_laser
  bc_x_max = open
  bc_y_max = open
end:boundaries

begin:species
  name = Electron
  density = 1.0e5
  frac = 1.0
  drift_x = -1.23e20
  drift_y = +1.23e20
  identify:electron
end:species

begin:output
  name = normal
  dt_snapshot = 25 * femto
  jxm = always
  jtm = always
  jrm = always
end:output
```

Figure 11: Input deck for the current density mode test.

7.3 Gaussian beam

The input deck in Figure 13 describes a focusing laser which follows a Gaussian beam profile. In this deck, the first constant block describes parameters which may be controlled by the user:

- `I_fwhm` - The full-width-at-half-maximum for the transverse intensity profile at the focused point.
- `I_peak_Wcm2` - The time-averaged intensity at the focal point.
- `las_lambda` - The wavelength of the laser pulse.
- `foc_dist` - The distance from the `x_min` boundary to the focal point position in x.

Again we note that the `y` parameter appearing in the `laser` block actually refers to the radial direction r.

Snapshots of E_y calculated from the Figure 13 input deck have been shown in Figure 14. Here, we have a peak $E_y \approx 8.6\times10^{10}$ V/m, which corresponds to a time-averaged intensity of 9.8×10^{14} Wcm^{-2}. The conversion between electric field E and time-averaged intensity I can be calculated according to

$$I = \frac{1}{2}\epsilon_0 c E^2 \tag{107}$$

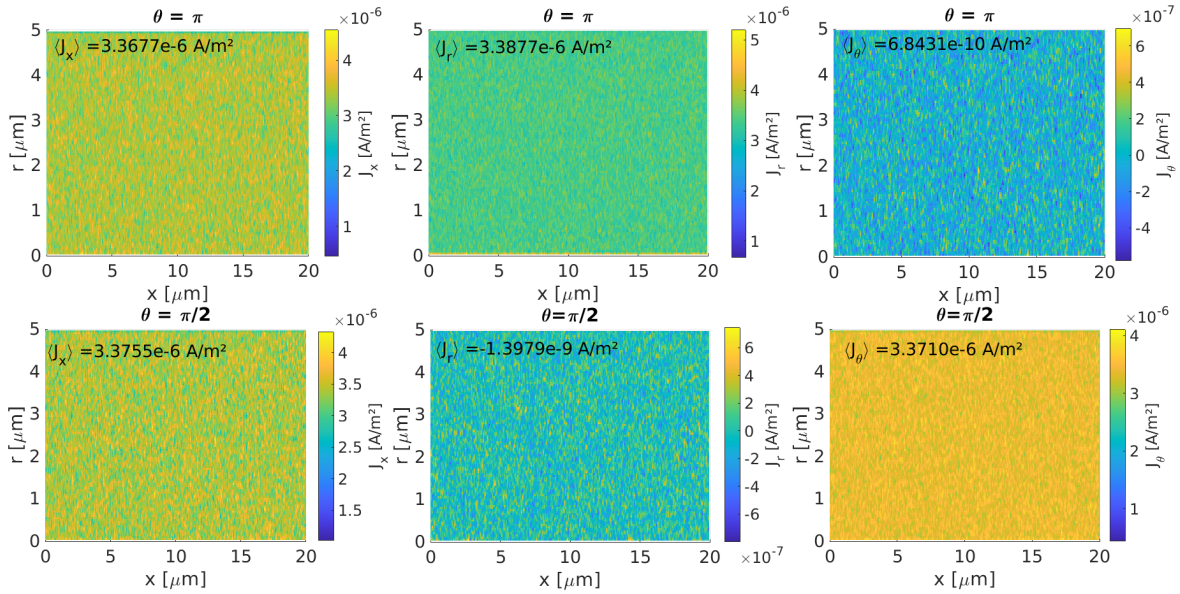

Figure 12: Current densities loaded from the `0001.sdf` file, obtained by running the Figure 11 input deck. Current density modes have been evaluated at different θ values, and the current density averaged across all cells is quoted.

where ϵ_0 is the permittivity of free space, and c is the speed of light.

The focal spot position occurs at $x \approx 9$ μm, and at this x point, E_y dips to $1/\sqrt{2}$ of the peak value at $r \approx 0.78$ μm. Thus, the full width at half maximum for the *intensity* profile is around 1.6 μm. Hence, the Gaussian beam propagates through the system as expected from the simulation setup.

```
begin:control
  nx = 500
  ny = 100
  t_end = 100 * femto
  x_min = 0
  x_max = 20.0e-6
  y_max = 5.0e-6
  stdout_frequency = 10
  n_mode = 2
end:control

begin:boundaries
  bc_x_min = simple_laser
  bc_x_max = open
  bc_y_max = open
end:boundaries

begin:constant
    I_fwhm = 1.5e-6          # FWHM of laser intensity
    I_peak_Wcm2 = 1.0e15     # 0.5 * eps0 * c * E_peak^2
    las_lambda = 1.0e-6      # Laser wavelength
    foc_dist = 10.0e-6       # Boundary to focal point distance
end:constant

begin:constant
    las_k = 2.0 * pi / las_lambda
    w0 = I_fwhm / sqrt(2.0 * loge(2.0))                  # Beam Waist
    ray_rang = pi * w0^2 / las_lambda                    # Rayleigh range
    w_boundary = w0 * sqrt(1.0 + (foc_dist/ray_rang)^2)  # Waist on boundary
    I_boundary = I_peak_Wcm2 * (w0 / w_boundary)^2       # Intens. on boundary
    rad_curve = foc_dist * (1.0 + (ray_rang/foc_dist)^2) # Boundary curv. rad.
    gouy = atan(-foc_dist/rad_curve)                     # Boundary Gouy shift
end:constant

begin:laser
    boundary = x_min
    intensity_w_cm2 = I_boundary
    lambda = las_lambda
    phase = las_k * y^2 / (2.0 * rad_curve) - gouy
    profile = gauss(y, 0, w_boundary)
end:laser

begin:output
  name = normal
  dt_snapshot = 25 * femto
  grid = always
  exm = always
  erm = always
  etm = always
  bxm = always
  brm = always
  btm = always
end:output
```

Figure 13: Input deck for the focusing laser, Gaussian beam test.

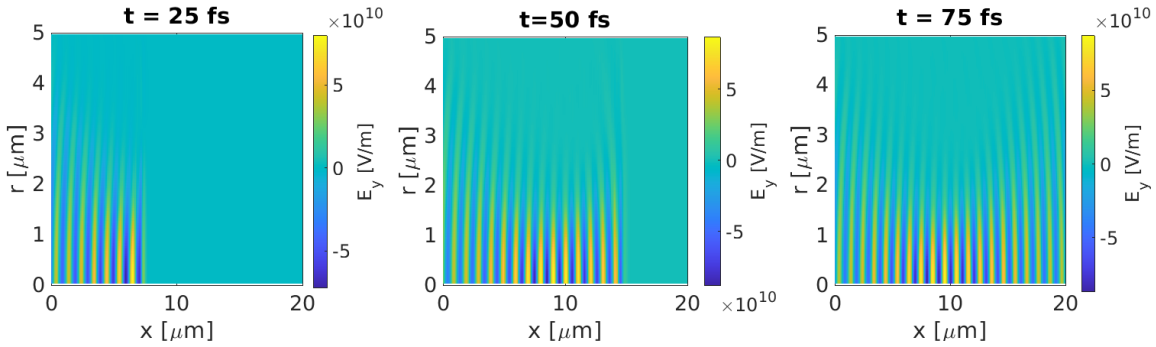

Figure 14: E_y snapshots from the Figure 13 Gaussian beam input deck. Here, E_y is deduced from $\hat{E}_r^m$ and $\hat{E}_\theta^m$ evaluated at $\theta = 0$ according to the field mode definition (1).

7.4 Two stream instability

One of the basic tests for benchmarking plasma behaviour in PIC codes is the two stream instability [5, 6, 7]. To summarise, when two electron plasma beams pass through each other, if the relative beam-speed is comparable to the phase-velocity of plasma oscillation, the beams may resonate and exchange energy. Any small perturbations grow exponentially, and the plasma behaviour changes. In noisy PIC codes, small perturbations are expected to be present, so even distributions initialised uniformly should generate this instability.

A two-stream instability set-up deck is present in Figure 15, and the resultant phase space plots are shown in Figure 16. This simulation took 6 minutes to run using 4 cores of a personal laptop. As can be seen, the simulation starts with the `Left` electron species moving leftwards (negative p_x), and the `Right` species moving rightwards (positive p_x). As the simulation progresses, energy is transferred between the beams, resulting in a mixing of momenta by 60 ms.

In order to get this simulation working, a few important keys have been included. To prevent loss of macro-particles on $r_{\max}$, we have imposed reflective boundary conditions for both species. The field boundary condition on $r_{\max}$ has been set to `zero_b`, as it was found that the code became numerically unstable when using the outflow boundary condition. Fields in the $m = 1$ mode grew to non-physical values, which also pushed the resulting momenta to a non-physical regime. This is why the `zero_b` boundary condition was developed.

```
begin:control
   nx = 400
   ny = 40
   x_min = 0
   x_max = 5.0e5
   y_max = 5.0e4
   t_end = 6.0e-2
   stdout_frequency = 100
   n_mode = 2
end:control

begin:boundaries
   bc_x_min = periodic
   bc_x_max = periodic
   bc_y_max = zero_b
end:boundaries

begin:constant
   drift_p = 2.5e-24
   temp = 273
   dens = 10
end:constant

begin:species
   name = Right
   charge = -1
   mass = 1.0
   temp = temp
   drift_x = drift_p
   number_density = dens
   npart = 4 * nx * ny
   bc_y_max = reflect
end:species

begin:species
   name = Left
   charge = -1
   mass = 1.0
   temp = temp
   drift_x = -drift_p
   number_density = dens
   npart = 4 * nx * ny
   bc_y_max = reflect
end:species

begin:output
   dt_snapshot = t_end / 100
   particles = always
   px = always
end:output
```

Figure 15: Input deck for the two-stream instability.

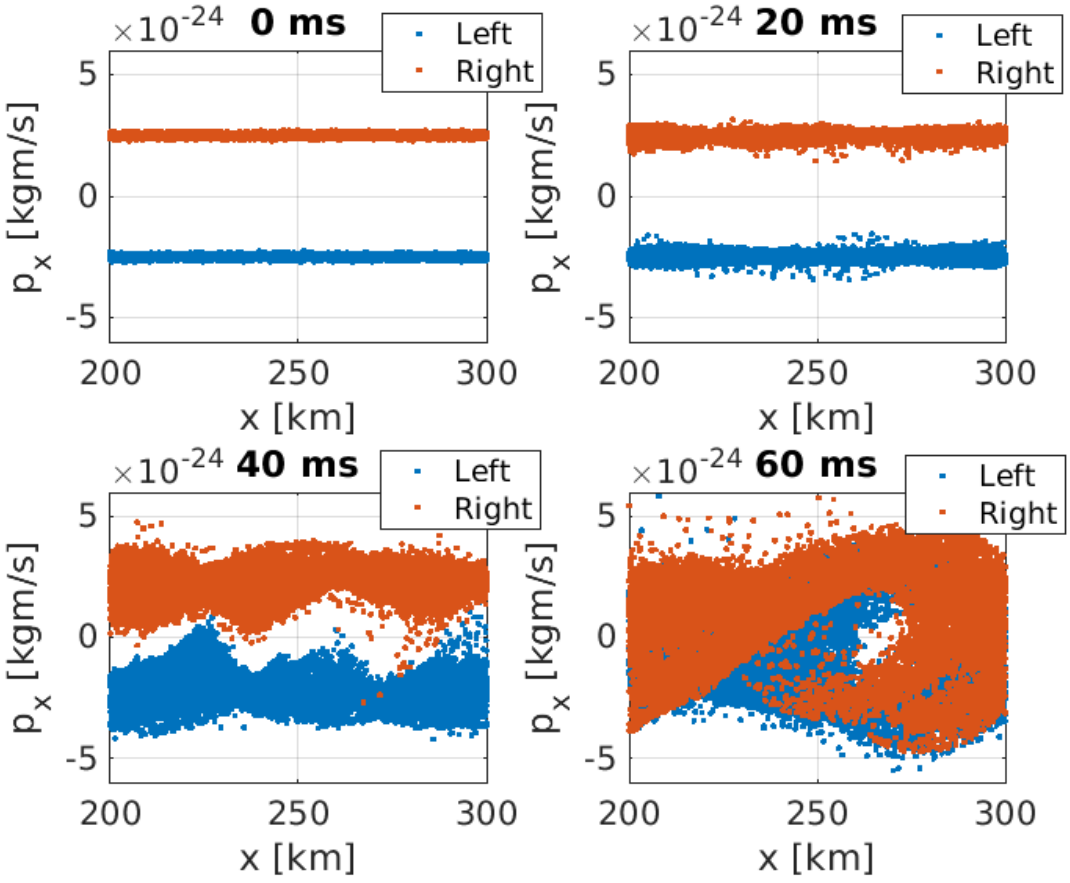

Figure 16: Snapshots of the (x, p_x) phase space from the two stream instability input deck shown in Figure 15. Each point represents a simulated macro-particle, and is coloured by the associated electron species.

7.5 Laser wakefield in a moving window

The most common application for a cylindrical PIC code is expected to be in laser-wakefield acceleration simulations. Here, a laser pulse passes through a long channel of plasma, driving a wake behind it. This was the original cylindrical PIC test performed by Lifschitz *et al*, in their Section 4.1 [2]. Using parameters from that section, we have created our own laser-wakefield input deck in Figure 17.

The laser is assumed to start at focus with an intensity of 3.4×10^{18} Wcm^{-2}, and with Gaussian spatial and temporal profiles with a radial FWHM of 9 μm, and a temporal FWHM of 30 fs. The peak laser intensity enters the simulation window at a time such that the temporal profile models the full-width-at-10%-maximum of the laser pulse. To match the large simulation domains described by Lifschitz *et al* [2], we have re-implemented a moving simulation window, as was present in the original EPOCH2D code.

The simulation took 6 hours and 32 minutes to run on a 6 core laptop. Six processors were chosen as it was found that further parallelisation slowed down the code. This is perhaps due to the small particle-per-cell count used in the simulation, as the improvement in particle push performance from domain splitting may have been balanced by the increased MPI overhead. A snapshot of the simulation has been illustrated in Figure 18.

This benchmark verifies a few aspects of the code. Firstly, we see the x domain in Figure 18 spans from 387 μm to 441 μm, which demonstrates the moving window capability of the cylindrical PIC code. We also see the field-modes evaluated on a 2D grid correspond to a 3D laser pulse, which continues to maintain the correct form after travelling for over 1 ps. Finally, the laser is correctly interacting with the background plasma, forming a wake behind the laser pulse.

```
begin:constant
  lambda0 = 0.8e-6
  cell_x = lambda0/25.0
  cell_r = lambda0/3.0
  t_fwhm = 30.0e-15
  r_fwhm = 9.0e-6
  t_fw01m = t_fwhm * 1.8226
  wt = t_fwhm * 0.60056
  wr = r_fwhm * 0.60065
end:constant

begin:control
  x_min = 0
  x_max = 68.0 * lambda0
  y_max = 40.0 * lambda0
  nx = (x_max - x_min) / cell_x
  ny = y_max / cell_r
  t_end = 2.9e-12
  stdout_frequency = 1
  n_mode = 2
end:control

begin:boundaries
  bc_x_min = simple_laser
  bc_x_max = open
  bc_y_max = open
end:boundaries

begin:window
  move_window = T
  window_v_x = 3.0e8
  window_start_time = 160.0e-15
  bc_x_min_after_move = simple_outflow
  bc_x_max_after_move = simple_outflow
end:window

begin:laser
  boundary = x_min
  intensity_w_cm2 = 3.4e18
  lambda = lambda0
  t_profile = gauss(time, 0.5*t_fw01m, wt)
  profile = gauss(y,0,wr)
end:laser

begin:species
  name = Electron
  charge = -1
  mass = 1
  npart = 16 * nx * ny
  density = 7.5e24
  bc_y_max = open
end:species
```

```
begin:species
  name = Proton
  charge = 1
  mass = 1836.2
  npart = 4 * nx * ny
  density = 7.5e24
  bc_y_max = open
end:species

begin:output
  name = normal
  dt_snapshot = t_end / 200
  grid = always
  exm = always
  erm = always
  etm = always
  number_density = always
end:output
```

Figure 17: Input deck for a laser-wakefield acceleration simulation with a moving window. This input deck spans two pages.

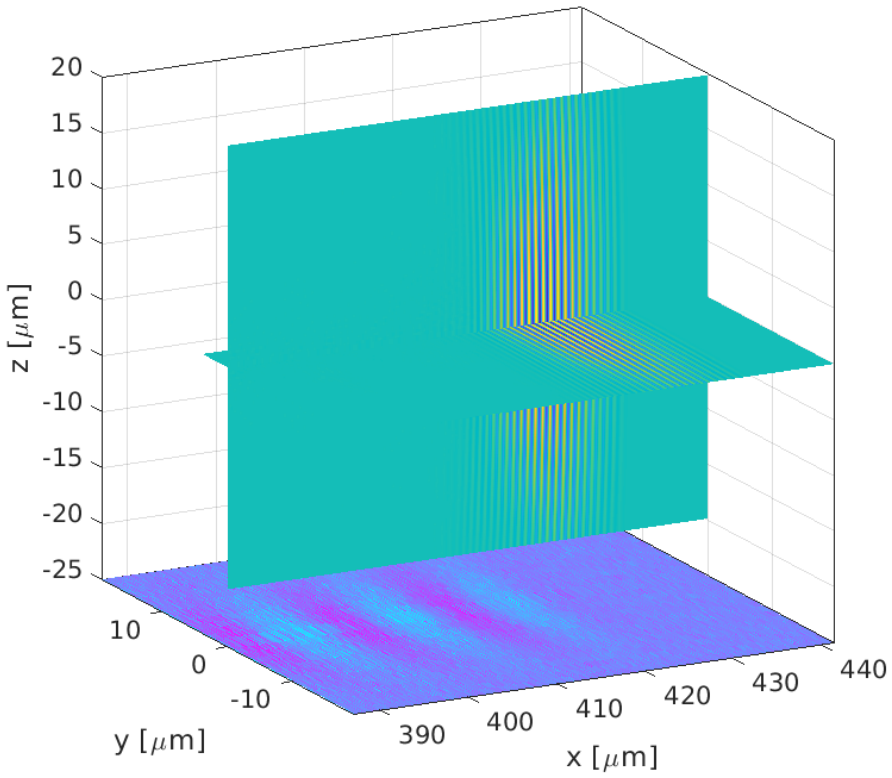

Figure 18: Visualisation of the laser wakefield simulation at $t = 1.45$ ps. E_y values have been calculated from electric field modes in two planes, for $y = 0$ and $z = 0$. The θ-integrated plasma number density has been included on the $z = -25$ μm plane.

7.6 Field ionisation

For laser-wakefield modellers, we anticipate the desire to include field ionisation in simulations - so this package has also been re-activated in the cylindrical EPOCH code. Field ionisation has been benchmarked as per the original test, first described by Nuter *et al* [8]. Here, we will attempt to recreate the simulation shown by Nuter *et al* in their Figure 2 [8].

The Nuter *et al* simulation shows how the relative populations of carbon charge-states change when a small, neutral carbon target is irradiated by a 3×10^{15} Wcm^{-2} laser pulse. Their simulation description has been converted into an equivalent cylindrical EPOCH simulation, where the resultant plot is given in Figure 19, and the generating input deck is quoted in Figure 20.

Through comparison between Figure 19 and Figure 2 of Nuter *et al* [8], it is clear that field ionisation in cylindrical EPOCH has been successfully re-activated. A direct comparison will show discrepancies, as the field-ionisation module used by EPOCH has a slightly different treatment than that of Nuter *et al*, as discussed by the original author of the EPOCH field ionisation package Lawrence-Douglas in his thesis [9].

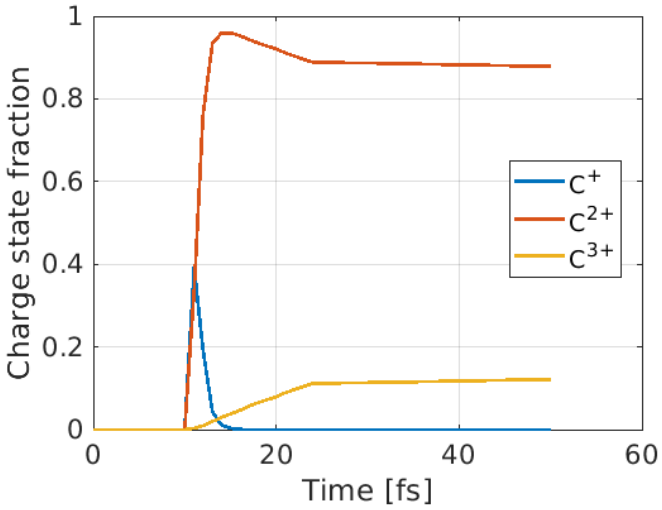

Figure 19: Relative proportions of the different carbon charge-states over time during irradiation by a 3×10^{15} Wcm^{-2} laser pulse. Simulation parameters taken from Nuter *et al* [8].

```
begin:control
  nx = 125
  ny = 200
  t_end = 50.0e-15
  x_min = 0
  x_max = 4.0e-6
  y_max = 6.4e-6
  npart = 1000 * (250.0e-9 / x_max) * nx * (125.0e-9 / y_max) * ny
  stdout_frequency = 10
  n_mode = 3
  field_ionisation = T
end:control

begin:boundaries
  bc_x_min = simple_laser
  bc_x_max = open
  bc_y_max = open
end:boundaries

begin:laser
  boundary = x_min
  intensity_w_cm2 = 3.0e15
  lambda = 0.8e-6
  t_profile = 1.0
  profile = 1.0
end:laser

begin:species
  name = Carbon
  charge = 0
  mass = 12.0 * 1836.2
  atomic_no = 6
  frac = 1
  density = if ( y lt 125e-9, 2.91e20, 0 )
  density = if ( x gt 3.05e-6 and x lt 3.3e-6, density(Carbon), 0)
  temperature = 0.0
  ionise = T
  ionise_limit = 3
  unique_electron_species = T
end:species

begin:output
  name = normal
  dt_snapshot = 1.0e-15
  weight = always
end:output
```

Figure 20: Input deck for the field ionisation benchmark from Figure 2 of Nuter *et al* [8].

7.7 Synchrotron radiation

The synchrotron radiation package has been re-activated for modelling gamma-rays produced due to electron acceleration in intense laser fields. The model was first discussed by Ridgers *et al* [10], and EPOCH implementation details can be found in the thesis of Morris [11].

To benchmark the code, a simulation was set up where a laser pulse counter-propagated with an electron bunch. The number of X-rays produced were counted, and a particle probe was introduced to ensure the photon-pusher subroutines worked.

The rate of photon production due to synchrotron radiation is given by

$$\frac{dN_\gamma}{dt} = \frac{\sqrt{3} m_e c^2}{h} \alpha \frac{\eta}{\gamma} h_s(\eta) \tag{108}$$

which uses the electron mass m_e, speed of light c, Planck's constant h, and the fine structure constant α. The function $h_s(\eta)$ is read from an EPOCH look-up table, γ is the electron beam Lorentz factor, and η is a dimensionless parameter given by

$$\eta^2 = \frac{1}{E_s^2} \left(\left| \gamma \boldsymbol{E} + \frac{1}{m_e c} (\boldsymbol{p} \times c\boldsymbol{B}) \right|^2 - \frac{1}{(m_e c)^2} (\boldsymbol{E} \cdot \boldsymbol{p})^2 \right) \tag{109}$$

for electron momentum p, where E_s is the Schwinger field.

Using these equations, we can set up a simulation where the laser intensity and electron beam momentum are chosen such that the rate of photon production can be set to a certain value. Once this rate is known, the number of photons emitted during the simulation can be estimated for a particular simulated run-time. The parameters of this input deck have been chosen such that the number of expected photons N_γ is $0.1 N_e$, the number of electrons.

```
begin:control
  nx = 300
  ny = 300
  nparticles = 100000
  t_end = 50.0e-15
  x_min = 0
  x_max = 15.0e-6
  y_max = 15.0e-6
  n_mode = 2
  stdout_frequency = 10
end:control

begin:qed
  use_qed = T
  qed_start_time = 0
  produce_photons = T
  produce_pairs = F
  photon_dynamics = T
end:qed

begin:boundaries
  bc_x_min = simple_laser
  bc_x_max = simple_laser
  bc_y_max = open
end:boundaries

begin:laser
  boundary = x_min
  intensity_w_cm2 = 3.0e16
  lambda = 1.0e-6
  t_profile = 1.0
```

```
  profile = 1.0
end:laser

begin:species
  name = Electron
  fraction = 1.0
  number_density = if ((x gt (x_max - dx)) and (y lt 10.0e-6), 6.4e21, 0)
  drift_x = -5.36e-18
  identify:electron
end:species

begin:species
  name = Photon
  nparticles = 0
  dump = T
  identify:photon
end:species

begin:output
  name = normal
  dt_snapshot = t_end / 10
  particle_weight = always
  particle_probes = always
end:output

begin:probe
   name = photon_probe
   point = (1.0e-6, 0.0, 0.0)
   normal = (-1.0, 0.0, 0.0)
   ek_min = 0.0
   ek_max = -1.0
   include_species:Photon
   dumpmask = always
end:probe
```

When summing the electron macro-particle weights in the first dump, and comparing this to the number of photons created when `photon_dynamics` is set to `F`, we find that $N_\gamma \approx 0.090 N_e$ - within 10% of the expected solution. When photon motion is permitted, and the code is compiled with `-DPROBE_TIME`, the photons arrive at the probe around 46.6 fs, which is consistent with travelling alongside the electrons as they move from 15 μm to 1 μm.

7.8 Bremsstrahlung radiation

Bremsstrahlung radiation has been re-activated for modelling X-rays produced when electrons accelerate in the electric fields of atom/ion nuclei. EPOCH implementation details can be found in the thesis of Morris [11].

Bremsstrahlung radiation can be described by the Seltzer-Berger cross-sections, σ [12] of a target material which has some number density n_i. The probability of a bremsstrahlung emission from an electron travelling distance Δx through such a material is

$$p = n_i \sigma \Delta x \tag{110}$$

In order to evaluate n_i at the macro-electron position, we can calculate the complex number density modes, and interpolate these modes over the macro-particle shape as is done for field modes in the particle push or synchrotron radiation subroutines. Number density field modes can be calculated by summing the number density contributions of individual macro-particles, and applying the conversion factors give in (89).

In order to test the successful implementation of a number-density-mode to macro-particle interpolation, let us set up a test problem with a known solution. The `input.deck` supplied in this section directs a beam of 10^5 electrons of energy 1 GeV through 5.5 mm of an Al target. The bremsstrahlung cross section for 1 GeV electrons in Al is 5.2×10^{-27} m^2, and the target number density is set to 6×10^{28}. Each macro-electron is expected to produce around 1.7 bremsstrahlung photons after passing 5.5 mm, so the full bunch should create 1.7×10^5.

```
begin:control
  nx = 128
  ny = 10
  npart = nx * ny * 128
  t_end = 2.0e-11
  x_min = 0
  x_max = 5.50e-3
  y_max = 5.0e-6
  n_mode = 2
  dt_multiplier = 0.8
  stdout_frequency = 10
end:control

begin:bremsstrahlung
  use_bremsstrahlung = T
  start_time = 0
  produce_photons = T
  photon_energy_min = 0
  photon_weight = 1.0
  photon_dynamics = F
end:bremsstrahlung

begin:boundaries
  bc_x_min = simple_laser
  bc_x_max = simple_laser
  bc_y_max = reflect
end:boundaries

begin:species
  name = Electron_Beam
  frac = 0.2
  rho = 1.0e5 / (pi * (0.5*y_max)^2 * dx)
  rho = if ((x lt dx) and (y lt 0.5*y_max), density(Electron_Beam), 0)
  drift_x = 5.344e-19
  identify:electron
end:species
```

```
begin:species
  name = Aluminium
  atomic_number = 13
  mass = 49218
  rho = 6.022e28
  frac = 0.8
  charge = 0
  immobile = T
end:species

begin:species
  name = Photon
  npart = 0
  identify:brem_photon
end:species

begin:output
  dt_snapshot = t_end
  particles = always
  px = always
  py = always
  pz = always
  number_density = always + species
  number_density_modes = always
  particle_weight = always
  particle_probes = always
end:output

begin:probe
  name = electron_probe
  point = (5.0e-3, 0.0, 0.0)
  normal = (1.0, 0.0, 0.0)
  include_species:Electron_Beam
  dumpmask = always
end:probe
```

The code was compiled with the `-DBREMSSTRAHLUNG` pre-compiler flag, all photons were added to the simulation window, and all were held immobile. Hence, by summing the macro-photon weights at the end of the simulation, we could determine how many photons were produced. Upon doing this, we did indeed find 1.7×10^5 photons were present, as expected. We conclude that bremsstrahlung radiation has been successfully re-activated in the code.

7.9 Collisions

PIC codes evaluate EM field components once per cell, and they do not capture the nuanced fields surrounding particles as they pass each other. To overcome this shortcoming, EPOCH utilises the collisions method of Pérez *et al* [13], which samples the expected cumulative scatter angle from a test particle after passing many background field particles. EPOCH supports two modes for collisions:

- Binary collisions: particles are paired together, and both are scattered.
- Fast-background collisions: particle species are classified as either "fast" or "background". The cell-averaged background properties are calculated, and the fast particle is scattered without affecting the background species.

Using the method outlined in Pèrez *et al* Section II.A [13], we were able to devise a test for the collisions algorithm. By passing a mono-energetic electron beam through an ion-slab, we can estimate the expected p_x momentum distribution of electrons as they emerge on the far-side post-collisions. This can be compared to the momentum distribution of marcro-particles in an equivalent EPOCH simulation to determine whether EPOCH reproduces the expected answer. This test was repeated twice for the two collisions modes. Both input decks are given here, and the resultant distributions are given in Figure 21.

For binary collisions:

```
begin:control
  nx = 600
  ny = 100
  t_end = 50e-15
  x_min = -2.0e-6
  x_max = 6.0e-6
  y_max = 1.0e-6
  n_mode = 2
  dt_multiplier = 0.8
  stdout_frequency = 10
end:control

begin:boundaries
  bc_x_min = open
  bc_x_max = open
  bc_y_max = reflect
end:boundaries

begin:collisions
   use_collisions = T
   use_nanbu = T
   coulomb_log = auto
   collide = all
   coll_n_step = 10
end:collisions

begin:species
  name = Electron
  npart = 0.1*nx*ny
  rho = if (x lt 0, 1.0e5, 0)
  drift_x = 1.0e6 * qe / c
  identify:electron
end:species

begin:species
  name = Carbon
  npart = 2 * nx * ny
  atomic_number = 6
  mass = 22034
```

```
  rho = if(x gt 0 and x lt 5.0e-6, 6.0e28, 0)
  charge = 6
  immobile = T
  temp_ev = 1.0e3
end:species

begin:output
  dt_snapshot = t_end
  particles = always
  particle_probes = always
end:output

begin:probe
  name = electron_probe
  point = (5.5e-6, 0.0, 0.0)
  normal = (1.0, 0.0, 0.0)
  include_species:Electron
  dumpmask = always
end:probe
```

and for background collisions:

```
begin:control
  nx = 600
  ny = 100
  t_end = 50e-15
  x_min = -2.0e-6
  x_max = 6.0e-6
  y_max = 1.0e-6
  n_mode = 2
  dt_multiplier = 0.8
  stdout_frequency = 10
end:control

begin:boundaries
  bc_x_min = open
  bc_x_max = open
  bc_y_max = reflect
end:boundaries

begin:collisions
   use_collisions = T
   use_nanbu = T
   coulomb_log = auto
   collide = none
   collide = Electron Carbon background
   coll_n_step = 10
   use_cold_correction = F
   rel_cutoff = 0.01
   back_update_dt = 5.0e-15
end:collisions

begin:species
  name = Electron
  npart = 0.1*nx*ny
  rho = if (x lt 0, 1.0e5, 0)
  drift_x = 1.0e6 * qe / c
  identify:electron
end:species

begin:species
  name = Carbon
  npart = 2 * nx * ny
```

```
  atomic_number = 6
  mass = 22034
  rho = if(x gt 0 and x lt 5.0e-6, 6.0e28, 0)
  charge = 6
  immobile = T
  temp_ev = 1.0e3
end:species

begin:output
  dt_snapshot = t_end
  particles = always
  particle_probes = always
end:output

begin:probe
  name = electron_probe
  point = (5.5e-6, 0.0, 0.0)
  normal = (1.0, 0.0, 0.0)
  include_species:Electron
  dumpmask = always
end:probe
```

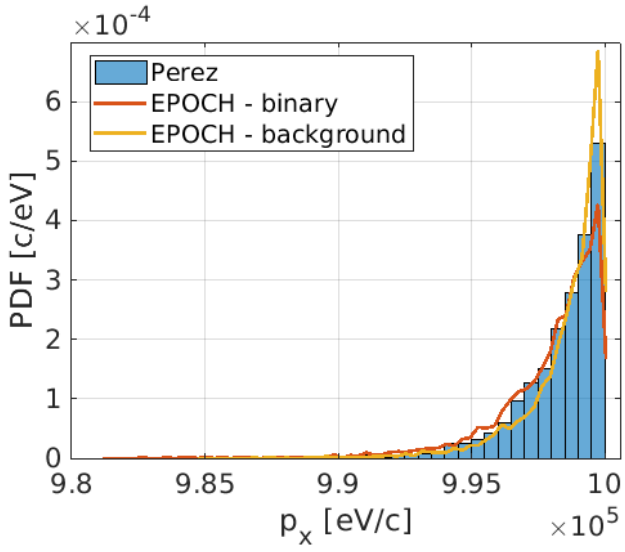

Figure 21: The p_x distribution of a mono-energetic beam of electrons passing a 5 μm slab of fully ionised carbon. The initial beam momentum was 1 MeV/c, and the target had a number density of 6×10^{28} m^{-3}, and a temperature of 1 keV. Two collision models in EPOCH were tested against an implementation of Pérez sampling [13].

As seen in Figure 21, both models show similar electron momentum distributions after passing the fully ionised carbon slab. Hence, it would appear that collisions have been successfully transferred to cylindrical EPOCH.

7.10 Collisional ionisation

The collisional ionisation model in EPOCH is described by Morris *et al* [14]. In this physics package, electrons are paired with atoms or partially-ionised ions, and the probability of electron-impact ionisation is calculated. In Section IV [14], a method is provided to benchmark the correct implementation of collisional ionisation.

If an electron bunch is directed through a target slab of length L with electron-impact-ionisation cross section σ, then the expected number of ions produced, N_i^+ is

$$N_i^+ = N_e \sigma n_i L \tag{111}$$

where N_e is the number of electrons travelling through the slab, and n_i is the number density of target particles within the slab. From Figure 7 of Morris *et al* [14], we expect a 9 keV electron beam passing through Xe to have a collisional ionisation cross section of around 2×10^{-21} m^2. A simulation was performed to model this, and N_i^+ was counted in order to deduce σ. The inferred σ from simulation matched the expected σ, which demonstrated collisional ionisation sampling was successful. The input deck for this test has been provided below.

```
begin:control
  nx = 1200
  ny = 100
  t_end = 200e-15
  x_min = -2.0e-6
  x_max = 10.0e-6
  y_max = 1.0e-6
  n_mode = 2
  dt_multiplier = 0.8
  stdout_frequency = 10
end:control

begin:boundaries
  bc_x_min = open
  bc_x_max = open
  bc_y_max = reflect
end:boundaries

begin:collisions
  collisional_ionisation = T
  ci_n_step = 100
  subsample_ci = T
  subsample_ci_number = 1.0e4
end:collisions

begin:species
  name = Electron
  npart = 0.1*nx*ny
  rho = if (x lt 0, 3.2e20, 0)
  drift_x = 5.4e-23
  identify:electron
end:species

begin:species
  name = Xenon
  npart = 2 * nx * ny
  atomic_number = 54
  mass = 238789
  rho = if(x gt 0, 2.7e25, 0)
  charge = 0
  immobile = T
  ionise = T
  ionise_limit = 1
```

```
  ionisation_electron_species = Electron
end:species

begin:output
  dt_snapshot = t_end
  particles = always
  px = always
  py = always
  pz = always
  particle_weight = always
end:output
```

References

[1] H. Ruhl, Classical particle simulations with the psc code, Ruhr-Universität Bochum (2005).

[2] A. F. Lifschitz, X. Davoine, E. Lefebvre, J. Faure, C. Rechatin, V. Malka, Particle-in-cell modelling of laser–plasma interaction using fourier decomposition, Journal of Computational Physics 228 (5) (2009) 1803–1814.

[3] J. Derouillat, A. Beck, F. Pérez, T. Vinci, M. Chiaramello, A. Grassi, M. Flé, G. Bouchard, I. Plotnikov, N. Aunai, et al., Smilei: A collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation, Computer Physics Communications 222 (2018) 351–373.

[4] T. Z. Esirkepov, Exact charge conservation scheme for particle-in-cell simulation with an arbitrary form-factor, Computer Physics Communications 135 (2) (2001) 144–153.

[5] W. Kruer, The physics of laser plasma interactions, crc Press, 2019.

[6] K. S. Thorne, R. D. Blandford, Modern classical physics: optics, fluids, plasmas, elasticity, relativity, and statistical physics, Princeton University Press, 2017.

[7] D. Anderson, R. Fedele, M. Lisak, A tutorial presentation of the two stream instability and landau damping, American Journal of Physics 69 (12) (2001) 1262–1266.

[8] R. Nuter, L. Gremillet, E. Lefebvre, A. Lévy, T. Ceccotti, P. Martin, Field ionization model implemented in particle in cell code and applied to laser-accelerated carbon ions, Physics of Plasmas 18 (3) (2011).

[9] A. Lawrence-Douglas, Ionisation effects for laser-plasma interactions by particle-in-cell code, Ph.D. thesis, University of Warwick (2013).

[10] C. P. Ridgers, J. G. Kirk, R. Duclous, T. Blackburn, C. S. Brady, K. Bennett, T. Arber, A. Bell, Modelling gamma-ray photon emission and pair production in high-intensity laser–matter interactions, Journal of computational physics 260 (2014) 273–285.

[11] S. J. Morris, High energy x-ray production in laser-solid interactions, Ph.D. thesis, University of York (2021).

[12] S. M. Seltzer, M. J. Berger, Bremsstrahlung energy spectra from electrons with kinetic energy 1 kev–10 gev incident on screened nuclei and orbital electrons of neutral atoms with z= 1–100, Atomic data and nuclear data tables 35 (3) (1986) 345–418.

[13] F. Pérez, L. Gremillet, A. Decoster, M. Drouin, E. Lefebvre, Improved modeling of relativistic collisions and collisional ionization in particle-in-cell codes, Physics of Plasmas 19 (8) (2012).

[14] S. Morris, T. Goffrey, K. Bennett, T. Arber, Improvements to collisional ionization models for particle-in-cell codes, Physics of Plasmas 29 (12) (2022).