UNIVERSITY OF BIRMINGHAM

EMPIRE-PIC

Bettencourt, Matthew T.; Brown, Dominic A.S.; Cartwright, Keith; Cyr, Eric; Glusa, Christian; Lin, Paul; Moore, Stan; McGregor, Duncan; Pawlowski, Roger ; Phillips, Edward; Roberts, Nathan; Wright, Steven A.; Maheswaran, Satheesh; Jones, John P.; Jarvis, Stephen

Document Version
Peer reviewed version

[Link to publication on Research at Birmingham portal](#)

Higher-Order Particle Representation for Particle-in-Cell Simulations[⋆]

Dominic A. S. Brown[a,∗], Matthew T. Bettencourt[b], Steven A. Wright[c], Satheesh Maheswaran[d], John P. Jones[e], Stephen A. Jarvis[f]

[a]*Department of Computer Science, University of Warwick, UK*
[b]*Sandia National Laboratories, Albuquerque, NM*
[c]*Department of Computer Science, University of York, UK*
[d]*Diamond Light Source Ltd, Diamond House, Harwell Science and Innovation Campus, Didcot, UK*
[e]*Atomic Weapons Establishment, Aldermaston, UK*
[f]*College of Engineering and Physical Sciences, University of Birmingham, UK*

Abstract

In this paper we present an alternative approach to the representation of simulation particles for unstructured electrostatic and electromagnetic PIC simulations. In our modified PIC algorithm we represent particles as having a smooth shape function limited by some specified finite radius, r_0. A unique feature of our approach is the representation of this shape by surrounding simulation particles with a set of virtual particles with delta shape, with fixed offsets and weights derived from Gaussian quadrature rules and the value of r_0. As the virtual particles are purely computational, they provide the additional benefit of increasing the arithmetic intensity of traditionally memory bound particle kernels. The modified algorithm is implemented within Sandia National Laboratories' unstructured EMPIRE-PIC code, for electrostatic and electromagnetic simulations, using periodic boundary conditions. We show results for a representative set of benchmark problems, including electron orbit, a transverse electromagnetic wave propagating through a plasma, numerical heating, and a plasma slab expansion. Good error reduction across all of the chosen problems is achieved as the particles are made progressively smoother, with the optimal particle radius appearing to be problem-dependent.

Keywords: Particle-in-Cell, high-order, unstructured, particle representation, shape function

1. Introduction

The behaviour of plasmas within various environments and conditions is studied extensively within the scientific community. In particular, there is significant interest in the field of fusion energy research, which seeks to realise fusion power via Inertial Confinement Fusion (ICF) or Magnetic Confinement Fusion (MCF) devices. Examples of such devices include the National Ignition Facility (NIF), located at Lawrence Livermore National Laboratory (LLNL), and the International Thermonuclear Experimental Reactor (ITER), located in France, which attempt ICF and MCF, respectively. Another major interest is the area of pulsed power systems and magnetically insulated transmission lines (MITL). The Z Pulsed Power Facility, otherwise known as the 'Z machine', located at Sandia National Laboratories (SNL) is one notable example of such a system. Additional fields of study include the behaviour of magnetrons in microwave generation systems, charged particle beams, laser-plasma interaction [1], astrophysical plasmas [2], and applications in biomedicine [3]. However, conducting such experiments can be both extremely time consuming and/or prohibitively expensive, leading researchers to use simulation to model such phenomena on computer systems. The Particle-in-Cell (PIC) method is a common approach used to carry out such simulations [4, 5].

Electrostatic and electromagnetic PIC methods are commonly used to simulate high power devices, and the behaviour of plasmas under various physical conditions. Traditionally, PIC algorithms employ structured computational

---

grids – representing the electric and magnetic fields on a staggered Yee grid [6] – and model particles as discrete Lagrangian points moving through the problem space [5, 7]. Notable examples of structured PIC codes include the Extendable PIC Open Collaboration (EPOCH) [1], OSIRIS [8], the Plasma Simulation Code (PSC) [9] and VPIC [10, 11]. Gyrokinetic PIC algorithms have also been applied to the challenge of plasma simulation in five-dimensional phase space, where rapid movement about the magnetic field lines allows a velocity dimension to be ignored in the simulation. One such code is GTC-P, the Gyrokinetic Toroidal Code developed at Princeton University [12]. The performance of the code at scale has been demonstrated on a number of notable HPC systems, including Sequoia, Piz Daint, Titan and Tianhe-2 [13, 14].

However, traditional structured meshes are poorly suited to representing problems that make use of high fidelity geometries, where they typically exhibit, at best, first-order convergence. There are multiple approaches to resolving this complication. One such method is proposed by Dey and Mittra [15], which adapts the commonly used Finite Difference Time Domain (FDTD) algorithm to use locally distorted cells that accurately model a curved geometry with simple changes to the FDTD scheme. Another conformal scheme proposed by Zagorodnov et al. [16] also models curved boundaries and does not require time-step sizes that are significantly smaller than a staircasing approach. Additionally, the algorithm results in convergence between first- and second-order, depending on the problem. Other notable examples that maintain the use of a structured mesh include the application of cut-cell algorithms which 'cut' bodies out of the background mesh [17], and Adaptive Mesh Refinement (AMR) methods that refine the mesh only in high interest areas [18]. The use of AMR-PIC has previously been explored for electrostatic and electromagnetic problems by Vay et al. [19, 20].

Alternatively, one can achieve high geometric flexibility through the use of fully unstructured meshes, which avoid the requirement of an impractically high grid resolution imposed by the structured approach. Like AMR, this provides the flexibility of refining the problem in areas of key interest, but without the restriction that the grid cells themselves retain their structured properties. Examples of such PIC codes include PTetra [21] and the open-source Spacecraft Plasma Interaction Software (SPIS) [22].

In addition to unstructured or adaptively refined meshes, many domain scientists have also experimented with the use of higher-order methods. While these have previously been seen as prohibitively computationally intensive, the extreme levels of parallelism offered by modern supercomputers is causing a revival of such methods. This is due to the increased arithmetic intensity of these methods improving the amount of floating point operations (FLOPs) performed per byte moved from RAM, providing an advantage in situations where limited memory capacity and bandwidth poses an obstacle to performance. The additional computational cost is also accompanied with improved simulation accuracy and convergence. Such methods have the benefit of enabling the use of coarser computational grids and reduced simulation constraints, while still reaching an acceptable solution due to the increased accuracy that they can provide. However, higher-order methods also require smoother particle shape functions in order to achieve the desired convergence, as in a higher-order Galerkin finite element scheme there is an assumption of smoothness in the source terms. The use of smooth particles also increases simulation accuracy by improving the sampling of the surrounding fields when interpolating from the mesh to the particles, and by reducing the effects of aliasing as particles move between elements.

Structured PIC codes generally implement higher-order PIC by using smooth particle shapes extending over multiple cells [1], combined with higher-order field solvers. One example of such a particle shape can be achieved by implementing the cloud-in-cell (CIC) representation proposed by Birdsall and Fuss [23]. Unfortunately, such methods are non-trivial to implement in practice for unstructured PIC codes as evaluating a higher-order basis often becomes intractable when spanning multiple elements.

Jacobs and Hesthaven present a discontinuous Galerkin PIC method that incorporates both higher-order time domain solution of Maxwell's equations and smooth particle shapes [24, 25]. Essex and Edwards also show a higher-order PIC algorithm, HOPIC, that extends the PIC method to fourth-order accuracy for transport problems [26].

Stindl et al. have also investigated higher-order methods within an electromagnetic discontinuous Galerkin PIC code, with a particular focus on the coupling of the particles and the unstructured grid [27]. Specifically, the authors compare first- and third-order B-spline interpolation functions to a reference Cell Mean Value (CMV) approach which distributes the charge of all particles in a cell equally to all cell nodes.

In this paper we propose modifications to the core PIC algorithm by representing particles as having a smooth quadratic shape, with compact support on a fixed radius, which is numerically integrated against the test function representing the weak form of the currents or charge densities. This integration is performed by numerical cubature

where the cubature points are represented by virtual particles surrounding each super-particle. Each virtual particle has an associated offset and weight derived from Gaussian quadrature rules and the chosen radius. This approach also has the advantage of requiring little extension to the core PIC kernels. This builds on the work of Pinto et al. [28] where representation of smooth particle shapes using numerical quadrature rules was proposed. Specifically we consider the implementation of a specific shape function in a production PIC code using this method, and examine its effects on the solution to various benchmark problems.

The algorithm has been implemented within Sandia National Laboratories' unstructured EMPIRE-PIC[1] code for electrostatic and electromagnetic problems, in both two and three dimensions using periodic boundary conditions. The effect of the algorithm on simulation solutions is explored using four representative benchmark problems. This greatly extends our previous work [29] where we presented only initial exploratory computational performance results for two-dimensional electrostatic problems, where simulation error was not considered. The work presented in this paper includes both electrostatic and electromagnetic simulations, considers both two- and three-dimensional particle shapes, and quantifies the increased accuracy gained through the use of the proposed particle representation.

While smooth particle shapes have been explored previously by other authors, the use of virtual particles is a unique feature of our implementation. This differs from the approach used by Jacobs and Hesthaven [24] where particles in a discontinuous Galerkin PIC code are represented as a cloud of constant size with particles weighted to all elements within the cloud radius. We instead examine weighting each virtual particle to/from its associated element in a continuous Galerkin code. The work presented in this paper also differs from the charge-conserving PIC scheme proposed by Squire et al. [30] as a Delaunay triangular grid is not required – our method can be applied to arbitrary unstructured meshes. Similarly to Moon et al. [31], in electromagnetic simulations the fields are expanded using the Whitney basis functions. Specifically, the Whitney 1-forms are used for the electric field, and the 2-forms for the magnetic field.

The virtual particle approach also has the advantage of being able to tune the offsets and weights of the virtual particles to reproduce a given shape function for the particle cloud with relative ease. Finally, as the virtual particles are computational, we obtain the additional benefit of adding increased arithmetic intensity to traditionally memory bound particle kernels within the PIC method.

In summary, we make the following contributions:

- We propose representing particles in the unstructured PIC algorithm as having a smooth shape that is limited by some finite radius;

- This shape is represented as a collection of delta shape virtual particles surrounding each delta shape super-particle in order to effectively give the super-particle a smoother shape. The virtual particles have fixed offsets and weights obtained from Gaussian quadrature rules and the chosen radius;

- The algorithm is implemented in SNL's unstructured EMPIRE-PIC code, for both electrostatic and electromagnetic problems with periodic boundary conditions;

- We compare the accuracy of the proposed algorithm to the base implementation of EMPIRE-PIC using four representative benchmark problems.

The remainder of this paper is structured as follows: Section 2 provides a summary of the PIC algorithm; Section 3 introduces our new higher-order particle shape representation and algorithm modifications, and how the core PIC components can be adapted to implement it; Section 4 examines the accuracy and convergence of the method on a set of benchmark problems; finally, in Section 5 we conclude the paper, and highlight areas of interest for future research.

2. The Particle-in-Cell Method

The PIC method is a commonly used technique to simulate the motion of charged particles, in which particles are tracked in a Lagrangian manner on an Eulerian mesh that represents the problem domain. The particles move freely through the domain, and the mesh is used to calculate fields and approximate interactions between particles. While the

---

[1] A reference paper detailing the capabilities and specifications of EMPIRE-PIC is currently under review at another journal.

charge and current density are calculated from the particles, these values are needed on the grid, requiring deposition to the grid at each step. Although the procedure has been detailed extensively in other works [4, 5, 31], we reiterate relevant parts of it here for completeness. The method applies an operator split approach, which can essentially be thought of as two coupled solvers, where one updates the values of the electric and magnetic fields, and the other updates the particle positions and velocities. This is typically accomplished via leapfrog integration, where variables are updated at interleaved points in time such that they 'leapfrog' over one another.

The time evolution of electric and magnetic fields is governed by Maxwell's equations, given below. Note that here we use the $\vec{E}/\vec{B}$ formulation of Maxwell's equations instead of the $\vec{D}/\vec{H}$ version. This provides the advantages of being easier to couple to the Klimontovich equation [32, 33], allowing the use of a compatible discretisation, and facilitating the strong preservation of the magnetic divergence constraint, (2). A further benefit is that the $\vec{E}$ and $\vec{B}$ terms are present in the Lorentz force law, meaning that our simulation fields can be used directly to compute this force.

$$\nabla \cdot \vec{E} = \frac{\rho}{\epsilon_0} \tag{1}$$

$$\nabla \cdot \vec{B} = 0 \tag{2}$$

$$\frac{\partial \vec{B}}{\partial t} = -\nabla \times \vec{E} \tag{3}$$

$$\frac{\partial \vec{E}}{\partial t} = \frac{1}{\mu_0 \epsilon_0} \nabla \times \vec{B} - \frac{1}{\epsilon_0} \vec{J} \tag{4}$$

Where $\vec{E}$ and $\vec{B}$ are the electric and magnetic fields, ρ is the charge density, $\vec{J}$ is the current, and μ_0 and ϵ_0 are the permeability and permittivity of free space, respectively. The force felt by charged particles in the presence of these fields is defined by the Lorentz force law. This can then be applied to update the velocity $\vec{v}$ and position $\vec{x}$ of the particles, resulting in the equations of motion shown below, assuming a particle mass m and charge q.

$$\frac{d\vec{v}}{dt} = \frac{q}{m} \left(\vec{E} + \vec{v} \times \vec{B} \right) \tag{5}$$

$$\frac{d\vec{x}}{dt} = \vec{v} \tag{6}$$

This update is typically handled through the use of the particle pusher proposed by Boris [34]. Note that the particles in a PIC simulation are not individual physical particles, as this would be computationally infeasible. Instead, super-particles that represent the phase space density are used. These super-particles follow the same equations of motion as their physical counterparts due to possessing the same charge-to-mass ratio.

The particles and fields (represented by Maxwell's equations) can then be assembled into the Klimontovich equation for plasma dynamics, which can be used to fully describe the time evolution of the system [32, 33].

In general, PIC simulations execute a loop that is made up of four key steps, and repeated for each simulation time-step. The four steps are as follows:

1. Solving Maxwell's equations to update the electric and/or magnetic fields.
2. Calculate the value of the electric and magnetic fields at each particle, by interpolating these values from the mesh.
3. Accelerating and moving the particles.
4. Interpolating charge/current contributions from the particles back to the mesh.

These steps are common to PIC codes in general, but production applications often merge these in various ways, e.g., current deposition is sometimes carried out during the particle move step. One should also note that the steps listed here are not exhaustive – diagnostic collection and particle injection can also be part of the main simulation loop.

2.1. Updating the Fields

Beginning with the field update, we must solve Maxwell's equations ((1)-(4)) for the new values of $\vec{E}$ and $\vec{B}$. Typically, the electric and magnetic fields are advanced in time using an FDTD method on a structured rectilinear

grid [6]. However, in the unstructured PIC code considered in this paper we instead employ a finite element method (FEM) to solve for the fields at each time-step [35], using edge- and face-based elements for the electric and magnetic fields, respectively, thus matching the Yee FDTD method.

A key point is that for electrostatic problems we need only solve Gauss' law (1), while for electromagnetic simulations we instead consider Ampère's and Faraday's laws ((3)-(4)). One should also note that the magnetic divergence constraint (2) is implicitly maintained in the FEM formulation from first principles due to the use of a compatible discretisation, meaning that we do not need to enforce it directly during the solve step. The equations can then be put into their respective weak forms [35] and integrated by parts in order to form mass and/or stiffness matrices, allowing them to be solved for the updated fields via various iterative or direct numerical methods.

2.2. Weighting Fields to Particles

During a PIC simulation the values of the fields are known only on specific locations inside the spatial grid, i.e., the interpolation points of the finite elements. Therefore, in order to correctly advance the particles in time during the simulation we must determine the values of the electric and magnetic fields at the precise location of each particle via interpolation. Given that we know the values of both fields at some arbitrary time-step n, the fields can be evaluated at a particular particle position $\vec{x}_i$ as follows, where $\hat{e}$ are the Nédélec edge elements [36], $\hat{b}$ are the Raviart-Thomas [37] face elements, and N_{edge} and N_{face} are the number of edges and faces of the containing element, respectively. This interpolation can be carried out from either the raw edge-/face-based fields, or from values that have been projected to the mesh nodes as is sometimes done in FDTD-PIC schemes.

$$\vec{E}(\vec{x}_i) = \sum_{j=0}^{N_{edge}} E_j \hat{e}_j(\vec{x}_i) \tag{7}$$

$$\vec{B}(\vec{x}_i) = \sum_{j=0}^{N_{face}} B_j \hat{b}_j(\vec{x}_i) \tag{8}$$

2.3. Particle Acceleration and Movement

In the simulation, the locations of the computational particles must also be advanced in time along with the fields by updating their individual velocities and positions. This step is commonly known as the 'particle push'. A detailed comparison of various particle movers has been conducted by other authors [38].

We update the particle velocities via solving for the force on the particles due to the electric and magnetic fields as defined by the updates given in (5) and (6), and the couplings given in (7)–(8). To determine the new velocity we employ the well known Boris method, which has become the de facto standard for pushing particles in PIC codes [34]. Once the new velocity is obtained it is then trivial to update the particle position.

2.4. Weighting of Particles to Grid

Particles are required to deposit their charge and/or current contribution back to the spatial grid via interpolation, prior to the beginning of the next field solve. For an electrostatic simulation it is sufficient for the particles to deposit charge contributions at the end of the particle move to the nodes of their newly containing element. This coupling is defined below, where $\hat{v}_i$ represents the nodal basis functions, j is the element index, Ω is the element volume, and N_P is the number of particles. As each computational particle represents multiple physical particles, we define the particle weight W such that W_k is the number of physical particles represented by the kth computational particle.

$$\int_{\Omega_j} \rho_j \hat{v}_i \, dV = \sum_{k=1}^{N_P} W_k q_k \hat{v}_i(\vec{x}_k) \tag{9}$$

Electromagnetics require a different interpolation scheme in order to deposit current, this commonly occurs during the particle move step and can be evaluated at the midpoint, $x_k^{n+1/2}$, as shown in (10), where $\vec{u}$ is the particle velocity. The particle trajectory must also be split such that each element visited by a particle in a given step has the correct

(a) Circular virtual particle layout using 3 quadrature points.

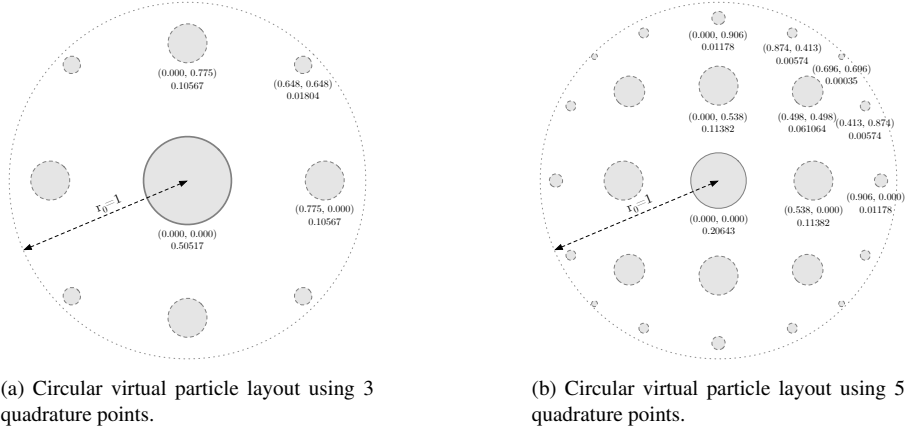(b) Circular virtual particle layout using 5 quadrature points.

Figure 1: Image showing two example virtual particle layouts of differing orders. Surrounding virtual particles are grey with dashed borders, with the physical location of the simulation particle represented by the central virtual particle (solid border). Virtual particles are sized proportionally to their weights.

proportion of current accrued to it. Note that here we use the edge basis, $\hat{e}$, instead of the nodal basis, and that each edge carries all components of the current.

$$\int_{\Omega_j} \vec{J}_j^{n+1/2} \hat{e}_i \, dV = \sum_{k=1}^{N_P} \int_{\Omega j} \int_{n\Delta t}^{(n+1)\Delta t} W_k q_k \vec{u}_k \cdot \hat{e}_i \, dt \, dV = \sum_{k=1}^{N_P} W_k q_k \vec{u}_k \cdot \hat{e}_i \left(\vec{x}_k^{n+1/2} \right) \tag{10}$$

Note that this method is only charge conserving for simplices. For non-simplex elements the basis functions used in EMPIRE-PIC are non-linear within the elements, so a higher-order integration is required – in this case we use two-point Gaussian quadrature using points at $\left(1 \pm 1/\sqrt{3}\right)/2$, each with a weight of $1/2$. These quadrature points are the standard Gaussian quadrature points with the interval adjusted to $[0, 1]$. This reduces to the charge conservation method presented by Villasenor and Buneman [39].

3. Higher-Order Particle Shapes

Let us first consider the electrostatic formulation of the standard PIC algorithm due to its simplicity, as the scheme can be later expanded to electromagnetics. As discussed previously we must formulate the weak form of Gauss' Law such that we can integrate the electric potential with a given test function and generate a stiffness matrix. Solving Gauss' Law also requires the charge density ρ to be computed from the computational particles. These particles are generally represented as shape functions S in space and velocity. In the standard FEM-PIC algorithm, the shape function is generally the Dirac delta function, δ. The charge density can then be integrated with the test function. One should note that integrating δ with the linear nodal basis is equivalent to piecewise linear interpolation in FDTD-PIC. This results in a summation at the particle locations when $S = \delta$.

$$\int_{\Omega_j} \rho_j \hat{v}_i \, dV = \sum_{k=1}^{N_P} \int_{\Omega_j} S \left(\vec{x} - \vec{x}_k\right) q_k \hat{v}_i \, dV = \sum_{k=1}^{N_P} W_k q_k \hat{v}_i(\vec{x}_k) \tag{11}$$

This simple integration is valid, independent of the order of the test function. However, the use of the Dirac delta function results in a particle shape that is not a smooth representation due to its nature. In the following section, we show how δ can be replaced with a smooth shape function, and how this can be implemented through the use of virtual particles.

6

Point	Position x_i	Weight w_i
0	$-\sqrt{\frac{3}{5}}$	$\frac{5}{9}$
1	0	$\frac{8}{9}$
2	$\sqrt{\frac{3}{5}}$	$\frac{5}{9}$

Table 1: Positions and weights for three-point Gaussian quadrature.

3.1. Smooth Particle Shape Function

In order to solve the problem of a non-smooth particle shape we now propose representing particles as having some defined fixed size. Specifically, we assume that particles possess some radius r_0, and have a parabolic shape subject to the following shape function – replacing the usual Dirac delta function. In (12) we also have normalisation constant $c = 2/\pi r_0^2$.

$$S(\vec{x} - \vec{x}_0) = \begin{cases} c\left[1 - \left(\frac{r}{r_0}\right)^2\right] & \text{if } r \leq r_0 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

The exact integral of the shape function with the test function in two dimensions is given below in (13). Equation (14) shows how this can extended to handle a three-dimensional case. However, integration of this shape function with the test function is generally computationally intractable when spanning more than a single element. We therefore handle the integration of this function via the application of Gaussian numerical quadrature.

$$\int_0^{r_0} \int_0^{2\pi} S(\vec{x} - \vec{x}_0)\, \hat{v}_i \, d\theta \, dr = \int_0^{r_0} \int_0^{2\pi} rc\left[1 - \left(\frac{r}{r_0}\right)^2\right] \hat{v}_i(r, \theta) \, d\theta \, dr \tag{13}$$

$$\int_0^{r_0} \int_0^{2\pi} \int_0^{\pi} S(\vec{x} - \vec{x}_0)\, \hat{v}_i \, d\phi \, d\theta \, dr = \int_0^{r_0} \int_0^{2\pi} \int_0^{\pi} rc\left[1 - \left(\frac{r}{r_0}\right)^2\right] \hat{v}_i(r, \theta, \phi) \, d\phi \, d\theta \, dr \tag{14}$$

3.2. Implementation

The smooth particle shapes described above are implemented by taking a given simulation particle, and surrounding it with a set of computational virtual particles. This allows one to move the quadrature weights from the mesh onto the virtual particles themselves. In this representation the particle radius is fixed independently of the size of its current element, and the central particle is used to track the physical location of the particle in the simulation space. The virtual particles represent quadrature points for the particle; each has a fixed associated position offset $\vec{o}_v$ and weight factor w_v, where the weight incorporates Gaussian quadrature weights and the shape function. It should be noted that the sum over the set of virtual particle weights must be equal to one to ensure the correct total contribution once all virtual particles are processed. Example particles represented in this way are shown in Figure 1. While this particular arrangement could give rise to azimuthal modes due to mesh imprinting, we do not believe this to be a significant issue in this paper. One method of addressing this biasing could be randomly rotating the shape of each loaded particle, thus reducing the bias towards the coordinate axes.

We now derive virtual particle weights and offsets using a shape represented by mapping a square to a circular shape. For 3D problems we instead map a cube to a sphere. The choice of a circular/spherical shape has certain benefits. Firstly, this symmetrical shape prevents the grid biasing/mesh imprinting that would occur with the use of a square/cube layout. Secondly, such a shape captures the notion of the Debye sphere [40, 41] and allows for a better representation of this concept in a simulation.

Given Gaussian quadrature of an arbitrary order, let r_0 be the chosen particle radius, and x and y be the positions of the Gaussian quadrature points. We can now calculate x' and y' which together make up the offset for the virtual particle being mapped. Additionally, let w_x and w_y be the weights of these points and $|\mathbf{J}|$ be the determinant of the Jacobian for the mapping at these points, which we include in order to correctly map from the reference volume to the mapped volume. For convenience, Table 1 shows the positions and weights for three-point Gaussian quadrature.

We can now calculate x' and y' which together make up the offset for each virtual particle being mapped, and we can also determine the values of w_v. Each permutation (with repetition) of the Gaussian quadrature points used

maps to a single virtual particle; in the case of three-point Gaussian quadrature this results in a total of $3^2 = 9$ virtual particles. Equation (15) shows the mapping used for 2D problems, and (16) shows the weight calculation. As a final step, the weights must be normalised to sum to one.

$$\vec{o}_v = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x\sqrt{1 - \frac{y^2}{2}} \\ y\sqrt{1 - \frac{x^2}{2}} \end{bmatrix} \tag{15}$$

$$w_v = w_x w_y \left(1 - \left(x'^2 + y'^2\right)\right)|\mathbf{J}| \tag{16}$$

$$|\mathbf{J}| = \left\| \begin{bmatrix} \frac{\delta x'}{\delta x} & \frac{\delta x'}{\delta y} \\ \frac{\delta y'}{\delta x} & \frac{\delta y'}{\delta y} \end{bmatrix} \right\| \tag{17}$$

For 3D problems we additionally define z, z', and w_z and carry out the mapping as shown below in (18) and (19). Each permutation (with repetition) of the Gaussian quadrature points continues to map to a single virtual particle, which for three-point quadrature results in a total of $3^3 = 27$ virtual particles. As before, the weights are normalised to sum to one.

$$\vec{o}_v = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x\sqrt{1 - \frac{y^2}{2} - \frac{z^2}{2} + \frac{y^2 z^2}{3}} \\ y\sqrt{1 - \frac{z^2}{2} - \frac{x^2}{2} + \frac{x^2 z^2}{3}} \\ z\sqrt{1 - \frac{x^2}{2} - \frac{y^2}{2} + \frac{x^2 y^2}{3}} \end{bmatrix} \tag{18}$$

$$w_v = w_x w_y w_z \left(1 - \left(x'^2 + y'^2 + z'^2\right)\right)|\mathbf{J}| \tag{19}$$

$$|\mathbf{J}| = \left\| \begin{bmatrix} \frac{\delta x'}{\delta x} & \frac{\delta x'}{\delta y} & \frac{\delta x'}{\delta z} \\ \frac{\delta y'}{\delta x} & \frac{\delta y'}{\delta y} & \frac{\delta y'}{\delta z} \\ \frac{\delta z'}{\delta x} & \frac{\delta z'}{\delta y} & \frac{\delta z'}{\delta z} \end{bmatrix} \right\| \tag{20}$$

While the shape function of a particle is usually represented by a delta function when this is put into the weak form it has an action on all the bases of the element it occupies thus being equivalent to using piecewise linear shape functions in FDTD-PIC. As described above the delta function is extended to a quadratic shape with compact support on the specified radius r_0 which is numerically integrated against the test function representing the weak form of the currents or charge densities. As long as it is guaranteed that the weights sum to unity then the properties of charge conservation will continue to be maintained. Errors in the cubature can effectively be thought as deviations to the shape function $S(r) = c(1 - r^2/r_0^2 + \epsilon f(r))$ where ϵ is the cubature error. Figure 2 shows the absolute error between the exact integral of the shape function with the basis function and of the integration using the proposed method. We see that for a single element ϵ converges to on the order of machine precision – the increase in noise at higher numbers of quadrature points can be explained by accumulated floating-point errors. However, much slower convergence is observed once a particle spans multiple elements. This is unsurprising as we do not expect convergence for the polynomial integration of a class C^0 function. As long as ϵ is below the statistical convergence rate of $\sqrt{(1/N)}$ this error is expected to be small when compared to other terms.

Given the offsets and weights defined above, the implementation of the PIC algorithm can now be modified to leverage this new particle shape. As the offsets and weights are shared by all virtual particles, the additional memory required to store this extra data is minimal. Assuming a three-dimensional case and double precision floating-point numbers, this results in an additional $8N_v$ bytes for the weights and $24N_v$ additional bytes for the offsets, where N_v represents the number of virtual particles used per simulation particle. In the case of five-point Gaussian quadrature this amounts to approximately four kilobytes. One should note that the positions of the virtual particles do not need to be stored, it is sufficient to track the physical particle location and apply the assigned offset. The extension to virtual particles only changes the coupling between the particles and the mesh, making the extensions to the particle move trivial. The modifications made to the PIC algorithm are now described in the subsequent sections.
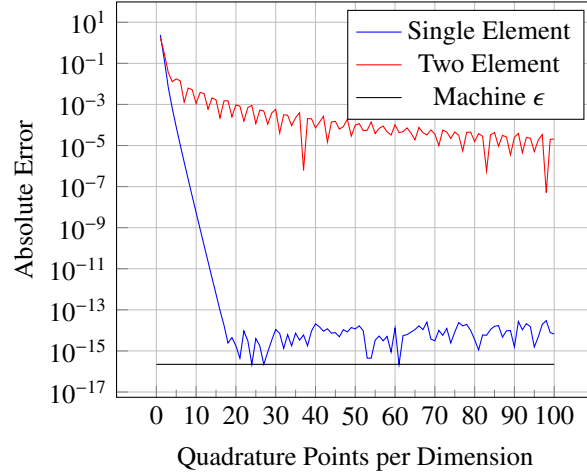
8

Figure 2: Graph showing how the absolute error of the modified integration converges with the number of quadrature points for a single element, and when spanning two elements.

3.2.1. Weighting Fields to Particles

As the electric and magnetic fields are only known on the computational mesh, they must be interpolated from the mesh to the particles in order to be able to update the particle forces and velocities. PIC usually accomplishes this through the use of basis functions to determine the field values at specific particle locations. In our algorithm we use the same approach to calculate these values at the position of the replicated virtual particles, and multiplying the field value by the virtual particle's associated weight. This can be expressed mathematically as shown in (21) and (22), where $\vec{x}_i$ is the physical position of particle i, and N_v continues to represent the number of virtual particles used per simulation particle. As before, w_i represents the weighting of virtual particle i as specified in (16) and (19) for two-dimensional and three-dimensional problems, respectively. Once this has been done for all virtual particles it is then trivial to accumulate each individual contribution to the central particle via summation.

$$\vec{E}(\vec{x}_i) = \sum_{v=0}^{N_v} \sum_{j=0}^{N_{edge}} E_j \hat{e}_j(\vec{x}_i + \vec{o}_v) w_v \tag{21}$$

$$\vec{B}(\vec{x}_i) = \sum_{v=0}^{N_v} \sum_{j=0}^{N_{face}} B_j \hat{b}_j(\vec{x}_i + \vec{o}_v) w_v \tag{22}$$

One should note that it is not necessary to carry out a neighbour search in order to determine which edges/faces/nodes will be interpolated from as the containing element of each virtual particle is updated during the particle move step. This discussed further in Section 3.2.2.

3.2.2. Particle Acceleration and Movement

Implementing a particle mover using the proposed modifications to the particle shapes as described is a relatively simple matter. This is due to the offset of each virtual particle used being fixed relative to the position of the central particle that is used to track the physical location. As in the standard PIC algorithm we apply the typical Boris Pusher in order to update the velocities of the central particles in the simulation. Additionally, we specify that the surrounding virtual particles share the same velocity as their associated central particle, meaning that they do not need to be processed during the acceleration step. The central particle position can then be updated as in the standard algorithm. Finally, we apply the same position update to the virtual particles, advancing them in lock-step with their associated central particle. This allows us to track the containing element of each virtual particle, thus removing the need for neighbour searches when interpolating values to and from the spatial grid. This choice results in storing a single 32-bit integer per virtual particle per simulation particle, i.e., $4N_P N_v$ extra bytes in total.

9

3.2.3. Weighting of Particles to Grid

As described in Section 2, the particles are coupled to the grid and must therefore make contributions back to the grid prior to the field solve that will take place at the beginning of the next time-step. This can be thought of as each constituent virtual particle making its own separate charge or current contribution, scaled by its pre-calculated weight factor. These couplings take place as defined in (9) and (10) for charge and current, respectively. The implementation of the charge weighting for electrostatic problems using the extension to virtual particles is simple. As the virtual particle weights sum to 1, the total amount of charge deposited will remain unchanged. We define this modified coupling below, using the same notation as defined previously.

$$\int_{\Omega_j} \rho_j \hat{v}_i \, dV = \sum_{k=1}^{N_P} W_k q_k \sum_{v=1}^{N_v} w_v \hat{v}_i \left(\vec{x}_k + \vec{o}_v \right) \tag{23}$$

A similar approach to that employed above can also be applied to the current weighting procedure with the difference that each virtual particle will make a contribution during its individual move, instead of all virtual particles making a deposit at the end of the move step. Additionally, deposits will be made to all elements crossed by the virtual particle during the move step. Specifically, the trajectory of each virtual particle is individually split as it passes through each element, which is crucial for a charge-conserving current deposition scheme. The particle to grid coupling for current deposition using virtual particles is given below in (24). This remains analogous to each virtual particle making a separate weight-scaled current contribution to the grid. Along with the base code, we continue to use two-point Gaussian quadrature in the case of non-simplex elements. As the virtual particle weights sum to one and our base implementation conserves charge, this current deposition is also charge conserving.

$$\int_{\Omega_j} \vec{J}_j \hat{e}_i \, dV = \sum_{k=1}^{N_P} \Delta t W_k q_k \sum_{v=1}^{N_v} w_v \vec{u}_k \cdot \hat{e}_i \left(\vec{x}_k^{n+1/2} + \vec{o}_v \right) \tag{24}$$

4. Results

In the following section we present results for four numerical experiments. These have been selected to be broadly representative of the problems that can be solved with EMPIRE-PIC. First, a simple 2D electrostatic electron orbit problem is examined. Second, the 3D simulation of a transverse electromagnetic (TEM) wave propagating through plasma is discussed. Third, we analyse the effect of our higher-order particle shapes on the amount of numerical heating observed. Finally, we look at a more complex electrostatic problem – the 1D expansion of a neutral plasma slab into a vacuum. For the results collected in the following experiments we used a virtual particle layout as defined in Section 3. We used 5-point Gaussian quadrature resulting in 25 virtual particles for the 2D problems, and 125 virtual particles for the 3D problem. For the electrostatic problems we also examine the effects of particle smoothing when second-order basis functions are used. This analysis was not conducted for electromagnetics as higher-order basis functions are not currently available in EMPIRE-PIC for electromagnetic problems.

4.1. 2D Orbit Problem

We first consider the behaviour of our algorithm on a very basic electrostatic problem, consisting of a stationary H^+ ion being orbited by a single electron for one period. This electron is treated as a test particle that does not make charge contributions back to the spatial grid. Using this simple test case we examine the effect of varying particle radius on the accuracy of the tracking of basic particle motion. The particles are situated on a quadrilateral mesh, the ion positioned at the centre, and the electron has an orbit radius of $r_{orbit} = 5.291 \times 10^{-8}$ m. The length of the domain in both x and y directions is equal to $3.0 \times r_{orbit}$, with initial $N_x = N_y = 14$, resulting in 196 elements in total. We also specify the problem boundary conditions to an analytical value defined as the exact value of the potential at the boundary: $\phi = \frac{q}{2\pi\epsilon_0} \ln \left(r^{-1} \right)$, using an ion charge of $q = +e$, where e is the elementary charge.

The initial conditions of the problem can be derived as follows. Given the electrostatic assumption, we can reduce the Lorentz force to $\vec{F} = q\vec{E}$. Then, from centripetal force and Gauss' Law we can write the following to obtain an
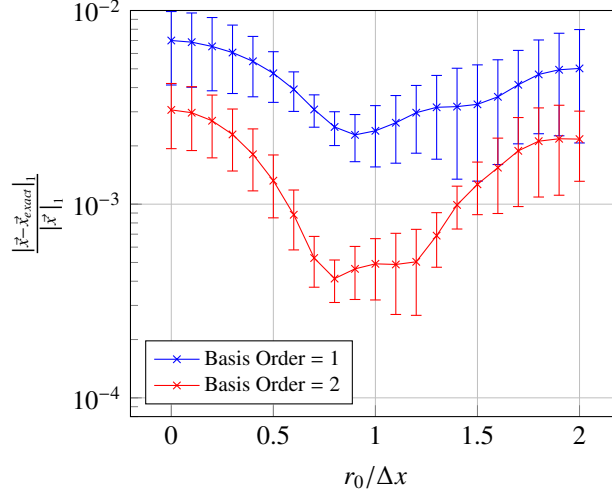
Figure 3: Graphs showing results for the 2D electron orbit experiments on the structured mesh. Error bars represent one standard deviation in the L_1 norm due to variation in the starting locations.

expression for the electric field:

$$qE(r) = m\omega^2 r \tag{25}$$

$$\int \vec{E} \cdot \hat{n}\, dA = \int_V \frac{\rho}{\epsilon} \tag{26}$$

We are using a test electron that does not deposit charge to the mesh in order to simplify the boundary conditions, due to being treated as having zero charge, but finite charge-to-mass ratio. Therefore, as the fields are not changed, the above can be simplified. We can now rewrite and substitute (25) and (26) in order to derive an expression for the angular velocity ω, and also velocity v which can then be resolved into its x and y components.

$$E(r) = \frac{1}{2\pi r}\frac{q}{\epsilon_0} \tag{27}$$

$$\omega^2 = \frac{q^2}{2\pi r^2 m \epsilon_0} \tag{28}$$

$$v = \omega r \tag{29}$$

Therefore we can define angular velocity $\omega = \sqrt{q^2/2\pi r^2 m \epsilon_0}$. With $x = r_{orbit}\cos(\omega t)$ and $y = r_{orbit}\sin(\omega t)$ it is now trivial to compare the simulated orbit to every point on the trajectory defined by the analytical solution.

For the base case of this test we place the hydrogen ion at the centre of the mesh, directly on top of an element vertex. In order to avoid the special case (a particle will almost never occupy this position in an actual problem), we repeat the test placing the central particle at 100 randomised positions within the element quadrant. As a result of all cell quadrants being identical, we can obtain data that consider a representative range of possible particle positions within an element. In the remainder of this section we examine the effects of increasing particle radius on the L_1 error of the position of the orbiting electron against the analytical solution (normalised via the orbit radius), and also consider these effects at increased levels of problem refinement, where we hold the ratio $\Delta x/\Delta t$ fixed in order to maintain a constant CFL value.

We first examine the effects of increasing particle radius for the base level of mesh refinement, consisting of 14 elements in both dimensions as defined above. In order to definitively rule out the influence of time integration on the orbit error due to large time-step size, we present data collected using a refined Δt to ensure that the improvement due to smoothing is visible. In this case, we use 320 time-steps per electron orbit. Figure 3 shows how the L_1 error varies as particle radius, r_0, is increased over various fractions of the cell size Δx. The error bars are used to represent the

standard deviation in the error due to the position of the hydrogen ion in the element quadrant. After the initial radii, it is clear to see that as the particle radius is increased the computed answer moves closer to the analytical solution, with a radius value of $0.9 \times \Delta x$ appearing to be optimal in this case.

We also observe a significant reduction in the standard deviation due to altering the position of the central H^+ ion within its quadrant, i.e., the level of statistical noise is lower as when using smooth particles the difference in the force felt at the centre of the element versus at the element vertex is lower than in the base code. This results in less variation in the result due to altering the position of the central ion. However, this improvement in error and statistical variation is reversed as particle radius continues to increase beyond the cell size, as using a large radius essentially means that a different problem is being solved. From this we can conclude that some smoothing of the charge distribution of the particle improves the ability of the PIC algorithm to track basic particle motion, whereas excessive smoothing results in reduced benefits.

Using second-order basis functions results in a 50% improvement over the base code in terms of error and standard deviation when no smoothing is applied. When smoothing is applied the results follow a similar trend to the first-order basis data, with an optimal radius value of $0.8 \times \Delta x$. This error value is significantly lower than the equivalent data point for the first-order basis. In the best case the electron position error is approximately 85% lower than the base code, and the standard deviation in the result is reduced by an order of magnitude. This is comparable to using a factor of 25 more simulation particles, assuming scaling of $1/\sqrt{N_P}$. This shift in compute intensity versus memory footprint is the main benefit in this test case – a key motivation for the work presented in this paper.

We now examine the effect of mesh refinement on this problem via a convergence study, with a base level of $\Delta x = 1.13 \times 10^{-8}$ m, $\Delta t = 4.615 \times 10^{-11}$ s and $N_x = N_y = 14$ grid elements. This results in a CFL condition value of $\vec{v}\Delta t/\Delta x \approx 0.0004675$. Figure 4 shows the results of this study for various particle radii, using the L_1 norm of the electron position as the error metric, as in the previous test. For clarity, we additionally include a reference line demonstrating theoretical second-order convergence. As seen in Figure 4a, a particle radius of approximately $0.9 \times \Delta x$ appears to be optimal for the majority of refinement levels used for the convergence study when considering the first-order basis. Figure 4b shows that when using a second-order basis the optimal radius remains consistent as the problem is refined. These results show a consistent improvement in the L_1 norm across a wide range of Δx values, consistent with the previous results for the coarse mesh. The error reduction appears to be approximately a stable factor of 2 when comparing results for the vanilla code against runs using the optimal radius value for the first-order basis, and a factor of 5 for the second-order basis. It is also evident from these results that, for a fixed field, the use of smoother particles causes earlier solution convergence then the standard FEM-PIC algorithm, indicating that this may provide acceptable convergence rates while allowing the use of coarser meshes which are less computationally expensive.

Finally, in order to assess the benefits of particle smoothing for non-regular grids, we carried out an additional parameter scan over $r_0/\Delta x$ using an unstructured mesh of 2272 triangular elements. This mesh and its dimensions are shown in Figure 5a, with the base orbit trajectory shown in red. The mesh has an average Δx value of approximately 7×10^{-9} m, calculated as $\Delta x = \sqrt{V/N_{elem}}$, where V is the volume of the mesh, and N_{elem} is the number of elements. As with the previous experiments, data was collected for each input using 100 randomised starting locations, this time varying the starting position by at most $\pm 0.5 \times r_{orbit}$ in each dimension. In this way we can determine the variation in the result due to the electron travelling through various levels of mesh distortion. We continue to use a refined time-step of 320 steps per orbit in order to rule out time integration error. Figure 5b shows the results of this experiment, with error bars again representing one standard deviation in the error due to the variation in orbit position. As in the radius scan experiment that was conducted for the structured mesh we again see a smooth reduction in average error as particle radius is increased. In this case we have an optimal value of $r_0/\Delta x = 1.3$, suggesting that greater amounts of smoothing may be beneficial on a distorted mesh. However, due to the high level of mesh distortion the improvements in the standard deviation are less significant. A similar trend is observed for the second-order basis where improvements are visible, but more pronounced than for the first test. Using the second-order basis causes the optimal amount of smoothing to become similar to the results in Figure 3. In the best case both the electron position error and standard deviation are reduced by an order of magnitude. From these results it is clear to see that the altered algorithm is capable of coping with such varying distortion, particularly when using a second-order basis.
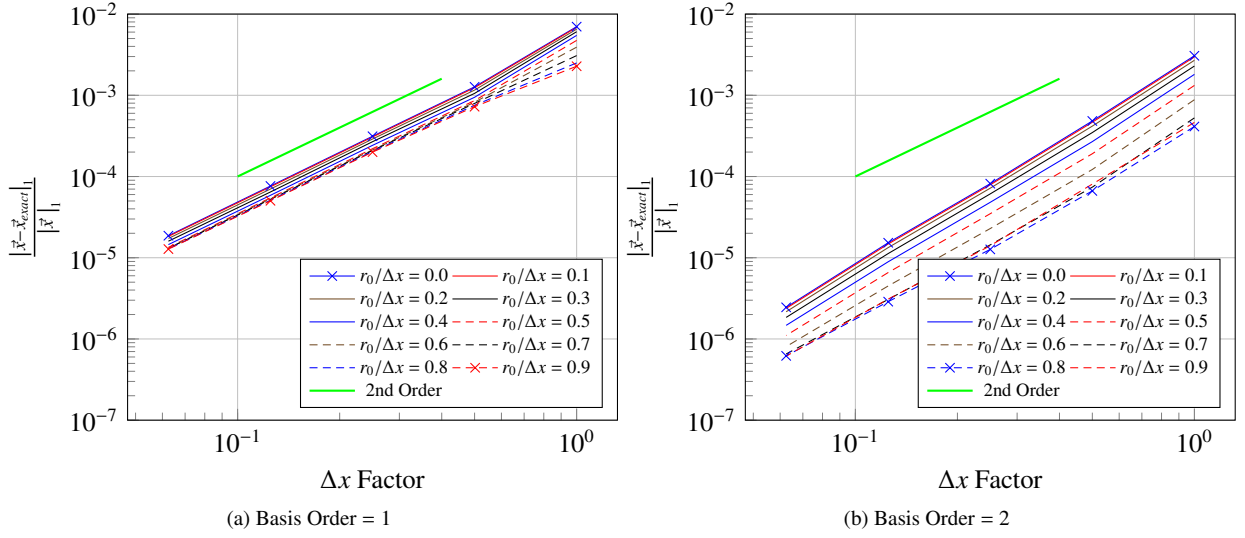
12

Figure 4: Convergence study results for the 2D orbit problem, $\vec{v}\frac{\Delta t}{\Delta x} = 0.0004675$

4.2. 3D Transverse Electromagnetic Wave Problem

To test the performance of our algorithm for 3D and electromagnetic problems we now consider an infinite, planar TEM wave propagating through an infinite neutral plasma made up of H^+ ions and electrons. This problem was chosen as an electromagnetic case study and has an analytical solution, given certain assumptions. The solution is given in Section 4.12 of Chen [40], which derives the differences between a TEM wave in a vacuum and a TEM wave in a plasma where the wave vector is held constant.

In this problem we choose the key controlling parameters as follows: we have the plasma number density as $n_0 = 10^{15}$ m^{-3}, initial temperature of 0 K, with a maximum electric field magnitude of $E_{mag} = 100$ V/m, and the vacuum frequency is the frequency of the hydrogen line, i.e., $f_v \approx 1.420$ GHz, and $\omega_v = 2\pi f_v$. The kinetic energy of the wave follows a sine-squared pattern, with maxima and minima that increase slowly over time due to numerical heating effects, which can be reduced with smaller time-step sizes and grid spacing, and higher numbers of simulation particles. We discuss numerical heating effects in greater detail in Section 4.3.

Next, we assume that the electromagnetic wave is of such a high frequency that the ions within the plasma remain stationary throughout the simulation, and also that the $\vec{J} \times \vec{B}$ forces on the particles are negligible. This has the effect that electrons are assumed to only oscillate linearly in the plane of the electric field. We have the plasma frequency and actual wave frequency as follows:
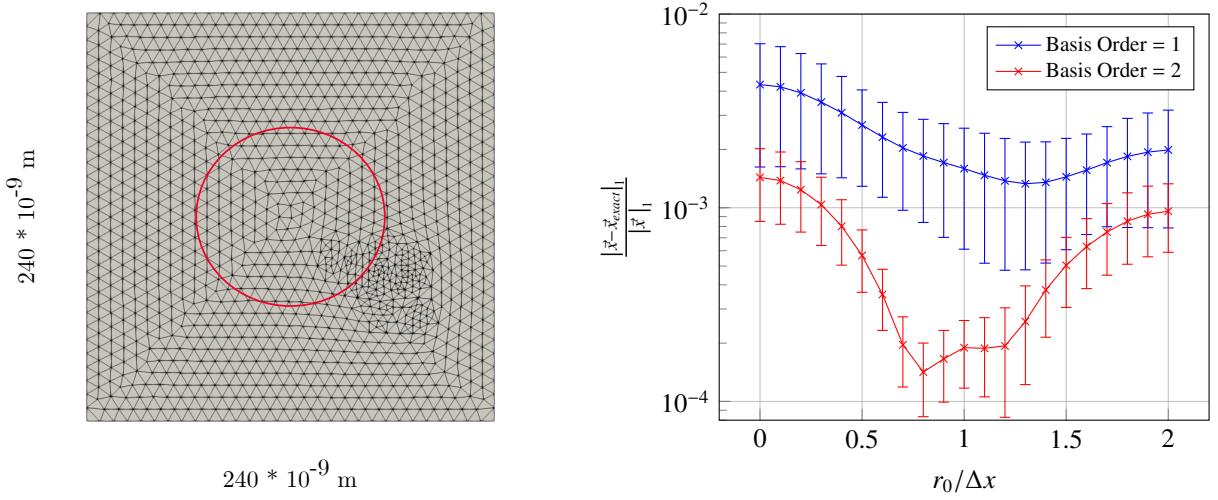
$$\omega_{pe} = \sqrt{\frac{n_0 q^2}{m_e \epsilon_0}} \approx 1.784 \times 10^9 \text{ rad/s} \tag{30}$$

$$\omega_{pi} = \sqrt{\frac{n_0 q^2}{m_i \epsilon_0}} \approx 4.163 \times 10^7 \text{ rad/s} \tag{31}$$

$$f = \frac{\omega}{2\pi} = \frac{1}{2\pi} \sqrt{\omega_p^2 + \omega_v^2} \approx 1.448 \text{ GHz} \tag{32}$$

As the wave is an infinite, steady wave, we can derive the constant phase velocity:

$$v_p = \sqrt{\frac{f}{f_v}} c \approx 1.02c > c \tag{33}$$

13

(a) Mesh used for unstructured orbit tests. The orbit base case is shown in red.

(b) Parameter scan of particle radius for the unstructured orbit tests

Figure 5: A parameter scan where (a) represents the geometry being studied and (b) shows the L_1 norm of the error in the electron position.

This gives the maximum initial electron velocity as defined below, which is then initialised in phase with the electric field. The values of v_x and v_z are initialised to zero.

$$v_y = \frac{qE_{mag}}{m_e\omega} \approx 1932.5 \text{ m/s} \tag{34}$$

The velocity $\vec{u}$ of a given particle can now be calculated as follows, where p is the z component of the particle position:

$$\vec{u} = \vec{v}\sin\left(p + \frac{\pi}{2}\right) \text{ m/s} \tag{35}$$

Finally, we define the maximum magnitude of the magnetic field such that it is congruous with the magnitude of the electric field.

$$B_{mag} = \frac{\lambda}{2\pi}\frac{E_{mag}}{c^2}\left(\frac{n_0 q^2}{m_e\epsilon_0\omega} + \omega\right) \approx 3.53 \times 10^{-7} \text{ T} \tag{36}$$

Using the derivation above it is simple to formulate a computational description of the problem. We set up the problem on a 3-dimensional grid of hexahedral finite elements with periodic simulation boundaries in all directions, effectively creating infinite space for the TEM wave, which we simulate for one wave period. The wave is defined to travel in the z dimension of the computational mesh, with the majority of grid elements also in the z dimension. The x and y dimensions are each defined to have a constant 4 elements, while the z dimension has 24 elements. As we assumed the ions to be stationary in our derivation, we force them to remain immobile during the simulation. The computational particles are placed randomly within each element and weighted in order to achieve our previously specified plasma number density. Each cell is loaded with an equal amount of particles of each species, with the immobile ions being used to provide a positive background in order to maintain neutrality. We additionally ensure that the initial electron velocity is confined to the transverse direction in the plane of the electric field.

We now present results for this problem for a variety of particle per cell counts, showing the average of 100 runs using random initial particle loads, using error bars to represent the standard deviation in the data. Figure 6 shows the effect of increased particle radius on the average L_1 error of the simulated electric and magnetic fields at the end of the simulation, presented as a breakdown of the field components. As the problem is set up with plane wave polarisation with only non-zero E_y and B_x, we refer to these components as the signal components, and the remaining components as non-signal components. As each of the non-signal components for a given field behave in the same manner, we
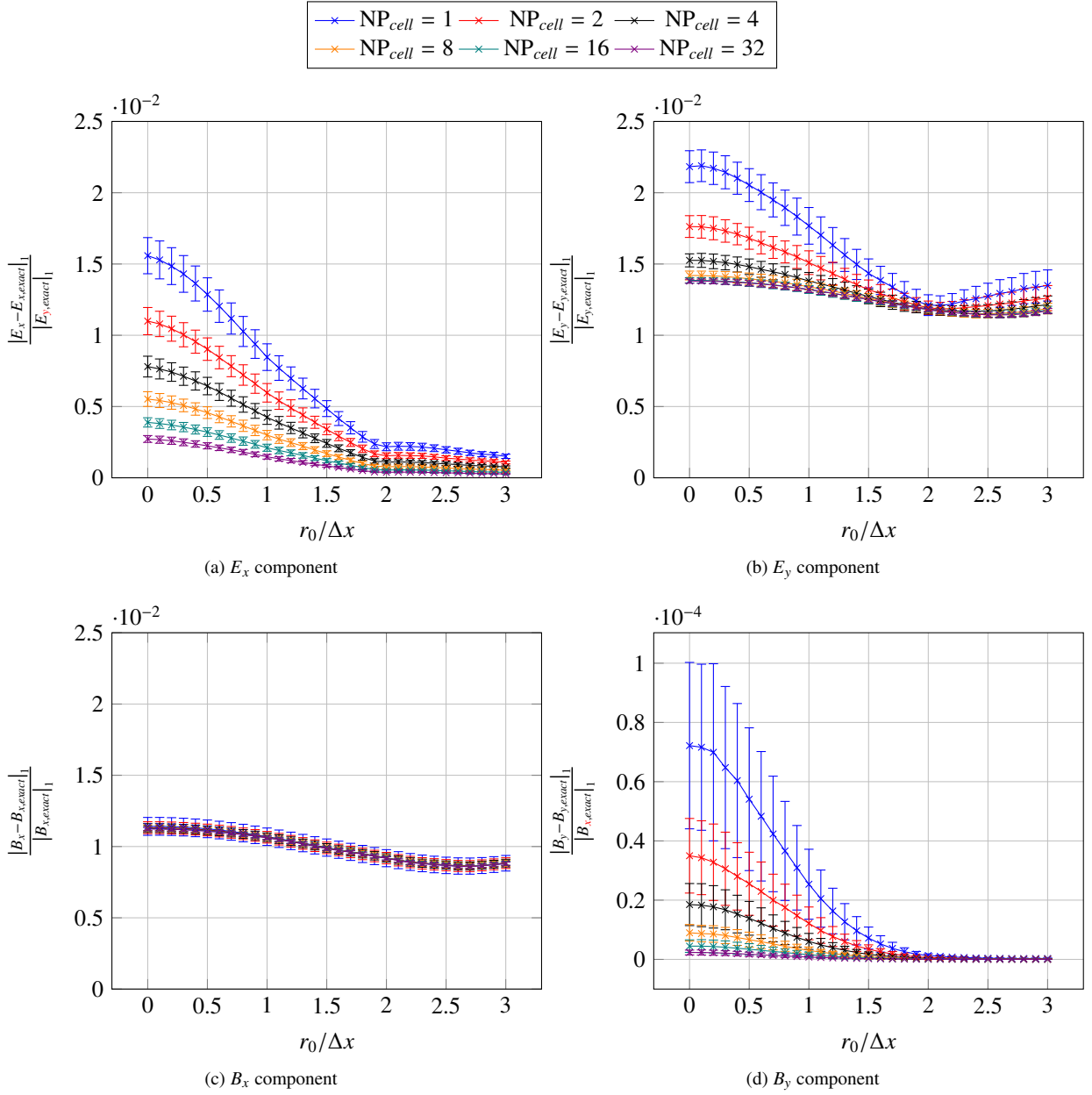
14

Figure 6: Graphs showing variation in L_1 norm of electric/magnetic field components as particle radius is increased.

choose to show data for E_x and B_y for these components, and E_y and B_x for the signal components. At first it is clear that we observe a smooth reduction in the L_1 error of the electric field which is reflected in the results shown for both the signal and non-signal components. This improvement continues to occur beyond the previously optimal value of $r_0 = 0.9 \times \Delta x$ observed in the orbit problem, continuing to improve as $r_0/\Delta x$ exceeds one. Additionally, we see a slight overall reduction in the variation from the initial seeds, but this effect appears to be negligible. Also of interest is that the error reduction due to smoothing for the non-signal field components is much greater than that observed in the component that contains the wave itself, suggesting that the noise in the wave is more sensitive to the particle distribution used. These differences are apparent in Figures 6a and 6b. Of particular note is that the E_y error converges to an approximate value of 0.014, whereas the other components continue to improve by tending towards
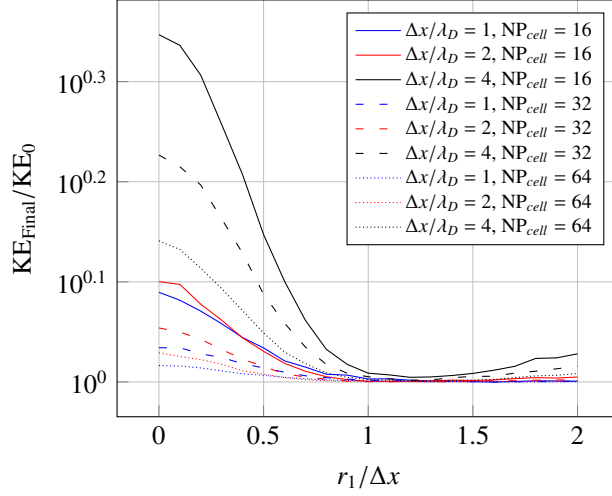
15

Figure 7: Ratio of final kinetic energy to starting kinetic energy for various particle radii.

zero at higher particle counts. We therefore conclude that the remaining E_y error is due to error in the scheme, and can be reduced by refining the problem further in space and/or time. This was verified through additional convergence tests, where the expected second-order convergence was observed.

Secondly, we examine the effects of particle smoothing on the computed result for the magnetic field. In accordance with the electric field data, we see a smooth reduction in L_1 error for both the signal and non-signal magnetic field components as particle radius is increased. There is good reduction in the B_x error, particularly as increasing the number of computational particles per cell has a negligible effect when compared to smoothing. However, the same does not hold true for the non-signal components where both smoothing and increasing particle count show good results, with smoothing performing particularly well at low particle counts. At higher particle counts smoothing reduces the error in these components to near zero. Regarding the statistical noise shown by the error bars, the B_x component shows almost no reduction in noise, in keeping with the trend observed regarding the electric field. Interestingly, the opposite holds true for the B_y component, showing a large reduction in statistical noise as radius is increased.

In general we conclude that, for this problem, the application of particle smoothing has the primary effect of reducing the noise in the solution for both the electric and magnetic fields in various ways. Specifically, where the solution should be zero there is a large reduction in the error in these components, and where the solution should be non-zero the errors converge to a seemingly constant value representing the space and time errors.

As a final note, we also examined the effect of smoothing on the frequency distribution of the error in the final result by applying a Fast Fourier Transform (FFT) to the E_y component of the electric field. However, we do not show these results in this paper as there appears to be little to no observable effect, beneficial or otherwise, on the resultant frequency distribution for this problem.

4.3. Numerical Heating

It is well documented that PIC codes are particularly susceptible to a phenomenon known as 'numerical heating', which leads to a growth in the kinetic energy of the system over the course of a simulation. This has previously been studied in detail by various authors [42, 43, 44], and is particularly prevalent in momentum conserving schemes such as that employed in EMPIRE-PIC [4]. This heating is typically controlled by three factors: (i) cell size, (ii) time-step size, and (iii) the number of computational particles used in the simulation. It has also been shown that the use of higher-order weighting schemes can significantly suppress such heating, even in cases where the Debye length is not completely resolved by the spatial grid [45, 46]. We now present our findings from numerical heating experiments within EMPIRE-PIC, with and without using the implemented higher-order particle shapes presented in this paper. To this end we examine the total kinetic energy (KE) of a neutral plasma consisting of electrons and hydrogen ions over 1000 plasma periods, at an initial temperature of 1.0 eV. Therefore we derive the key parameters of this problem
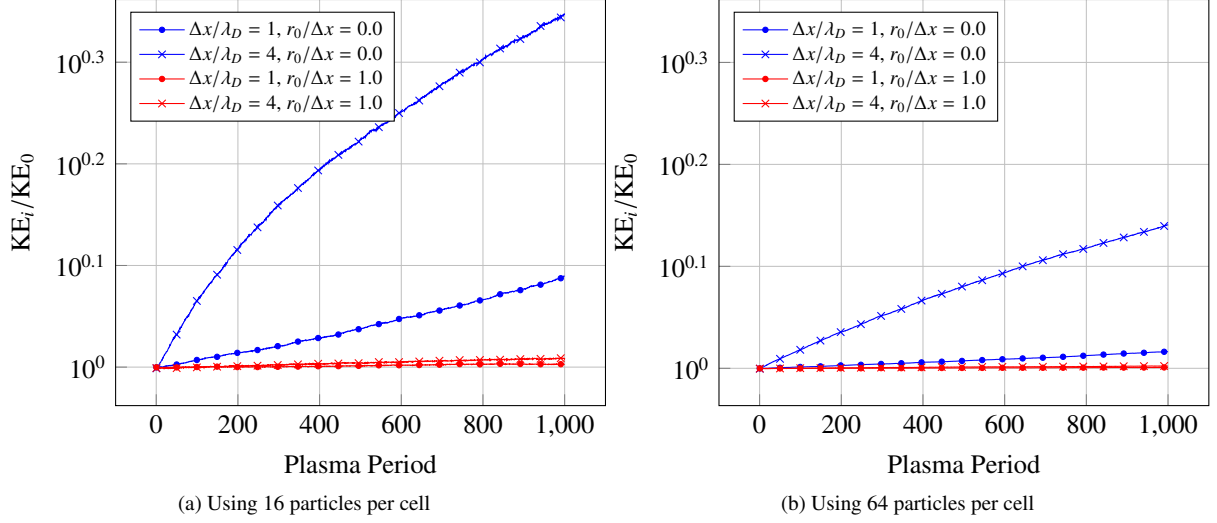
(a) Using 16 particles per cell

(b) Using 64 particles per cell

Figure 8: Graphs showing kinetic energy change over time for the vanilla code vs the optimal particle radius, for resolved and under-resolved λ_D.

as follows. We chose a number density of $n_0 = 10^{15}$ m^{-3} resulting in a plasma frequency $\omega_p \approx 1.784 \times 10^9$ rad/s, assuming the thermal motion of the electrons can be ignored.

Computationally, we use a 16×16 mesh of triangular elements with periodic boundaries in x and y, 10 time-steps per plasma period, with various amounts of particles per grid element and a range of particle radii. We also keep the amount of grid elements fixed, instead altering the size of the problem domain in order to determine the ratio between the Debye length and the cell size, Δx. Figure 7 shows the variation of the growth in simulation kinetic energy as the ratio of particle radius relative to Δx is increased, for problems using 16, 32, and 64 computational particles per cell. Additionally, the ratio of Δx to the Debye length is set at one of three levels: 1, 2, or 4. As expected, when the Debye length is severely under-resolved we observe large increases in the overall kinetic energy at the end of the simulation against that at the beginning. However, we see that such growth rapidly decreases as the particles are made smoother, particularly for the $\Delta x/\lambda_D = 4$ case. Interestingly, as we approach the radius of $r_0 = \Delta x$, we observe very little difference in the growth of kinetic energy for the problems with $\Delta x/\lambda_D \leq 2$ at both 16 and 32 particles per cell. This is promising in terms of performance, as we can maintain similar kinetic energy stability while using less grid cells and super-particles, reducing both computational requirements and load on the memory system.

The results are further validated in Figure 8, which shows the growth of the system kinetic energy at each simulated plasma period for values of $\Delta x/\lambda_D$ of 1 and 4, for the vanilla code and the optimal radius value of $r_0 = \Delta x$, using 16 and 64 particles per grid cell. Where the Debye length is under-resolved we observe extremely rapid growth in the kinetic energy of the system, increasing by 50% in approximately 300 periods for the 16 particle per cell case. In the case where $\Delta x/\lambda_D = 1$, the smoothed particles all but eliminate the numerical heating effects, with only mild kinetic energy growth throughout the simulation – 0.6% and 0.1% when 16 and 64 particles are used per grid cell, respectively, in contrast to the increases of 22.8% and 3.8% observed when using the base code. The benefits are also significant for the under-resolved case, with optimal particle smoothing resulting in a heating trend similar to that of the base code with a resolved mesh. Additionally, these results remain consistent for Figure 8b, exhibiting good reduction in heating effects. Again, the smooth particles almost eliminate self heating where λ_D is resolved, and continue to show good performance on an under-resolved mesh – on par with the $r_0/\Delta x = 0$ results in the resolved case.

4.4. Electrostatic Plasma Slab Problem

In order to properly assess the behaviour of our algorithm for electrostatic PIC simulations it is prudent to examine a more complex test case than the simple orbit discussed in Section 4.1. We now consider the 1D expansion of a collisionless slab of plasma into a vacuum, a benchmark problem that has previously been used for verifying PIC
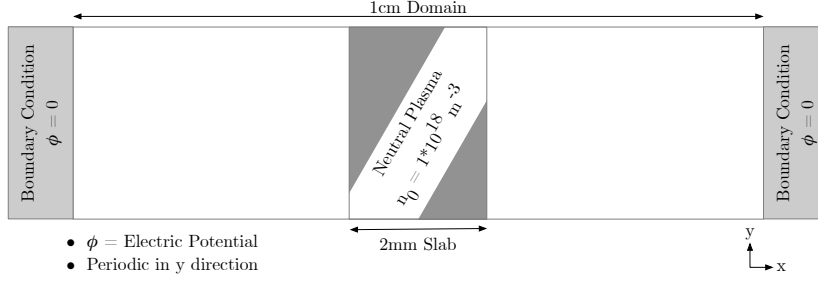
17

Figure 9: Image showing the setup of the plasma slab expansion problem.

simulations [47]. As EMPIRE-PIC is a 2D/3D PIC code, it cannot be used to directly simulate an entirely 1D problem. We therefore set up a 2D mesh with fixed $N_y = 2$, using periodic simulation boundaries for the y direction, and quadrilateral elements. We use a Dirichlet boundary condition in the x direction, setting the electric potential to zero in order to ensure that the problem is well posed. The problem starts with a charge neutral slab with a thickness of 2 mm placed at the centre of a domain of length 1 cm, allowed to expand for a total time of 2.5×10^{-9} s. The ions are initialised cold, whereas the electrons are assigned a finite initial temperature of 1 eV. Additionally, the ions have a mass of $10 \times m_e$. We choose such an artificially low ion to electron mass ratio in order to accelerate the expansion of the plasma slab. This setup is shown pictorially in Figure 9. Each grid cell of the simulation that contains plasma is initially loaded with 8000 particles of both species, weighted such that we achieve a plasma number density of $n_0 = 1 \times 10^{18}$ m^{-3}. Given these parameters, we can now derive the plasma frequencies and Debye length as follows. $\omega_{pe} \approx 5.641 \times 10^{10}$ rad/s, $\omega_{pi} \approx 1.317 \times 10^9$ rad/s, and $\lambda_D \approx 6.89 \times 10^{-6}$ m. This allows us to choose a base N_x such that $\Delta x/\lambda_D \approx 1$, and Δt such that $\omega_p \Delta t < 0.1$. We now have $N_x = 1600$, $N_y = 2$, and $N_{\Delta t} = 250$.

We now show results for the simulated cell-centred ion density for this problem for the base code, and for the smooth particle implementation with $r_0/\Delta x = 1$. As the density solution output by the vanilla code is extremely noisy, we filter the data using a one-dimensional Gaussian filter with $\sigma = 3\Delta x$. We display the error in the ion density as a shaded area, which shows the standard deviation in the raw data in both cells in the y dimension for the given point, and the six surrounding pairs of cells in the x dimension, specifically three pairs on each side of the point.

Figure 10 shows the results of these experiments. It is clear to see from Figure 10a that the vanilla code exhibits a very high amount of noise in the simulated ion density, with most of this noise building up at the interface between the slab of plasma and the vacuum, with an RMSE value of 0.1402 (normalised by the number density). It is also evident that the use of a second-order basis can reduce this noise, with RMSE of 0.0288. Figure 10b shows the results of the same experiment for the smoothed particle representation. The magnitude of the noise in the solution is greatly reduced by particle smoothing, both at the interface and in the centre of the plasma slab to the point of being only marginally visible (RMSE = 0.0127). Close inspection reveals that the second-order basis continues to outperform the first-order basis (RMSE = 0.0074).

These experiments were also repeated using 800 particles per cell. These results are shown in Figures 10c and 10d. When comparing Figure 10a to Figure 10d we can see that using 800 smoothed particles produces a result that is significantly less noisy than when 8000 traditional particles are used for both basis orders – RMSE = 0.0469 and 0.0207 for first- and second-order bases, respectively. This is significant as we can use an order of magnitude less particles while also maintaining a greatly improved solution over the base code. This resulted in a runtime approximately 2.5 times slower than the base code runs using 8000 traditional particles per cell – as expected given that $N_v = 25$ and PPC = 800 for the smoothed particle tests. This is a positive result as, for a first-order basis, the smoothed particle tests result in an RMSE approximately three times lower than that achieved with the base code. Additionally, the purely computational nature of the virtual particles means that the memory footprint of the smoothed particle tests is lower by comparison.

We also examined the error in simulated electric potential for both a resolved and under-resolved Debye length. This error did not appear to be sensitive to particle smoothing in this case. This result is interesting as it suggests that the large reduction in density noise has negligible effect on the simulated potential.
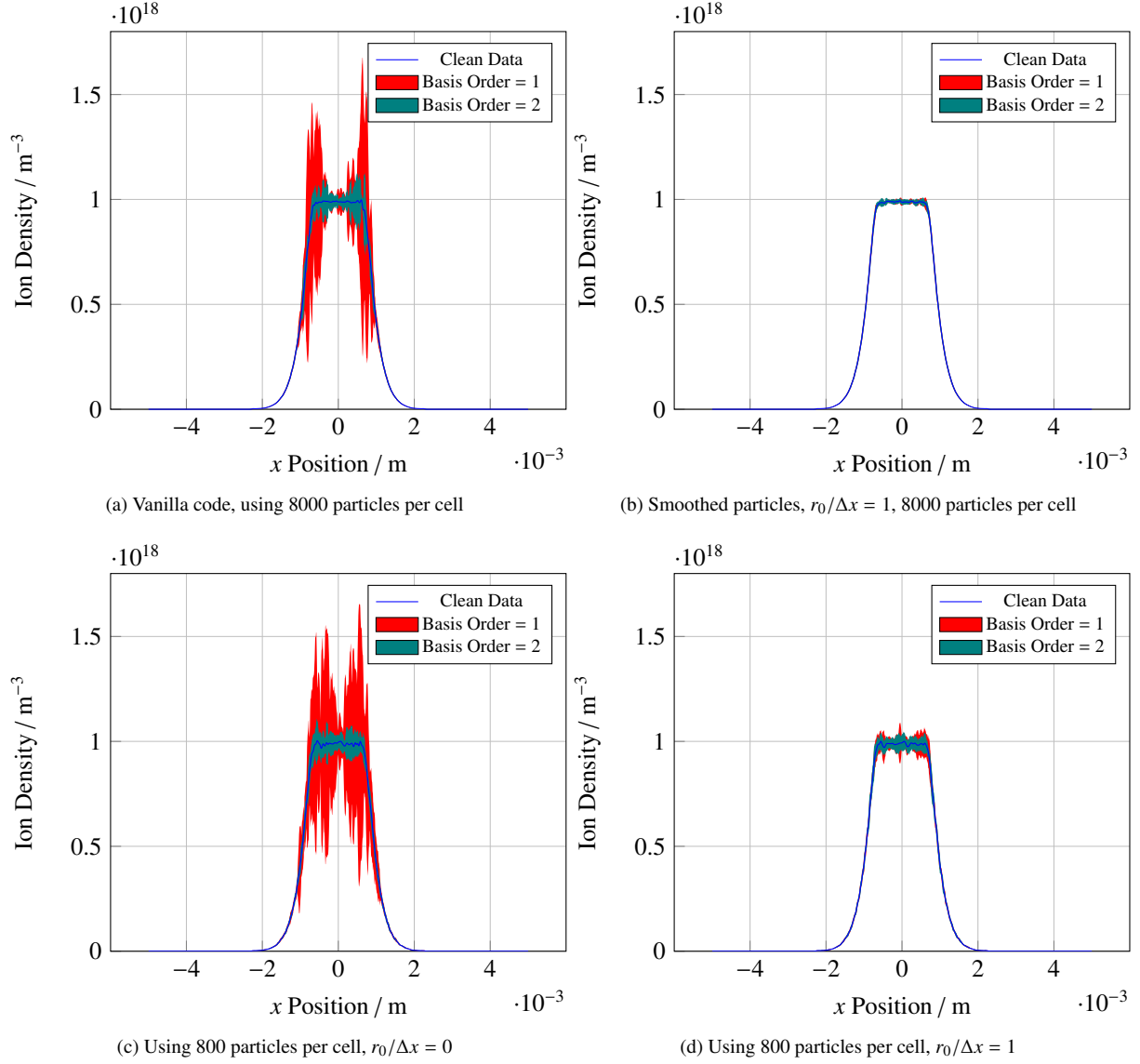
(a) Vanilla code, using 8000 particles per cell

(b) Smoothed particles, $r_0/\Delta x = 1$, 8000 particles per cell

(c) Using 800 particles per cell, $r_0/\Delta x = 0$

(d) Using 800 particles per cell, $r_0/\Delta x = 1$

Figure 10: Graphs showing the noise in the simulated ion density for the vanilla code and smoothed particle shapes.

5. Conclusion

As the need to simulate the behaviour of plasmas within devices under various conditions using complex geometry continues to grow, PIC algorithms must adapt to these changing requirements. As a result, both higher-order PIC methods and the use of unstructured FEM-PIC has become an area of great interest to the plasma simulation community. While higher-order unstructured meshes show promise, they also impose the additional requirement that the particles being simulated possess some smooth shape instead of the usual Dirac delta function due to the assumption that the source terms are smooth.

In this paper, we have proposed a higher-order representation of particles in PIC algorithms, where each particle has a smooth shape function that is limited by a specified finite radius. A unique feature of our approach is that the implementation of this smooth representation is achieved by surrounding super-particles with delta shape computational virtual particles that have fixed offsets and weights. As this moves the quadrature from the mesh to a set of points

19

surrounding the particle we can use the same PIC procedures as the base code with minimal modifications. While we derive the offsets and weights from Gaussian quadrature rules, the applications of this representation are broad as the offsets and weights may be tuned to represent any desired shape of the particle cloud.

We show how the proposed changes to the core PIC algorithm are implemented within SNL's unstructured PIC code, EMPIRE-PIC, using periodic boundary conditions for both electrostatic and electromagnetic problems. The accuracy and convergence of the modified algorithm was examined using a set of representative benchmark problems and contrasted to the behaviour of the base EMPIRE-PIC code. Our results show approximately 70% improvement in the tracking of basic particle motion on a distorted mesh, with this increasing to an order of magnitude improvement when a second-order basis is used. We additionally show extremely successful suppression of self-heating for both resolved and under-resolved grids, and a significant reduction in noise of the simulated ion density in an electrostatic plasma slab expansion while being able to use an order of magnitude fewer super-particles.

The work in this paper represents a step towards more accurate PIC applications, enabling improved simulations of plasma phenomena. Our method additionally increases the computational intensity of the PIC algorithm, without drastically raising the burden on the memory system, possibly being beneficial on new architectures, where the memory sub-system becomes a bottleneck.

5.1. Future Work

The work presented in this paper opens up a number of avenues for further research. As the algorithm implementation theoretically allows for particles of arbitrary shapes to be represented, further work could examine the effects of a multitude of particle shapes on the results of a PIC simulation.

Finally, significant work must also be undertaken to implement the more advanced non-periodic simulation boundary conditions to enable behaviour such as particle scattering and absorption. A method of handling such boundaries while using higher-order charge weighting has previously been shown by Pointon [48], where the charge weighting is smoothly transitioned back to first-order as the particle approaches a boundary. In the case of our algorithm, this would be analogous to gradually reducing the radius of smooth particles back towards zero as simulation boundaries are approached. Implementing such a variable radius remains charge-conserving as long as the virtual particle weights to sum to one and all element crossings continue to be tracked. However, such a method is likely to exhibit the energy conservation issues present in structured and AMR-PIC codes.

6. Acknowledgements

References

[1] T. D. Arber, K. Bennett, C. S. Brady, A. Lawrence-Douglas, M. G. Ramsay, N. J. Sircombe, P. Gillies, R. G. Evans, H. Schmitz, A. R. Bell, C. P. Ridgers, Contemporary particle-in-cell approach to laser-plasma modelling, Plasma Physics and Controlled Fusion 57 (11) (2015) 113001.

[2] M. A. Riquelme, E. Quataert, D. Verscharen, Particle-in-cell simulations of continuously driven mirror and ion cyclotron instabilities in high beta astrophysical and heliospheric plasmas, The Astrophysical Journal 800 (1) (2015) 27.

[3] G. Fridman, G. Friedman, A. Gutsol, A. B. Shekhter, V. N. Vasilets, A. Fridman, Applied plasma medicine, Plasma Processes and Polymers 5 (6) (2008) 503–533.

[4] C. K. Birdsall, A. B. Langdon, Plasma Physics via Computer Simulation, Plasma Physics Series, Institute of Physics Publishing, Bristol BS1 6BE, UK, 1991.

[5] J. M. Dawson, Particle simulation of plasmas, Reviews of Modern Physics 55 (1983) 403–447.

[6] K. S. Yee, Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media, IEEE Transactions on Antennas and Propagation 14 (3) (1966) 302–307.

[7] A. B. Langdon, C. K. Birdsall, Theory of plasma simulation using finite-size particles, The Physics of Fluids 13 (8) (1970) 2115–2122.

[8] R. A. Fonseca, L. O. Silva, F. S. Tsung, V. K. Decyk, W. Lu, C. Ren, W. B. Mori, S. Deng, S. Lee, T. Katsouleas, J. C. Adam, OSIRIS: A three-dimensional, fully relativistic particle in cell code for modeling plasma based accelerators, in: P. M. A. Sloot, A. G. Hoekstra, C. J. K. Tan, J. J. Dongarra (Eds.), Computational Science ICCS 2002, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[9] K. Germaschewski, W. Fox, S. Abbott, N. Ahmadi, K. Maynard, L. Wang, H. Ruhl, A. Bhattacharjee, The plasma simulation code: A modern carticle-in-cell code with patch-based load-balancing, Journal of Computational Physics 318 (2016) 305 – 326.

[10] K. J. Bowers, B. J. Albright, L. Yin, B. Bergen, T. J. T. Kwan, Ultrahigh performance three-dimensional electromagnetic relativistic kinetic plasma simulation, Physics of Plasmas 15 (5) (2008) 055703.

[11] K. J. Bowers, B. J. Albright, B. Bergen, L. Yin, K. J. Barker, D. J. Kerbyson, 0.374 PFLOP/s trillion-particle kinetic modeling of laser plasma interaction on roadrunner, in: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, SC'08, IEEE Press, Piscataway, NJ, USA, 2008, pp. 63:1–63:11.

[12] B. Wang, S. Ethier, W. Tang, T. Williams, K. Z. Ibrahim, K. Madduri, S. Williams, L. Oliker, Kinetic turbulence simulations at extreme scale on leadership-class systems, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'13), 2013, pp. 1–12.

[13] W. Tang, B. Wang, S. Ethier, G. Kwasniewski, T. Hoefler, K. Z. Ibrahim, K. Madduri, S. Williams, L. Oliker, C. Rosales-Fernandez, T. Williams, Extreme scale plasma turbulence simulations on top supercomputers worldwide, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'16), 2016, pp. 502–513.

[14] E. Wang, S. Wu, Q. Zhang, J. Liu, W. Zhang, Z. Lin, Y. Lu, Y. Du, X. Zhu, The Gyrokinetic Particle Simulation of Fusion Plasmas on Tianhe-2 Supercomputer, in: 2016 7th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA), 2016, pp. 25–32.

[15] S. Dey, R. Mittra, A locally conformal finite-difference time-domain (FDTD) algorithm for modeling three-dimensional perfectly conducting objects, IEEE Microwave and Guided Wave Letters 7 (9) (1997) 273–275.

[16] I. Zagorodnov, R. Schuhmann, T. Weiland, Conformal FDTD-methods to avoid time step reduction with and without cell enlargement, Journal of Computational Physics 225 (2) (2007) 1493 – 1507.

[17] G. Yang, D. M. Causon, D. M. Ingram, R. Saunders, P. Battent, A cartesian cut cell method for compressible flows part a: Static body problems, The Aeronautical Journal (1968) 101 (1002) (1997) 47–56.

[18] M. J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, Journal of Computational Physics 53 (3) (1984) 484 – 512.

[19] J.-L. Vay, P. Colella, J. W. Kwan, P. McCorquodale, D. B. Serafini, A. Friedman, D. P. Grote, G. Westenskow, J.-C. Adam, A. Héron, I. Haber, Application of adaptive mesh refinement to particle-in-cell simulations of plasmas and beams, Physics of Plasmas 11 (5) (2004) 2928–2934.

[20] J.-L. Vay, A. Almgren, J. Bell, L. Ge, D. Grote, M. Hogan, O. Kononenko, R. Lehe, A. Myers, C. Ng, et al., Warp-X: A new exascale computing platform for beam–plasma simulations, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 909 (2018) 476–479.

[21] R. Marchand, PTetra, a tool to simulate low orbit satellite–plasma interaction, IEEE Transactions on Plasma Science 40 (2) (2012) 217–229.

[22] J. Roussel, F. Rogier, G. Dufour, J. Mateo-Velez, J. Forest, A. Hilgers, D. Rodgers, L. Girard, D. Payan, SPIS open-source code: Methods, capabilities, achievements, and prospects, IEEE Transactions on Plasma Science 36 (5) (2008) 2360–2368.

[23] C. K. Birdsall, D. Fuss, Clouds-in-clouds, clouds-in-cells physics for many-body plasma simulation, Journal of Computational Physics 3 (1969) 494–511.

[24] G. Jacobs, J. Hesthaven, High-order nodal discontinuous galerkin particle-in-cell method on unstructured grids, Journal of Computational Physics 214 (1) (2006) 96–121.

[25] J. Hesthaven, T. Warburton, Nodal high-order methods on unstructured grids: I. time-domain solution of maxwell's equations, Journal of Computational Physics 181 (1) (2002) 186–221.

[26] E. Edwards, R. Bridson, A high-order accurate particle-in-cell method, International Journal for Numerical Methods in Engineering 90 (9) (2012) 1073–1088.

[27] T. Stindl, J. Neudorfer, A. Stock, M. Auweter-Kurtz, C.-D. Munz, S. Roller, R. Schneider, Comparison of coupling techniques in a high-order discontinuous galerkin-based particle-in-cell solver, Journal of Physics D: Applied Physics 44 (19) (2011) 194004.

[28] M. C. Pinto, S. Jund, S. Salmon, E. Sonnendrücker, Charge-Conserving FEM–PIC Schemes on General Grids, Comptes Rendus Mecanique 342 (10–11) (2014) 570–582.

[29] D. A. S. Brown, S. A. Wright, S. A. Jarvis, Performance of a second order electrostatic particle-in-cell algorithm on modern many-core architectures, Electronic Notes in Theoretical Computer Science 340 (2018) 67–84.

[30] J. Squire, H. Qin, W. M. Tang, Geometric Integration of the Vlasov-Maxwell System with a Variational Particle-in-Cell Scheme, Physics of Plasmas 19 (8) (2012) 084501.

[31] H. Moon, F. L. Teixeira, Y. A. Omelchenko, Exact Charge-Conserving Scatter-Gather Algorithm for Particle-in-Cell Simulations on Unstructured Grids: A Geometric Perspective, Computer Physics Communications 194 (2015) 43–53.

[32] Y. L. Klimontovich, The Statistical Theory of Non-Equilibrium Processes in a Plasma: International Series of Monographs in Natural Philosophy, Vol. 9, Elsevier, 2013.

[33] T. H. Dupree, Kinetic Theory of Plasma and the Electromagnetic Field, Physics of Fluids (1958-1988) 6 (12) (1963) 1714–1729.

[34] J. Boris, Relativistic plasma simulation: Optimization of a hybrid code, in: Proceedings of the Fourth Conference on Numerical Simulation of Plasmas, Naval Research Laboratory, Washington, D.C, 1971, pp. 3–68.

[35] J. Jin, The Finite Element Method in Electromagnetics, 3rd Edition, Wiley-IEEE Press, 2014.

[36] J.-C. Nédélec, Mixed finite elements in $\mathbb{R}^3$, Numerische Mathematik 35 (3) (1980) 315–341.

[37] F. Brezzi, M. Fortin, Mixed and Hybrid Finite Element Methods, Vol. 15, Springer Science & Business Media, 2012.

[38] B. Ripperda, F. Bacchini, J. Teunissen, C. Xia, O. Porth, L. Sironi, G. Lapenta, R. Keppens, A comprehensive comparison of relativistic particle integrators, The Astrophysical Journal Supplement Series 235 (1) (2018) 21.

[39] J. Villasenor, O. Buneman, Rigorous charge conservation for local electromagnetic field solvers, Computer Physics Communications 69 (2)

(1992) 306 – 316.

[40] F. F. Chen, Introduction to Plasma Physics and Controlled Fusion, 3rd Edition, Springer, 2016.

[41] J. A. Bittencourt, Fundamentals of Plasma Physics, 3rd Edition, Springer, 2004.

[42] A. Langdon, Effects of the spatial grid in simulation plasmas, Journal of Computational Physics 6 (2) (1970) 247 – 267.

[43] R. Hockney, S. Goel, J. Eastwood, Quiet high-resolution computer models of a plasma, Journal of Computational Physics 14 (2) (1974) 148 – 158.

[44] R. Hockney, Measurements of collision and heating times in a two-dimensional thermal computer plasma, Journal of Computational Physics 8 (1) (1971) 19 – 44.

[45] M. Shalaby, A. E. Broderick, P. Chang, C. Pfrommer, A. Lamberts, E. Puchwein, SHARP: A spatially higher-order, relativistic particle-in-cell code, The Astrophysical Journal 841 (1) (2017) 52.

[46] P. Rambo, Numerical heating in hybrid plasma simulations, Journal of Computational Physics 133 (1) (1997) 173 – 180.

[47] B. I. Cohen, A. B. Langdon, D. W. Hewett, R. J. Procassini, Performance and optimization of direct implicit particle simulation, Journal of Computational Physics 81 (1) (1989) 151 – 168.

[48] T. Pointon, Second-order, exact charge conservation for electromagnetic particle-in-cell simulation in complex geometry, Computer Physics Communications 179 (8) (2008) 535–544.