Journal of Computational Physics 373 (2018) 571–604

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

CHAOS: An octree-based PIC-DSMC code for modeling of electron kinetic properties in a plasma plume using MPI-CUDA parallelization

Revathi Jambunathan [*], Deborah A. Levin

Department of Aerospace Engineering, University of Illinois, Urbana-Champaign, IL 61801, United States of America

A R T I C L E I N F O

Article history:
Received 23 October 2017
Received in revised form 21 June 2018
Accepted 4 July 2018
Available online 10 July 2018

Keywords:
Particle-In-Cell
Neutralization of plasma plume
Electron kinetics
Forest of Morton-Z ordered linear octrees
Heterogeneous architectures

A B S T R A C T

A new computational framework for a coupled PIC-DSMC tool using multiple GPUs to model the kinetic behavior of electrons in plasma plumes is presented in this work. The disparate length scales of the Debye length and the collisional mean free path are resolved by using separate, independent linearized Morton Z-ordered forest of trees. A 2:1 restraint is imposed for the PIC module to solve partial differential equations in the context of an AMR/Octree framework. The MPI-CUDA parallelization strategies used to implement a preconditioned conjugate gradient method for solving the electrostatic Poisson's equation on the 2:1 octree are discussed and the scaling of the code to near ideal speedup as a function of the number of GPUs is demonstrated. The PIC method is validated using analytical test cases, and the octree-based PIC simulations are found to be ten times more efficient compared to the uniform grid method especially for plume simulations which have large density variations. The computational strategies are then demonstrated with the simulation of collisionless, mesothermal plasma plumes using a kinetic approach for both ions and electrons. The effect of ion mass and electron source location are analyzed by comparing plume dynamics and electron velocity distribution functions. It is shown that a more confined mesothermal plume is observed for the heavier xenon ions, present in electric propulsion devices, compared to protons. The confinement of the xenon plume traps the electrons resulting in higher electron temperatures compared to the proton plasma case. In both the simulations, however, the electron temperature is found to be anisotropic. Finally, when a shifted electron source location case is considered the electron velocity distributions in all three directions are found to be unequal and non-Maxwellian, contrary to the co-located case. The ion beam is observed to attract the electrons, which initially oscillate between the radial edges of the ion beam as confirmed by the bi-modal velocity distribution at early times. As the plume evolves, it electrostatically traps these electrons within the ion beam, allowing them to thermalize, as observed from the single-peak electron velocity distribution functions at later times. These results demonstrate that GPUs can efficiently accelerate computations of a fully kinetic approach to enable the study of electron kinetics and neutralization with the highest physical fidelity.

[*] Corresponding author.
E-mail address: jambuna2@illinois.edu (R. Jambunathan).

https://doi.org/10.1016/j.jcp.2018.07.005
0021-9991/© 2018 Published by Elsevier Inc.

1. Introduction

Electric propulsion (EP) is an attractive solution for station-keeping and attitude control of small-satellites and cube-satellites in the space environment. With increase in the sophistication of satellites, the prediction of their reliability and operational lifetime of the solar-cell arrays is gaining importance. EP devices generate thrust by accelerating ionized propellant, usually xenon, using an applied electric and sometimes magnetic field, to produce a plume with the required exhaust velocity [1]. This weakly ionized plasma plume consists of 10% xenon ions with a speed of 30–60 km/s and 90% un-ionized neutrals. The high speed beam ions undergo collisions, called charge-exchange (CEX) [1] reactions, with the plume and ambient neutrals to generate slower ions with a speed of 0.5–1 km/s [2]. These CEX ions are influenced by the electric field induced between the plume and spacecraft surfaces or solar panels resulting in their backflow or reverse streaming. The ions in the backflow region impinge on the solar panels with sufficient energy, on the order of 10 eV [3], to cause changes in the electro-chemical properties of the surface as well as erosion, affecting their efficiency. In order to prevent a build-up of positive charge in the plume and to reduce the spacecraft charge, the plume is neutralized by an electron beam emitted from an external hollow cathode device on the thruster. Inclusion of these hollow cathode plumes [4,5] is essential as the electrons play an important role in the charge distribution, density, and energy of the CEX ions in the backflow region, and in turn, the prediction of solar cell performance. Although this work is motivated by EP, the numerical methodologies discussed in this paper are also relevant to other plasma applications, such as, understanding instabilities in astrophysical jets [6,7], modeling of solar wind interactions [8], streamer formation studies [9–11], as well as plasma–surface interactions to modify biomaterials [12,13].

Due to the low density of charged species in the thruster plume, on the order of $10^{15}/\mathrm{m}^3$ for ion thruster plumes [1], the continuum assumption becomes invalid and a kinetic approach is required to model the physical processes of the EP thruster plume. Particle-In-Cell (PIC) [14] is a well established kinetic approach used to compute the time evolving characteristics of the charged particles in the plasma and their interactions with the induced electric field. In this method, the charged and neutral species are modeled as computational macroparticles, wherein each macroparticle represents a large number of real ions, electrons, or neutrals. The computational domain is first discretized into a grid and the charge density distribution is determined by mapping particles to the cells. The electric field is obtained by solving Poisson's equation in an electrostatic application, and the electric and magnetic fields are evaluated by solving Maxwell's equations for electromagnetic applications. Finally, the forces exerted by the fields push the charged particles to new positions thereby generating a new charge distribution every timestep. The PIC approach has been used for a wide variety of applications such as, modeling of electromagnetic solar wind interactions with lunar crustal magnetic anomalies [8], laser–plasma interactions [15], relativistic modeling of pulsar magnetosphere [16], dusty plasmas [17–19], as well as for thruster plumes [20–25].

PIC codes perform particle movement and field-solve using explicit or implicit schemes. In explicit PIC schemes, where the particle movement and field solve are decoupled, the cell size of the grid used for the electric field calculations must be less than the local Debye length [26] and the timestep should be such that, $\Delta t \omega_{pe} < 0.1$, to obtain accurate results without numerical instabilities [14,26,27]. On the other hand, numerically stable semi-implicit [28,29] and implicit [30,31] PIC algorithms can accurately model non-linear plasma evolution using coarser cell sizes and at least an order of magnitude larger timestep compared to the explicit method, but with increased complexity in their implementation. In this work, since we are interested in accurately resolving the transient electron kinetics, the use of cell sizes less that the local Debye length and small timesteps is essential. In addition, the use of linearized octrees and parallelization strategies discussed in this work are applicable even for semi-implicit, and implicit schemes.

PIC tools have traditionally used a uniform Cartesian grid [32] to discretize the domain and compute the self-consistent electric and magnetic fields. However, for problems with multiple length-scales, such as, the expanding plasma plume as well as plasma–surface interactions, the Debye length is smaller near the thruster exit or in the sheath region compared to the far-field. The use of a uniform grid with cell size less than the smallest Debye length would unnecessarily increase the number of cells in the domain and the computational cost. Therefore, an adaptive mesh refinement (AMR) approach [2,33–38] to refine cells in the regions with small Debye lengths and coarsen in the regions where the Debye length is large can be advantageous. The three-dimensional hierarchical tree structure used to store such an AMR grid is called an octree [39]. Solving Poisson's equation accurately on such octree grids is crucial for computing the electric field in PIC, but, it requires strategies different from the uniform grid approach to account for neighboring cells with different sizes. A number of Poisson solver libraries, such as, PETSC [40], Dendro [41], and Deal.II [42] are available for use on octree grids. Poisson's equation has been solved on octree grids using a multigrid approach as well [43–45]. Also a finite volume approach has previously been used to solve Poisson's equation on an octree structure [46,33,47]. These approaches have used multiple distributed-memory CPU systems for parallelization by employing the MPI paradigm.

Recently, general-purpose computing on Graphics Processor Units (GPGPUs) is being exploited to accelerate code performance by employing massively large number of cores or threads. GPUs have been used for solving Poisson's equation for PIC methods [48–51]. PIConGPU [49,52] used 18,000 GPUs to simulate Kelvin–Helmholtz instabilities in an effort to model astrophysical jets and obtained nearly 70% strong scaling efficiency. Bastrakov et al. [53] developed a three-dimensional PIC tool called PICADOR to perform simulations using GPUs as well as Intel accelerators. However, a uniform grid was used in these tools for solving Poisson's equations. For the plasma plume simulations of interest here an octree approach is required since the Debye length is highly non-uniform.

In addition to the PIC method that models the electric field, the well known particle-based Direct Simulation Monte Carlo [54] (DSMC) approach is required to model the neutral-ion momentum and CEX collisions. In our recent work [55], we developed an octree-based solver that uses multiple-GPUs, called Cuda-based Hybrid Approach for Octree Simulations (CHAOS), to perform DSMC simulations. The code achieved 85% strong scaling and 100% weak scaling efficiency for argon gas flow through porous media. The main contribution of our previous work was that we employed a linearized octree structure as opposed to a pointer-based data structure, and exploited the Morton encoding approach to map particles to the octree cells, also known as leaf nodes. This particle-to-grid mapping, which has been shown to be a computationally expensive procedure, even for the PIC method [56], is optimized for linearized octrees and accelerated using GPUs [55]. Previously, Korkut et al. [2] implemented the momentum and charge-exchange collisions between the ions and the neutrals, including species weighting factors and species-dependent timestep to account for their different number densities and speeds. Since a plasma plume evolution is generally governed by both collisions and the electric field, a coupled PIC-DSMC [2,33,57] approach has been used.

In the present work, we implement an explicit Particle-In-Cell module in CHAOS that can be coupled with the DSMC module using the MPI-CUDA paradigm for parallelization. The dominant length scales for PIC and DSMC are governed by the electron Debye length and ion-neutral mean free path, respectively. To satisfy the multiple length-scales for the different physical processes, we implement a modularized framework, wherein the DSMC module constructs a linearized octree to satisfy collision length-scales and the PIC module uses its own linearized octree that satisfies the electric field length scales. The grid construction and domain decomposition methodologies for the PIC-DSMC framework as well as the strategies used to solve Poisson's equation on an octree grid, using multiple GPUs, are discussed in Sec. 2. In Sec. 3, we demonstrate the accuracy of the PIC solver and its ability to conserve momentum and energy by performing standard test cases. We also compare the octree simulations with uniform grid, and show that for cases with large density gradients, the octree method is approximately ten times more efficient than uniform grid.

Finally, we perform *collisionless* mesothermal plasma simulations such that $v_{te} >> v_{ibeam} >> v_{ti}$, where v_{te}, v_{ibeam}, and v_{ti}, are the electron thermal velocity, beam velocity, and ion thermal velocity, respectively [58]. PIC-DSMC coupled results will be presented at a future date. Wang et al. [58], and Hu et al. [59,60], have modeled two-dimensional mesothermal plumes with a reduced ion-to-electron mass ratio of 1836, where the ion and electron sources are co-located. But, electric thrusters typically use xenon as propellant and the electron source is shifted from the thruster exit. Usui et al. [61] have modeled a mesothermal plume with a shifted electron source, but, the ions were modeled with a reduced mass equal to that of a proton. For comparison of electron kinetics with these previous studies, we first perform three-dimensional mesothermal plume simulations with reduced ion mass and co-located electron source using the PIC module in CHAOS. Then we sequentially increase the ion mass to model xenon ions and shift the electron source location to study their effect on the electron kinetics as well as the plume neutralization. The simulation set-up is given in Sec. 4, and the plume structure as well as the interesting electron kinetic behavior due to the induced electrostatic forces is analyzed in Sec. 5.

2. Numerical implementation and parallelization strategies

In this section, we present the strategies used to implement the MPI-CUDA parallelization of the PIC algorithm and to couple the DSMC and PIC modules. A flowchart of the coupled PIC-DSMC framework in CHAOS is shown in Fig. 1. The unique features in CHAOS, that is common to both particle methods, PIC and DSMC, is the implementation of linearized Z-ordered octrees and efficient fast bitwise computations to perform particle-to-octree mapping using GPUs. Even though the strategy to construct the linearized octrees is the same for PIC and DSMC, due to the different cell size criteria required for collisions and computation of the electric field, separate forest of trees (FOT) are used in the two modules, as will be discussed in subsection 2.1. Particle movement is grid-independent and the particles are mapped to the respective octrees in the DSMC and PIC modules. Since Poisson's equation is solved on the octree, an additional spatial restraint, known as 2:1, is imposed, as discussed in Sec. 2.2. In Sec. 2.3, we discuss the domain decomposition strategy implemented for load balancing as well as the methodology used to store the list of face neighbors that are contained on different processors. The parallelization strategy used to implement the preconditioned conjugate gradient (PCG) method to compute the electric potential is discussed in Sec. 2.4. The near-ideal speedup obtained with 128 GPUs is presented in Sec. 3.5 for a mesothermal plume simulation with xenon ions.

2.1. Octree construction for PIC-DSMC

An octree is a three-dimensional tree data structure that is used to bin particles into nodes, wherein each node spans a certain spatial volume of the computational domain. The entire domain represents the root of the tree, and the root undergoes recursive division until a predetermined criterion, usually dictated by the dominant length scales, is satisfied to generate the leaf nodes. The leaf nodes are analogous to the computational cells in a uniform grid. If the problem size is large, the domain is divided into multiple roots and each root undergoes recursive division to give a forest of octrees (FOT). As mentioned earlier, the modeling of ion or hall thruster plasma plumes is governed by the DSMC collisions and the electric field computed using PIC, which have different length scales and therefore separate octree division criteria. For the DSMC collisions, the root is recursively sub-divided until the final leaf node size is less than the local mean free path, but, for the electric field PIC computations, the local Debye length must be used as the octree division criterion. However,

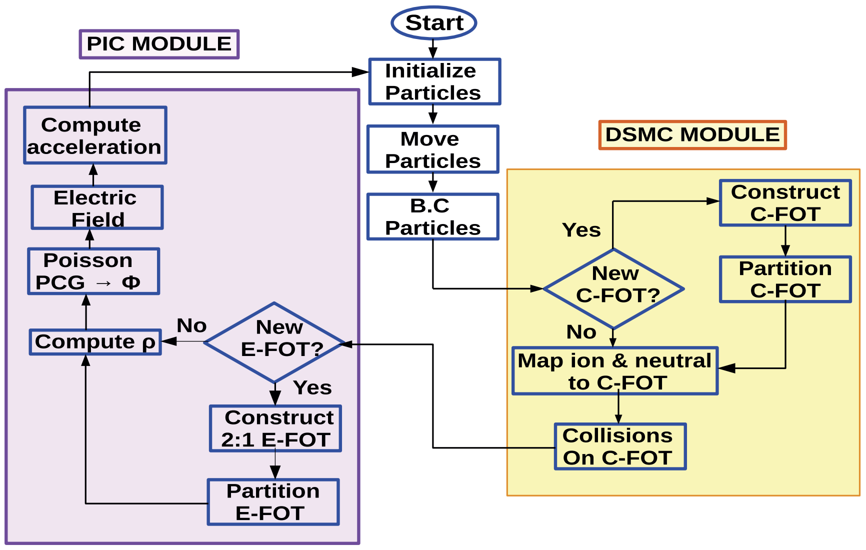


Fig. 1. Flowchart for the PIC-DSMC framework in CHAOS.

for the plume simulations of our interest, the local Debye length increases by a factor of five from the thruster exit to the leading edge of the plume, and the electron number density and temperature at the thruster exit are such that, at a given location, the local Debye length is at least two orders of magnitude smaller than the local mean free path required for collisions. The leaf nodes of the FOT can satisfy only one refinement criterion, and therefore to satisfy the disparate length scales of the two coupled physical process without compromising on the computational efficiency, we use two linear FOTs, one for the PIC computations constructed using the local Debye length criterion called the E-FOT, and the second for the DSMC collisions constructed using the local mean free path criterion, called the C-FOT. Such an implementation with multiple FOTs has been previously implemented by Korkut et al. [2], but the FOTs were stored using a pointer-based data structure and the Boltzmann relation was assumed to obtain the electric potential on the E-FOT. In this work, we employ the computationally efficient linearized Morton-Z ordered C- and E-FOTs to couple the DSMC and PIC modules in CHAOS. Even though particles are mapped to two different FOTs every timestep, the use of two FOTs is efficient since the mapping procedure has been optimized to use fast bit-wise calculations and locational keys for linearized FOTs [55].

2.2. 2:1 Implementation for E-FOT

Since the electric field computation in PIC requires the solution of elliptic partial differential equations, additional spatial constraints are imposed on the E-FOT to achieve an accurate solution. This spatial criterion ensures that all the leaf nodes are, at maximum, only one level coarser than the neighboring leaf nodes and is known as the 2:1 balance [41,62]. The 2:1 constraint has been implemented in many other works that involve solving partial differential equations using multi-grid approaches [41,44,45,63], as well as using iterative procedures to ensure ease of flux calculations at the interface between cells at different refinement levels [47,64]. The implementation of the 2:1 constraint is illustrated using two E-quadtrees shown in Fig. 2, where trees 0 and 1 are constructed by processors 0 and 1, respectively. In CHAOS, the 2:1 balance condition is enforced by comparing the leaf level of each leaf node with its face neighbor's leaf level. Face neighbors are the neighboring leaf nodes that share a common interface. It can be observed in Fig. 2 that the leaf node 11 in tree 1 has three bottom face neighbors, namely 6, 8, and 10, and they do not satisfy the 2:1 balance because leaf node 11 is two levels coarser than its smallest face neighbors, leaf nodes 6 and 8. Similarly, leaf node 2 of quadtree 0, contained in Processor 0, is two levels coarser than its smallest right face neighbors, 5 and 6, contained in Processor 1. Since the trees, 0 and 1, are stored on different processors, the 2:1 constraint is implemented on the E-FOT in two stages, namely, a local and global balance stage.

In the local balance stage, each processor imposes the 2:1 restraint on the leaf nodes that belong to the octrees in its sub-domain, i.e., processors 0 and 1 independently enforce the 2:1 on the trees 0 and 1, respectively. The first step in this stage is to determine the face neighbor leaf nodes and their leaf levels within the same octree. Morton order and the locational keys [65] are exploited to find the face neighbors within the same octree, in six directions, namely, left, right, top, bottom, front, and back. The code snippet to compute the face neighbors using Morton encoding and locational key [55] is given in Appendix A. When comparing the leaf level of face neighbors within the same tree, known as local face neighbors, the leaf nodes that do not satisfy the balance constraint are flagged and refined until they are only one level coarser than their finest local face neighbor. Therefore, leaf node 11 in quadtree 1 is refined until it is only one level coarser than its smallest face neighbors, 6 and 8. The quadtree structure after the local balance stage is completed is shown in Fig. 3(a).

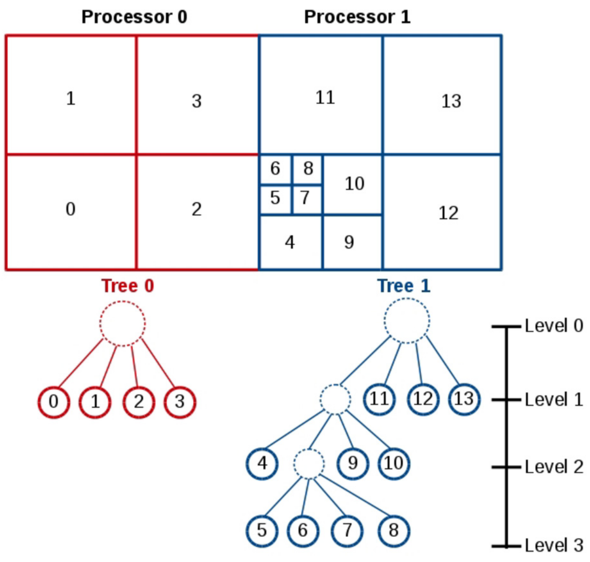
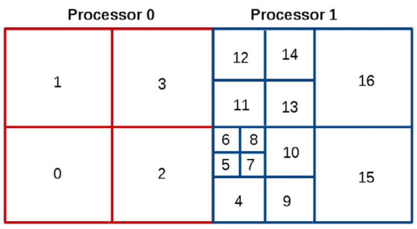
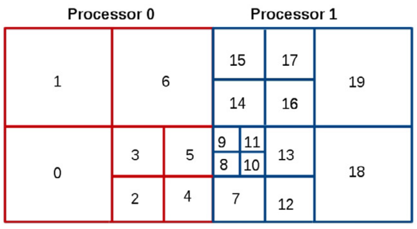


Fig. 2. A forest of E-quadtrees constructed by Processor 0 and 1, that does not satisfy the 2:1 balance.



(a) Forest of quadtrees which satisfy 2:1 balance constraint across face neighbors within the same tree.



(b) The final forest of quadtrees after with 2:1 balance across face neighbors throughout the domain.

Fig. 3. E-quadtree structure generated with 2:1 balance restraint.

But the local 2:1 implementation stage does not ensure that the leaf node 2 in quadtree 0 will satisfy the 2:1 constraint with face neighbors 5 and 6 stored on a different processor. Therefore, during the global balance stage, inter-processor MPI communication of the neighboring leaf node's information is implemented. Each processor determines the number of leaf nodes at the boundary of its sub-domain, and forms a list of face neighbor leaf IDs that belong to a different processor. These face neighbor leaf IDs are also determined using bit-wise Morton encoding and the locational key feature [55,65]. If the level difference between any boundary leaf node, e.g., leaf node 2, and its face neighbor from a different processor, leaf nodes 5 and 6, is greater than one, then the coarser leaf node is refined until the 2:1 criterion is satisfied. The final quadtree structure after the global balance stage is shown in Fig. 3(b). However, the global balance stage, may again lead to a level imbalance between the newly refined leaf nodes with local face neighbors within the same tree, the effect of which would propagate to leaf nodes in other processors, known as the *ripple effect* [62]. Therefore, an iterative local-global cycle is implemented to ensure that all the leaf nodes throughout the domain satisfy this balance criterion with their respective face neighbors on both local and global levels. The number of local-global cycles depends on the variation of local Debye length. For this work, a maximum of three such cycles were required to obtain a 2:1 E-FOT.

2.3. Domain decomposition and face neighbor storage

After the C- and E-FOT are constructed, their respective leaf nodes are assigned a computational weight to quantify the run-time required to perform the computations. For the DSMC module, since the simulation run-time is most sensitive to particle number density, in the absence of immersed bodies [55], the leaf nodes of the C-FOT are assigned weights based on their respective number densities and collision frequencies. Since the electric field solver dominates the PIC module and the electric potential is evaluated on all the leaf nodes, each E-FOT leaf node is assigned a computational weight of one. Finally, the leaf nodes of FOTs are partitioned such that the respective total computational weights of both, the C- and the E-FOT, are load balanced across all processors. Since this partitioning is performed at the leaf level, some of the leaf nodes may need to be transferred from the processor that they were constructed on to the one they are assigned to for

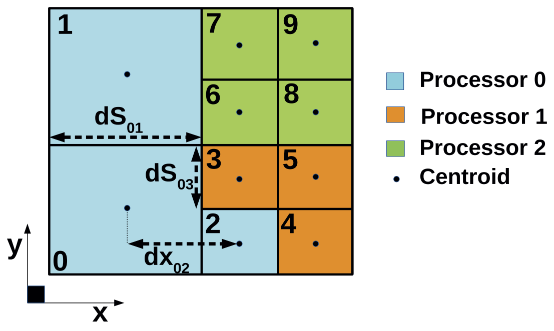


Fig. 4. Domain decomposition of a quadtree for illustration of 3D algorithms.

computations. For example, let us say that the 20 leaf nodes of the final E-FOT, shown earlier in Fig. 3(b), are to be equally divided between the two processors, i.e., leaf nodes 0 to 9 and 10 to 19 are assigned to Processors 0 and 1, respectively, for PIC computations. But, leaf nodes 7, 8, 9 that belong to tree 1 are stored in Processor 1, and must be communicated to Processor 0 during domain decomposition to achieve load balance. A similar leaf node communication among processors is performed when the C-FOT is partitioned using the computational weights assigned to its respective leaf nodes. Thus, for the plasma plume simulations, the number of total leaf nodes in the C- and E-FOT will be different due to different sub-division criteria and length scales, and since the leaf nodes are partitioned among the same processors to achieve load balance on their respective FOTs, the sub-domain contained in a processor for the DSMC computations will not overlap with the sub-domain for PIC calculations. However, since the collisions on the C-FOT and the electric-field calculations on the E-FOT are independent and coupled only through the particles, the overlap of the sub-domains, i.e., the portion of the computational domain contained by the processor, for both DSMC and PIC is not necessary. The methodology for partitioning the linearized Z-ordered FOT is described in detail in Ref. [55].

The PIC module, unlike DSMC, requires each E-FOT leaf node to store the list of its face neighbor leaf Ids in order to determine the potential gradient and the resulting forces that move the charged particles. Therefore, when the leaf nodes of the E-FOT are partitioned, their corresponding face neighbor ID list, computed by the 2:1 balance subroutine, is also transferred from the processor that constructed the octree to the assigned processor after partitioning. The leaf nodes with face neighbors that belong to a different processor are called Z-boundary leaf nodes and their face neighbors are called ‘ghost’ leaf nodes or ‘ghost’ neighbors. Each processor, P_i, flags the Z-boundary leaf nodes and determines the list of processors, P_g, which contains the corresponding ghost neighbors. A *communication link* is set-up between processors that share the partitioned boundaries, such that, each processor can transfer the information of the Z-boundary leaf nodes to the list of neighboring processors stored as P_g, as well as receive information of the ghost neighbors from them. It will be shown in Sec. 2.4.2 that this *communication link* across neighboring processors is used at every iteration by the Poisson solver. Since, the boundaries of the sub-domain remain fixed after partitioning, the list of neighboring processors, and the list of Z-boundary as well as ghost leaf nodes is not determined every timestep, unless the domain is re-partitioned. For an evolving flow, the FOTs are destroyed, re-constructed and re-partitioned periodically, the interval for which typically ranges from 100 to 10,000 time-steps depending on the transient characteristics and is a user-input. For this work, the E-FOT is re-constructed and re-partitioned every 100 timesteps. In addition, due to the modularized structure of the code, the C- and E-FOTs can have different adaptation intervals. After the steady state is achieved, all the computations are performed on the last FOTs that are constructed.

Consider a schematic of a two-dimensional Z-ordered E-quadtree shown in Fig. 4 with global Morton Ids. A weight of one is assigned to each leaf node, after which the tree is equally partitioned among three processors in this illustration. It can be seen that processor 2 has three Z-boundary leaf nodes, 6, 7 and 8, that have ghost face neighbors in a different processor. Processor 2 determines that leaf nodes 6 and 7 share the same ghost neighbor on the left, i.e., leaf node 1 from processor 0, and that the bottom ghost neighbors of leaf nodes 6 and 8 are 3 and 5, respectively, from processor 1. Processor 2, therefore, stores processors 0 and 1 as its neighbors in an array for data transfer. Along with the neighboring processor information, the number of ghost neighbors in each neighboring processor is determined, thus setting up the *communication link*. Also the arrays to store the physical values, such as, the potential ϕ_{ghost}, of these ghost nodes are allocated. In the PIC module, each processor transfers the ϕ values of the Z-boundary leaf nodes to the respective neighboring processors and receives the array of the ghost neighbors ϕ values using the *communication link*.

2.4. PIC method in CHAOS

The electric potential, ϕ, is calculated by solving Poisson’s equation,

$$\nabla^2\phi = -\frac{\rho}{\epsilon_o} \tag{1}$$

where ϵ_o is the permittivity of free space and ρ is the total charge density computed for all leaf nodes of the E-FOT using the nearest-grid point (NGP) method [14] and is equal to the difference between number densities of ions, n_i, and electrons, n_e, multiplied by their respective charges as,

$$\rho = e(Zn_i - n_e). \tag{2}$$

The induced electric field is obtained by taking the gradient of ϕ which is then applied as an external force on the charged particles in the PIC simulations. As the ion and electron spatial distributions evolve in time, Eqs. (1) and (2) are resolved at every time step in the PIC module of CHAOS. The numerical approach used to obtain the electric potential and electric field is described in this section.

2.4.1. GPU computation of charge density

To compute the nearest-grid point [14] charge density distribution, the charged particles are mapped to the E-FOT leaf nodes. A kernel, equivalent to a subroutine in C++, is launched on each GPU with CUDA threads equal to the number of ions and electrons in its sub-domain. Each CUDA thread determines the leaf node for the particle assigned to it, using their position coordinates and fast bitwise computations described in our recent work [55]. Since the mapping of a particle is independent of other particles, this thread-independent procedure is efficiently parallelized by the multi-core GPU. After mapping, if the leaf node that the particle was mapped to, resides in a different GPU's sub-domain, then the particle data is communicated to the destination GPU using MPI-CUDA communication. That is, if particle data in GPU-1 is mapped to a leaf node in GPU-2, then the particle data is first transferred from GPU-1 to its host, CPU-1, using *cudaMemcpy*, followed by an *MPI* communication between CPU-1 and CPU-2, which, in turn, transfers the data to GPU-2 using *cudaMemcpy*. Note that, each CPU or MPI-rank has one GPU device assigned to it. After mapping and particle communication, particles are distributed among the processors based on the E-FOT leaf nodes and the number density of ions and electrons for each leaf node are computed based on the E-FOT leaf ID of the particles [55]. Finally, each CUDA thread computes the total charge density of the leaf node by solving Eq. (2) and all the GPUs concurrently determine the charge density distribution in their respective sub-domains.

2.4.2. Parallel implementation of preconditioned conjugate gradient Poisson solvers

The electrostatic Poisson's equation, given in Eq. (1) is solved on the 2:1 E-FOT using a cell-centered finite volume (FV) approach. It is known that the integral form of Eq. (1), upon employing the divergence theorem, gives [1,66],

$$\oint_S \nabla\phi \cdot \hat{n} dS = -\int_\Omega \frac{\rho}{\epsilon_o} d\Omega, \tag{3}$$

where, the integral is calculated over the surface S and volume Ω of the control volume which in this case is the E-FOT leaf node. Discretizing the above equation for the ith leaf node of the three-dimensional E-FOT, we obtain,

$$\sum_{k=0}^{(k<N_{fi})} \nabla\phi_{ik} \cdot dS_{ik} = -\frac{\rho_i}{\epsilon_o} dV_i, \tag{4}$$

where, N_{fi} is the number of face neighbors of the ith leaf node, dS_{ik} is the face area shared between leaf node i and its kth face neighbor, ρ_i is the leaf centered charge density computed previously, and dV_i is the leaf node volume. In a 2:1 octree, a leaf node may have a maximum of four face neighbors for each of the six faces, i.e., a maximum of 24 face neighbors (maximum value of $N_{fi} = 24$), unlike a uniform grid cell that can have only one face neighbor for every face. The gradient, $\nabla\phi_{ik}$, at the interface between leaf node i and its kth face neighbor, leaf node j, is approximated using the central difference scheme,

$$\nabla\phi_{ik} = \frac{\phi_j(k) - \phi_i}{dx_{ij}}, \; k \in \{0 - N_{fi}\} \tag{5}$$

where, ϕ_i and $\phi_j(k)$ are the leaf-centered values for leaf node i and its kth face neighbor, leaf node j, respectively, and dx_{ij} is the perpendicular distance between the centroid of leaf node i and j across the shared face. For example, the discretized FV equation (4) for leaf node 0 shown in Fig. 4, using Eq. (5) to compute the gradient, is written as

$$\left(\frac{\phi_2 - \phi_0}{dx_{02}}\right) dS_{02} + \left(\frac{\phi_3 - \phi_0}{dx_{03}}\right) dS_{03} + \left(\frac{\phi_1 - \phi_0}{dy_{01}}\right) dS_{01} = -\frac{\rho_0}{\epsilon_o} dV_0 \tag{6}$$

where, all the ϕ's, ρ_0, and dV_0 are leaf-centered values of electric potential, charge density, and leaf volume, respectively, dx_{02} and dx_{03} are the perpendicular x-distance between the centroids of leaf node 0 and its right face neighbors, 2 and 3, respectively, dy_{01} is the y-distance between the centroids of leaf node 0 and its top face neighbor leaf node 1 and dS_{0j} is the face area shared by leaf node 0 with its respective face neighbor leaf node j. For this example we have assumed that a homogeneous Neumann boundary condition is applied on the domain boundaries adjacent to leaf node 0, i.e.,

$d\phi_{DomainBndry} = 0$. Since the shortest distance between the centroids perpendicular to the shared face, dx_{ij}, is equal to the average cell length of the neighboring leaf nodes for a 2:1 octree, the leaf level information of the respective leaf nodes is used for the computation. This implementation is simple and allows for optimized memory usage on the GPU, because storing the integer array of leaf levels is more efficient compared to storing three arrays of doubles corresponding to the x, y, and z coordinates of their centroids, to compute the distance. Grouping the coefficients of ϕ_0 and its face neighbors, ϕ_j, we get,

$$A_{00}\phi_0 - \sum_{j=0}^{j<N_{f0}} (A_{0j}\phi_j) = \frac{\rho}{\epsilon_o} dV_0 \tag{7}$$

where, $A_{00} = \sum_{j=0}^{j<N_{f0}}(A_{0j})$ and A_{0j} form the non-zero coefficients of row 0 for the $N \times N$ sparse matrix, where, N is the total number of leaf nodes in the domain, as shown in the matrix A of Eq. (8) for the quadtree example in Fig. 4.

$$\begin{array}{c} \\ \textbf{GPU 0} \\ \\ \\ \textbf{GPU 1} \\ \\ \\ \\ \textbf{GPU 2} \\ \\ \end{array}
\begin{pmatrix}
A_{00} & A_{01} & A_{02} & [A_{03}] & 0 & 0 & 0 & 0 & 0 & 0 \\
A_{10} & A_{11} & 0 & 0 & 0 & 0 & [A_{16}] & [A_{17}] & 0 & 0 \\
A_{20} & 0 & A_{22} & [A_{23}] & [A_{24}] & 0 & 0 & 0 & 0 & 0 \\
[A_{30}] & 0 & [A_{32}] & A_{33} & 0 & A_{35} & [A_{36}] & 0 & 0 & 0 \\
0 & 0 & [A_{42}] & 0 & A_{44} & A_{45} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & A_{53} & A_{54} & A_{55} & 0 & 0 & [A_{58}] & 0 \\
0 & [A_{61}] & 0 & [A_{63}] & 0 & 0 & A_{66} & A_{67} & A_{68} & 0 \\
0 & [A_{71}] & 0 & 0 & 0 & 0 & A_{76} & A_{77} & 0 & A_{79} \\
0 & 0 & 0 & 0 & 0 & [A_{85}] & A_{86} & 0 & A_{88} & A_{89} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & A_{97} & A_{98} & A_{99}
\end{pmatrix}
\begin{pmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \\ \phi_9 \end{pmatrix}
=
\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{pmatrix} \tag{8}$$

Note that each GPU computes the coefficients of the rows corresponding to the leaf nodes in its sub-domain, as illustrated in Eq. (8). However, the computation of some coefficients, highlighted with square parentheses in the matrix of Eq. (8), requires leaf level information of corresponding ghost neighbors, which were identified during the partitioning step. That is, coefficient A_{03} for leaf node 0 in row 0, depends on the leaf level information of its ghost neighbor, leaf node 3 that belongs to Processor 1 as shown in Fig. 4. Therefore, the leaf levels of the Z-boundary leaf nodes are communicated among neighboring processors using the communication link discussed previously in Sec. 2.3. Additionally, each GPU stores only the non-zero elements of the sub-matrix it contains using the compressed sparse row format, and the matrix is not determined every timestep but only when the E-FOT is constructed or re-constructed. The partial differential equation (1) is thus transformed into a system of linear equations of the form $Ax = b$ as shown in Eq. (8) for the 2:1 E-FOT. The vector b is obtained by computing the RHS of Eq. (4) for all the leaf nodes and including the known boundary conditions, depending on the problem set-up. It can be seen from Eq. (8) that the matrix is sparse and by construction, it is also symmetric, i.e., $(A_{ij} = A_{ji})$, and has positive values for the diagonal elements. Such a symmetric, positive definite matrix can be inverted using a number of iterative numerical recipes. The implementation and communication steps involved in the steepest descent method used in this work is discussed next.

The preconditioned conjugate gradient method (PCG) [67,68] is employed in this work, where to obtain a converged solution with fewer iterations, the algebraic equation, $Ax = b$ is multiplied with a preconditioner, M^{-1}, and the modified equation $M^{-1}Ax = M^{-1}b$ is solved. In CHAOS, we use the diagonal elements of the matrix as a preconditioner, $Diag(A) = M$, which is simple to implement and parallelize using GPUs. The specific MPI-CUDA communications employed in CHAOS and parallelization strategies implemented for the iterative PCG solver are given in Algorithm 1 [67,68]. The GPUs execute the PCG algorithm on their respective sub-domains, by employing large numbers of CUDA threads that concurrently evaluate the equations for a given leaf node, i. The two most important operations involved in Algorithm 1 are sparse-matrix vector multiplication (SPMV) and dot products of vectors. It can be seen that SPMV is executed in steps 4, 5, 12, and 18 of the algorithm, but each GPU stores only a part of the matrix and the vector. For example consider the evaluation of SPMV, $A\phi$, for the system of equations in Eq. (8) for the quadtree example in Fig. 4. GPU 0 computes the product sum, $\sum_{j=0}^{j<N_{fi}} A_{ij}\phi_j$, where the subscript i corresponds to the leaf nodes 0–2 in its sub-domain, and the subscript j are their respective face neighbors. This summation involves computing the product of the matrix coefficient with the ϕ value of a ghost neighbor that is stored in the neighboring processor, i.e., to compute the summation for leaf node 0, the product $A_{03} * \phi_3$ requires the ϕ_3 value of ghost neighbor, leaf node 3. To obtain information from these ghost leaf nodes, the ϕ_{ghost} values of the identified ghost neighbors are communicated between neighboring processors before SPMV using the communication link

Algorithm 1 Conjugate gradient with diagonal preconditioner on multiple GPUs.

```
 1: procedure INITIALIZATION:
 2:     Initialize $\phi$ for all leaf nodes
 3:     Communication $\phi_{ghost}$ : $\text{GPU}_{source} \xrightarrow{CudaMemcpy} \text{CPU}_{source} \xrightarrow{MPI} \text{CPU}_{dest} \xrightarrow{CudaMemcpy} \text{GPU}_{dest}$
 4:     Kernel to Compute : $r = b - A\phi$ ... ($A\phi$ is SPMV)
 5:     Kernel to Compute : $d = M_{inv} \cdot r$, where, $M = Diag(A)$ ... ($M_{inv} \cdot r$ is SPMV)
 6:     Async. Communication of $d_{ghost}$ : $\text{GPU}_{source} \xrightarrow{CudaMemcpy} \text{CPU}_{source} \xrightarrow{MPI} \text{CPU}_{dest} \xrightarrow{CudaMemcpy} \text{GPU}_{dest}$
 7:     Compute local $\delta_{new} = r \cdot d$ .. (Dot product of doubles local using cublas and global sum from MPI_AllReduce)
 8:     Initialize $\delta_o = \delta_{new}$
 9: end procedure
10: procedure ITERATIVE SOLVER
11:     while $i < i_{max}$ && $\delta_{new} > \epsilon^2 \delta_o$ : do
12:         Kernel to Compute : $q = Ad$ (q for local leaf nodes, using SPMV for $Ad$)
13:         Compute $\gamma = d \cdot q$ (locally on the GPU using cublasDdot, then use MPI_AllReduce to obtain global sum)
14:         Compute $\alpha = \delta_{new}/\gamma$ on the CPU
15:         Kernel to Update $\phi = \phi + \alpha d$
16:         Async Comm : $\phi_{ghost}$ : $\text{GPU}_{source} \xrightarrow{CudaMemcpy} \text{CPU}_{source} \xrightarrow{MPI} \text{CPU}_{dest} \xrightarrow{CudaMemcpy} \text{GPU}_{dest}$
17:         Kernel to update residual : $r = r - \alpha q$
18:         Kernel to Compute $S = M_{inv} r$, where, $M_{inv} = Diag(A)$ (.. SPMV)
19:         Update $\delta_{old} = \delta_{new}$ on the CPU
20:         Compute $\delta_{new} = r \cdot S$, .. ( dot product using cublasDdot + MPI_AllReduce )
21:         Compute $\beta = \delta_{new}/\delta_{old}$ on the CPU
22:         Async. Communication of $d_{ghost}$ : $\text{GPU}_{source} \xrightarrow{CudaMemcpy} \text{CPU}_{source} \xrightarrow{MPI} \text{CPU}_{dest} \xrightarrow{CudaMemcpy} \text{GPU}_{dest}$
23:         Update iteration counter : $i = i + 1$
24:     end while
25: end procedure
```

set-up during domain decomposition. In this communication, first, each GPU transfers its array of Z-boundary ϕ values to the host CPU, using *cudaMemcpy*. The CPUs communicate the respective values of the ϕ_{Zbndry} array required by the neighboring processors using non-blocking point-to-point MPI communications, and the receiver CPU stores these values in the ϕ_{ghost} array. Finally, the CPUs communicate the ϕ_{ghost} values received from all the neighboring processors to its GPU.

The computation of vector dot products, required in steps 7, 13, and 20 of Algorithm 1, is performed in two steps since each GPU contains only a part of the vectors. In the first step, each GPU computes the partial dot product of the elements corresponding to the N_p leaf nodes in its sub-domain. An in-built function, *cublasDdot*, available in the optimized linear algebraic package *cublas*, is employed to compute this partial sum that is finally stored on the host CPU. Finally, all the partial sums stored on the CPUs are added using the *MPI_AllReduce* function, to obtain the final scalar value on all the processors. Consider a dot product, $g = b \cdot b$, where the b vector is partitioned among three GPUs as shown in Eq. (8). All the GPUs simultaneously compute the partial sum, $g_{loc} = \sum_{i=0}^{i<N_p} b_i \cdot b_i$ for all the N_p leaf nodes in their sub-domains, and store this value on the respective host CPU. In the next step, the partial sums, g_{loc}, from all the CPUs are added to obtain g, which is available on all the CPUs. The code-snippet for the partial and global sum values using MPI-CUDA is shown in Appendix B. The while loop in the algorithm is executed until a converged value for the electric potential is obtained, and the convergence criteria used for the plasma plume calculations are such that, $\delta_{new} < (1 \times 10^{-7}\delta_o)$, where δ is the residual of the preconditioned equation, $M^{-1}Ax = M^{-1}b$, computed in steps 8 and 20 of Algorithm 1. For the simulations presented in Sec. 5, typically 10 to 40 iterations were required for the PCG solver to satisfy the above criteria, where, the value of δ_o ranged from 0.01 to 1 as the plume simulations evolved.

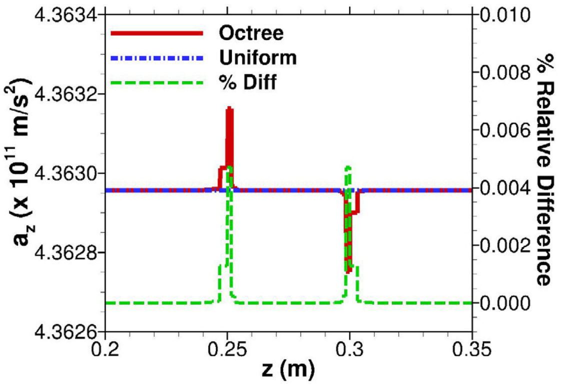
The electric field at the leaf-node center is obtained such that each CUDA thread independently and simultaneously computes the average potential gradient, using Eq. (5), for the corresponding leaf node assigned to it. For example, the electric field in the x-direction, E_x, for leaf node 2 in the 2:1 quadtree shown in Fig. 4 is computed as follows,

$$E_x = (\nabla\phi_{20} + \nabla\phi_{24})/2 \tag{9}$$

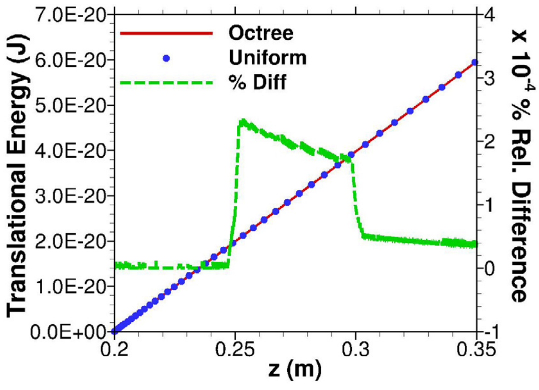
where, $\nabla\phi_{20}$ and $\nabla\phi_{24}$ are obtained by using Eq. (5). This equation reduces to the central differencing scheme for a uniform grid approach. As discussed earlier, the gradient computation requires the value of the face neighbor potential, which are already stored on the GPU during the iterative Poisson solver step. In addition, even the ϕ_{ghost} values are communicated during the iterative PCG algorithm. Therefore no additional communications are required for the computation of the electric field. The force for all the charged particles is obtained based on the cell-centered electric field value of the leaf node that it belongs to. Finally, the particle positions and velocities are updated using the leap-frog scheme [14,69] and the new charge distribution at the next timestep is computed.

3. PIC module validation, error analysis, and performance studies

To analyze the effect of the nearest-grid point (NGP) charge distribution and the 2:1 octree grid in CHAOS we perform four validation cases, similar to those performed by Duras et al. [70], and Averkin et al. [71]. The solutions from the octree-



(a) Comparison of z-acceleration component computed from uniform and octree structure.



(b) Comparison of electron translational energy computed from uniform and octree structure.

Fig. 5. Effect of uniform versus 2:1 octree on electron acceleration and translational energy.

based PIC module are compared with a uniform grid, obtained by enforcing a 1:1 octree grid, as well as the analytical solutions.

3.1. Comparison of single particle trajectory on a 2:1 octree and uniform grid

In the first case, a single electron is initialized at the center of a $(0.4 \times 0.4 \times 0.4)$ m domain with an initial velocity of 10,000 m/s in the z direction and zero velocity in the x and y directions. A Dirichlet boundary condition of $\phi = 0$ and 1 V is implemented at the $z = 0$ m and $z = 0.4$ m boundaries, respectively, and a homogeneous Neumann boundary condition is implemented at the other boundaries. To analyze the effect of grid-refinement on the electron trajectory, we perform two simulations, one with a uniform 1:1 octree where all the leaf nodes are at the same level with $\Delta x = 3.125$ mm, and the second simulation with a 2:1 octree where, refinement is enforced from $z = 0.25$ to $z = 0.3$ m such that the leaf node size in this region is $\Delta x/2$, i.e., 1.5625 mm. Due to the Dirichlet boundary conditions of 0 and 1 V at the z-boundaries, a linear profile is expected for the electric potential, resulting in a constant electric field and acceleration. It is known that no self-force errors [70] are generated in a uniform grid approach, however, for an octree approach, a self-force error may be expected due to the loss in second-order accuracy of the Poisson solver at the coarse-to-fine and fine-to-coarse interface regions, that is, at $z = 0.25$ to $z = 0.3$ m for this case.

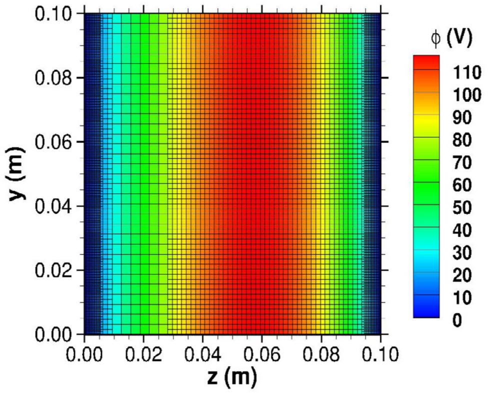
To analyze the effect of this self-force error on the charged particle trajectory, the acceleration and translational energy are recorded as the electron travels from $z = 0.2$ m to 0.35 m. The values obtained from the 1:1 uniform grid and 2:1 octree simulation are compared in Figs. 5(a) and 5(b), respectively. In comparison with the uniform grid case, the electron acceleration computed from the 2:1 octree grid is found to increase at $z = 0.25$ m and decrease at $z = 0.3$ m as a result of the change in the refinement level. However, the relative difference in the electron acceleration is less than 0.005% throughout the trajectory. Similarly, the electron translational energy computed from the uniform and 2:1 octree agree within 5×10^{-4}% as shown in Fig. 5(b). From this study we conclude that, even though the electron acceleration undergoes a change due to self-force at the interface between two levels of refinement, the relative difference in electron acceleration and translational energy is less than 0.1% and does not significantly influence the accuracy of the PIC simulation compared to the uniform grid.

3.2. Rate of convergence studies

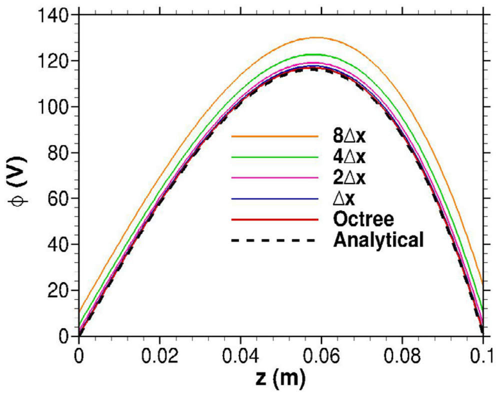
In the second test case, we perform simulations with a known stationary charge density variation to test the convergence rate of the electric potential and field with decrease in the cell size. For this study we initialize a $(0.1 \times 0.1 \times 0.1)$ m domain with a linearly varying ion number density, such that, $n_i = n_o(z/L)$, where, n_i is the ion number density at a given location (x, y, z), $n_o = 1 \times 10^{13}/\text{m}^3$, and L is the size of the cube, i.e., 0.1 m in this case. A Dirichlet boundary condition of $\phi = 0$ V is implemented at $z = 0$ and 0.1 m boundaries and a homogeneous Neumann boundary condition is implemented at the remaining boundaries. The analytical expression for the static potential in the z-direction for the stationary charge density variation and boundary conditions is

$$\phi = \frac{-qn_o}{6L\epsilon_o}(z^3 - zL^2). \tag{10a}$$

The electric field is zero at $z = L/\sqrt{3}$, where the potential is maximum and is equal to $\phi_{max} = 116.12$ V. Four static simulations are performed with a 1:1 uniform grid by decreasing the cell size by a factor of two in each case from $8\Delta x$



(a) Potential variation along the $y-z$ slice at the center of the domain with 2:1 octree, that has a level difference of four between the coarsest and finest leaf node.



(b) Effect of cell size on the potential variation compared with analytical results.

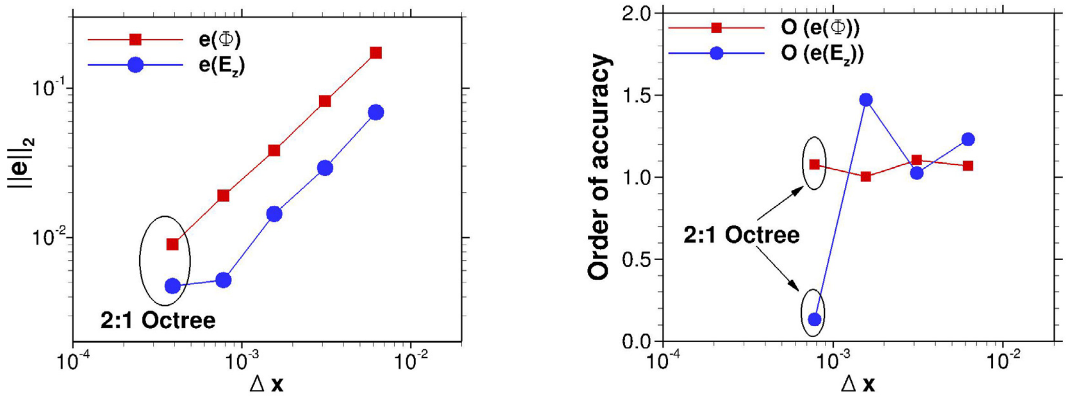
Fig. 6. Comparison of electric potential with the analytical solutions for a linearly varying ion charge density distribution.

to $\Delta x = 0.78125$ mm. For comparison, the same test case is also performed with a 2:1 octree, with a leaf node size of 0.3906 mm, which is much smaller than the local Debye length near the boundaries to accurately capture the potential gradient. Everywhere else within the domain, the leaf node size is less than the local Debye length, computed assuming an ion temperature of 2 eV, and all the simulations are performed with 88.7 million ion particles. The variation of potential on the y–z slice at the center of the domain, obtained from the octree simulation is shown in Fig. 6(a) along with the 2:1 octree structure. The ϕ in the z-direction, extracted along a line passing through the center of the y–z slice, is compared with that obtained from the uniform grid simulations in Fig. 6(b). It can be seen that as the cell size is decreased for the uniform grid case, the solution obtained approaches the analytical solution. Since the octree structure inherently has a smaller leaf node size near the boundaries, it can accurately capture the large field gradients at the boundary as well as the potential variation within the domain, as shown by the good agreement with the analytical solutions.

To verify the order of accuracy for the potential and electric field solver implemented in CHAOS, we first compute the L_2-norm of the relative error in potential and electric field, defined as $e_2(f_h) = \frac{\|f_h - f_e\|_2}{\|f_e\|_2}$, where f_h is the solution, f, obtained with a grid size of h, and, f_e is the exact analytical solution. The L_2-norm of a function, f, is given as $\|f\|_2 = \sqrt{(1/N)\sum_{i=1}^{i=N} f_i^2}$, where N is the total grid points [72]. The decrease in the L_2-norm of the relative error in $\phi(x, y, z)$ and E_z with decrease in the cell size is shown in Fig. 7(a), where the error for the 2:1 octree grid case is highlighted. It can be seen that the $e_2(E_z)$ is almost equal for the 1:1 case with $\Delta x = 0.78125$ mm and the 2:1 octree case, meaning that the electric field is converged. The rate of convergence, p, is obtained by evaluating $p = \frac{\ln(e_2(f_{2h}) - e_2(f_h))}{\ln 2}$, where $e_2(f_{2h})$ and $e_2(f_h)$ denote the L_2-norm of the relative error in f for a grid with cell size of $2h$ and h, respectively. The rate of convergence for the potential and electric field is shown in Fig. 7(b), where it can be seen that the order of accuracy for ϕ is greater than 1, which is typical for a second-order accurate method with first-order Neumann boundary conditions [72], and for E_z the order of accuracy is greater than unity for all the cases with 1:1 uniform grid. Since the electric field computed with the smallest refined grid and the octree are the same, as observed from the same $e(E_z)$ in Fig. 7(a), the order of accuracy for the electric field decreases to 0.1, showing that a converged electric field is computed from the 2:1 octree structure. Thus, we conclude from this study that the rate of convergence obtained for the potential and electric field is between 1 and 2.

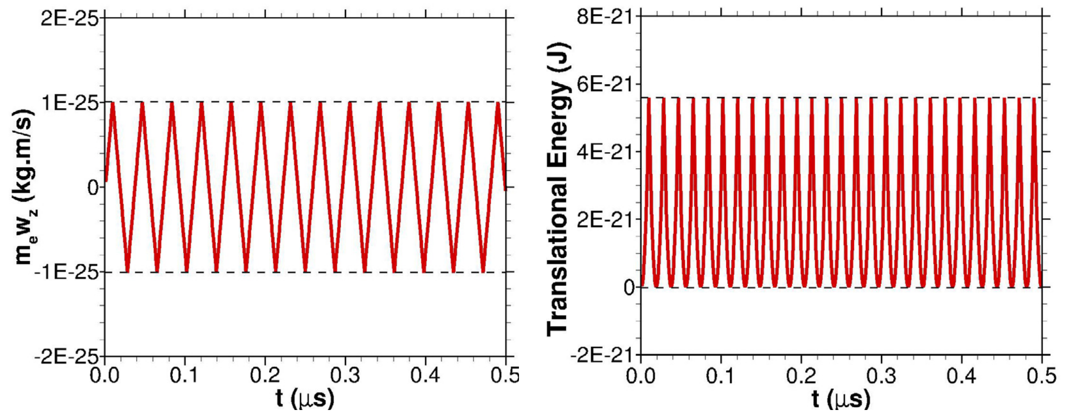
3.3. Momentum and energy conservation studies

In the third case, we study the effect of the 2:1 octree and the particle movement algorithm on the conservation of momentum and energy, similar to the studies performed by Averkin et al. [71]. Ions are initialized in a $(0.1 \times 0.1 \times 0.1)$ m domain with a linearly varying number density of $n_i = n_o\big(1 - \frac{2}{L}|\frac{L}{2} - z|\big)$, where, L is the size of the cubic domain, and $n_o = 1 \times 10^{13}$ m^{-3}. The boundary condition is similar to that used for the rate of convergence studies discussed in Sec. 3.2. An electron particle with zero initial velocity is placed at the center of the domain, $(0.05, 0.05, 0.049)$, and due to the potential induced by the stationary ion charge density distribution, the electron will be trapped and oscillate within the domain. As time progresses, if the oscillations in its momentum and energy have equal amplitude, then these properties are conserved. The plasma frequency for this case is $\omega_{pe} = 1.785 \times 10^8$ rad/s, assuming an electron temperature of 2 eV, and to accurately capture the electron oscillations we use a timestep of $\Delta t = 0.05\omega_{pe}^{-1} = 0.28$ ns. A 2:1 octree, similar to that constructed for the convergence studies, is used to discretize the domain, and the explicit leap-frog particle integration scheme is implemented [14]. The computational domain consists of 24 million ion particles, 0.64 million leaf nodes with



(a) L_2-norm of the relative error in ϕ and E_z (b) Convergence rate of ϕ and E_z.

Fig. 7. Effect of cell size on L_2-norm of the relative error and rate of convergence of ϕ and E_z.



(a) Electron $z-$momentum (b) Electron translational energy.

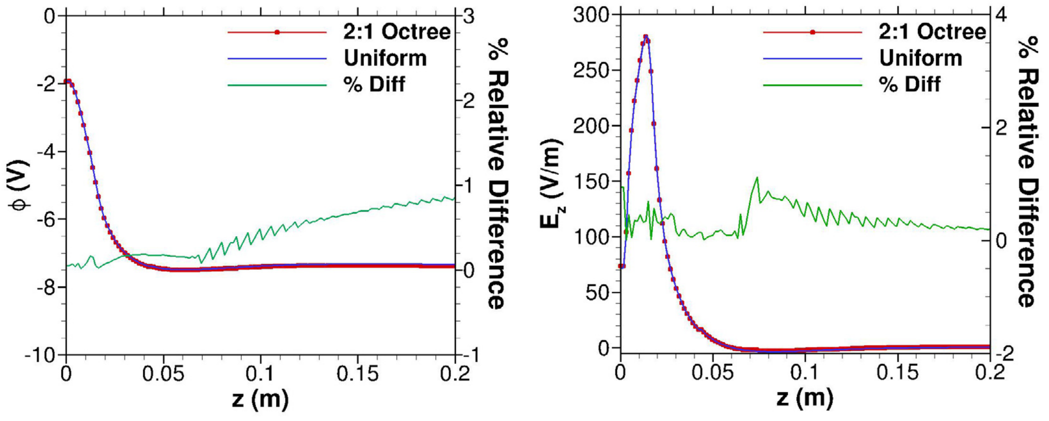
Fig. 8. Variation in electron z-momentum and translational energy with time showing conservation.

the smallest leaf node of 0.3 mm, and coarsest leaf node of size 1.25 mm, and the simulation is performed using 16 GPUs. The z-velocity and translational energy of the electron up to 0.5 µs obtained from the simulation are shown in Figs. 8(a) and 8(b), respectively. It can be seen that the z-velocity and translational energy of the electron has equal amplitude, demonstrating that the momentum and translational energy are conserved in the simulation.

Numerical heating studies are also performed similar to Averkin et al. [71] for which the electrons and ions are initialized in a $(0.1 \times 0.1 \times 0.1)$ m cubic domain with number densities of $n_i = n_e = 1 \times 10^{13}/\text{m}^3$, and a Maxwellian velocity distribution with $T_i = T_e = 2$ eV. A Dirichlet boundary condition of 0 V is imposed on all boundaries of the domain and periodic boundary condition for the particles is implemented, with a timestep of 2.8×10^{-10} s. Two test cases are performed, one with a 1:1 uniform grid of $\Delta x = 0.3125$ mm and the second with a 2:1 octree where a refinement is imposed for $z >$0.05 m, and at least 100 particles per cell were used for both the cases. The average energy of the system was conserved up to $t\omega_{pe} = 10{,}000$ and found to increase by 0.008% compared to the initial energy of $(3/2)k_bT$ in both the cases.

3.4. Comparison of computational efficiency of a 2:1 octree and uniform grid plume simulation

Finally, to test the computational accuracy and efficiency for cases with large density variations, we perform a collisionless mesothermal plume simulation using co-located xenon ion and electron sources, described as case 2 in Sec. 4, for a 2:1 octree and a 1:1 uniform grid. The domain size for this verification case was limited to $(0.4 \times 0.4 \times 0.4)$ m to reduce the computational cost inherent to uniform grid computations. The plasma source of radius 0.625 m is located at $(0.2, 0.2, 0.0)$ m and all the remaining input parameters are the same as that mentioned in Table 2 for case 2. Since the 2:1 octree satisfies the local Debye length, the smallest cell size is 1.5625 mm within the plume and increases to 6.25 mm beyond the beam-front, while for the uniform grid, all the cell sizes are equal to 1.5625 mm. The two simulations are performed up to $t\omega_{peo} = 100$, i.e., 560 ns, using 16 Tesla K20 GPUs, and to avoid statistical noise involved in the instantaneous results, we sample the flow-field macroparameters up to 560 ns, or 2,000 timesteps. Since sampling requires a static grid,



(a) Potential variation along the plume axis. (b) Streamwise electric field variation along the plume axis.

Fig. 9. Comparison of sampled potential and electric field computed using the uniform and 2:1 octree along the plume axis.

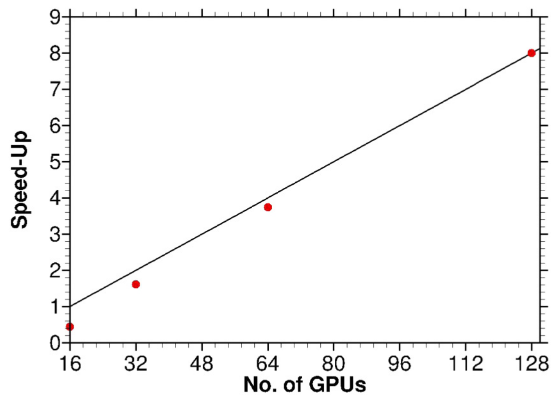


Fig. 10. Strong scaling studies for case 2.

we construct an initial 2:1 octree that satisfies the Debye length criteria up to 560 ns and turn off the dynamic adaptation. The comparison of the sampled potential and z-electric field along the plume axis obtained using the 2:1 octree and the uniform grid are shown in Figs. 9(a) and 9(b), respectively. It can be seen that the sampled macroparameters from the octree and the uniform grid simulations agree within 2% and the maximum relative error in electric field is 1.2% at $z = 0.07$ m, where the octree level undergoes a change in refinement. Since the octree is refined only in the regions with smaller Debye length, the total number of leaf nodes required for the simulation is 1.085 million, compared to the 16.77 million cells required by the uniform grid simulation. For the initial 560 ns of the plume evolution with 17 million total particles in both simulations, the run-time of the octree simulation is found to be 8 minutes, which is nearly 50 times faster than the uniform grid simulation that required 7.13 hours. However, it must be noted that the number of leaf nodes in the octree simulation will dynamically increase with the advancement of the plume, which in turn will increase the computational time. Nevertheless, compared to the octree, the three-dimensional uniform grid PIC simulations will still be ten times more expensive.

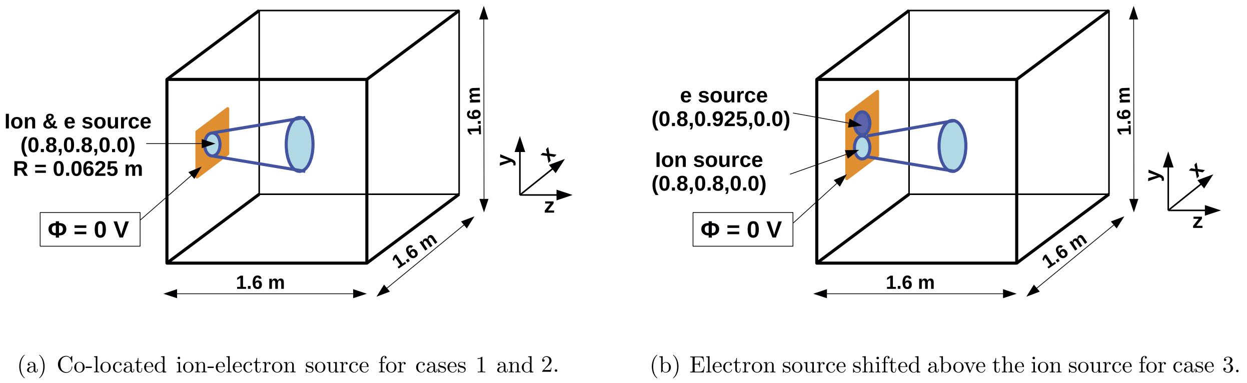
3.5. Scaling studies

Strong scaling studies were performed on a collisionless mesothermal xenon plume, defined as case 2 in Sec. 4, by varying the number of GPUs used to simulate the plume from 16 to 128. The case 2 input parameters, given in Table 2 of Sec. 4, are used for all the scaling simulations in order to keep the problem size identical. Each leaf node was assigned a computational weight of unity, i.e., the leaf nodes of the forest were equally divided among the GPUs during the simulation. The scaling simulations were performed on the XStream supercomputer which contains eight Nvidia Tesla K80 GPUs per node and the run-time was obtained by performing the simulation from 18,000 to 20,000 timesteps, when the plume has sufficient particles in the domain to test the load balancing.

Fig. 10 compares the speed-up from CHAOS with the ideal scaling, which is computed using the 128 GPU run-time as the baseline. It can be seen that CHAOS scales efficiently with increase in the number of GPUs. Since the leaf nodes are

Table 1
Profiling studies.

Subroutine	Percentage time
Particle mapping + MPI-CUDA particle communication	40%
Poisson solver	20%
Ghost leaf node communication in the Poisson solver	20%
Boundary condition and particle movement	7.5%
Electric field and acceleration	7%
Re-partitioning and miscellaneous	5.5%



(a) Co-located ion-electron source for cases 1 and 2. (b) Electron source shifted above the ion source for case 3.

Fig. 11. Simulation set-up for the mesothermal plasma plume cases with domain size of (1.6 × 1.6 × 1.6) m. A Dirichlet boundary condition of $\phi = 0$ V is implemented in the highlighted orange region at the inlet plane, surrounding the radial sources. At all other boundaries and within the circular source region, a homogeneous Neumann boundary conditions is applied. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

equally divided and the plume is confined for case 2, the partitioning is not truly load balanced especially for simulations with fewer GPUs. That is, the leaf nodes in the plume region contained more particles compared to the leaf nodes in the surrounding region and therefore assigning equal weight to the leaf nodes would not distribute the computational weight equally. It can be seen that this load imbalance is more pronounced for the 16 and 32 GPU simulations. However, when the same problem is divided among more processors, the core-region of the plume is equally distributed, and therefore, the total computational weight is load balanced even though the varying particle density is not accounted for while assigning computational weight to the leaf nodes. The scaling increases to near-ideal with the increase in the number of GPUs.

Profiling studies were performed for the 128-GPU case using the TAU-profiler and the relative contribution of each subroutine towards the computational cost is shown in Table 1. For 2,000 timesteps with 12.5 million leaf nodes and 80 particles per leaf node within the plume, the run-time was approximately 11 minutes, which is at least an order of magnitude faster than the two-dimensional formulation with a uniform grid that employs CPUs [58]. As seen from Table 1, the particle sorting and communication sub-routine requires 40% of the total computational time. However, within this subroutine the relative time required for mapping particles to leaf nodes is negligible due to the fast bit-wise Morton encoding method, and the main contribution to the 40% computational cost is due to the MPI-CUDA particle communication and sorting of arrays based on the leaf node ID. The pre-conditioned conjugate gradient method used to solve Poisson's equation requires 20% of the total time, but the communication of the ghost neighbor information, performed every iteration, requires 20% due to the use of MPI_WAIT command, that waits until all the non-blocking MPI_Isend and MPI_Irecv communications are performed. The particle movement and boundary condition subroutines require 7% each, and the re-partitioning subroutine requires 5% of the total computational cost.

4. Simulation set-up and case description

Having validated the PIC module by performing canonical test cases and analyzing its computational performance, we now study the effects of hollow cathode location on the electron kinetics using two types of electron source configurations. In the first configuration, the electron source is co-located with the ion source, as shown in Fig. 11(a), while in the second configuration, the electron source is shifted above the thruster exit, in the y-direction, as shown in Fig. 11(b). To systematically analyze the effect of ion mass and electron source location, three cases are performed. In case 1, the ions are introduced with mass equal to that of the protons and the electron source is co-located with the ion source, similar to the mesothermal plume simulations performed by Wang et al. [58] and Hu et al. [59,60]. For case 2, heavier xenon ions are introduced from the co-located ion and electron source, shown in Fig. 11(a), to understand the effect of using a higher ion mass. Finally, case 3 is a realistic simulation, with the electron source shifted above the xenon ion source using the configuration set-up shown in Fig. 11(b). Since collisions are not required to model a mesothermal plasma plume [58], the DSMC module in CHAOS is not executed.

Table 2
Parameters for mesothermal plasma plume simulations[a].

Cases	1 – Proton plasma co-located e$^-$ source	2 – Xenon plasma co-located e$^-$ source	3 – Xenon plasma shifted e$^-$ source
n_{io} (m^{-3})	4.0×10^{13}	4.0×10^{13}	1.0×10^{13}
m_i/m_e	1836	239,669.5	239,669.5
ω_{pio} rad/s	8.33×10^6	7.295×10^5	7.295×10^5
No. of particles at t_{final}[b]	164.9 million	165.1 million	64.5 million
Total leaf nodes at t_{final}[b]	7.09 million	4.45 million	1.8 million

[a] $n_{eo} = 1.0 \times 10^{13}/\text{m}^3$, $\omega_{peo} = 1.78 \times 10^8$ rad/s, $v_{teo} = 592{,}892$ m/s, $T_{eo} = 2$ eV $= 23{,}210$ K, domain size is $(1.6 \times 1.6 \times 1.6)$ m, $T_{io} = 0.01T_{eo}$, $\rho_o = en_{eo}$, and $v_{ibeam} = 30{,}000$ m/s.
[b] $t_{final} = 3000\omega_{peo}^{-1} = 16.8$ µs.

The initial input parameters for the PIC simulations of cases 1 to 3 are given in Table 2. To facilitate qualitative comparison of the three-dimensional plume characteristics with the 2-D simulations performed by Wang et al. [58] and Hu et al. [59,60], the ratio of initial ion temperature, T_{io}, to initial electron temperature, T_{eo}, used in all the cases is 0.01. For all the cases, electrons are initialized with a temperature of $T_{eo} = 2$ eV and an initial number density, n_{eo}, of $1.0 \times 10^{13}/\text{m}^3$. These selected values for T_{eo} and n_{eo} result in an initial Debye length, $\lambda_{do} = 3.32 \times 10^{-3}$ m and initial electron plasma frequency, $\omega_{peo} = 1.78 \times 10^8$ rad/s. The beam velocity, v_{ibeam}, for cases 1, 2, and 3, is taken to be $0.05v_{teo}$ [59], where v_{teo} is the initial thermal velocity of the electrons corresponding to T_{eo}. The ratio of initial ion to electron number densities, $n_{io}/n_{eo} = 4$, is such that the respective current densities are equal at the co-located source [58] for cases 1 and 2. For the shifted electron source in case 3, the ratio of n_{io}/n_{eo} is initially equal to unity, such that the charge density at the respective sources are equal. The radius, R, of the ion and electron sources in both the configurations, shown in Fig. 11, is equal to $18.5\lambda_{do}$, i.e., 0.0625 m, representative of real thruster devices [1], and the size of the three-dimensional domain is chosen to be equal to $500\lambda_{do}$ to avoid boundary-effects [58]. For all three cases, the ion source center is located at $(0.8, 0.8, 0.0)$ m, and only for case 3, the electron source is centered at $(0.8, 0.925, 0.0)$ with radius equal to 0.0625 m, which is shifted from the ion source center by one diameter length in the y-direction.

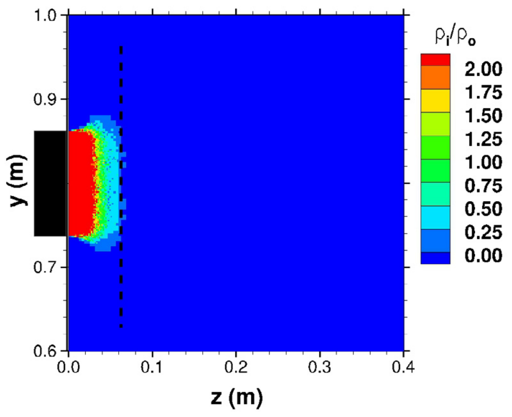
The simulations are performed for a duration of $t\omega_{peo} = 3000$, which is equal to 16.8 µs, in order to resolve the electron time scales [58]. Due to the difference in the ion mass, the ion plasma period at the end of the simulation for case 1 is $t\omega_{pio} = 140$, while for cases 2 and 3, it is $t\omega_{pio} = 12.25$. The plasma time scales for case 1 are similar to the mesothermal plasma simulations of Hu et al. [60]. A uniform timestep resolution of $\Delta t = 2.8 \times 10^{-10}$ s is used for both ions and electrons in all the cases, such that, $\Delta t \times \omega_{peo} = 0.05$ [58]. At every timestep, ions are emitted with a Maxwellian distribution corresponding to temperature T_{io}, and a beam velocity, v_{ibeam}, along the z-direction. The electrons, however, are introduced every timestep with a stationary Maxwellian distribution with temperature T_{eo}, and no bulk component. To model the thruster surface surrounding the ion and electron source, a Dirichlet boundary condition with $\phi = 0$ V is applied in the region surrounding the sources, as shown by the highlighted orange region in Fig. 11. On all the other boundaries and within the circular source region, a homogeneous Neumann boundary condition, $d\phi/dn = 0$, is implemented to model the zero electric field for plume expansion into the vacuum of space. An outflow boundary condition is used for the particles, wherein, the particles are removed if they cross the computational boundaries. The electric potential, electric field, and the resulting acceleration of the charged particles are computed on the E-FOT using the methodology discussed in Sec. 2.

5. Results and discussion

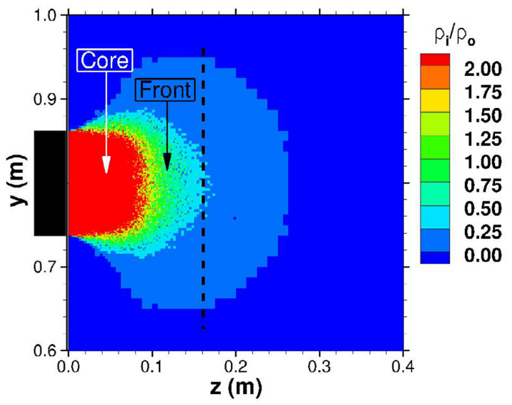
The unsteady evolution of the plume as well as the effect of ion-to-electron mass ratio and electron source location on electron kinetics is discussed in the following subsections. To capture the evolving plume, the octree is dynamically destroyed and reconstructed every 100 timesteps, such that the leaf node size is less than the local Debye length. Due to the higher ion mass, the ion plasma frequency, ω_{pio} for cases 2 and 3 is an order of magnitude lower than that for case 1. Even though the simulations are performed for the same electron time period, $t\omega_{peo} = 3000$, the ion plasma time periods, $t\omega_{pio}$, are different. Our objective, however, is to analyze the plume dynamics and electron kinetics on the electron time-scale, such that, the beam-front of both the plumes are at the same location for comparison.

5.1. Plume characteristics and electron kinetics for case 1

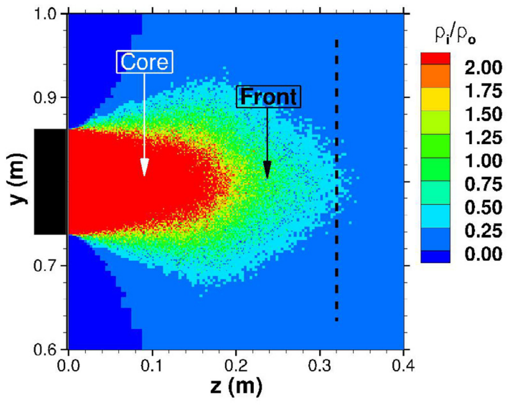
The instantaneous macroparameters of the unsteady plume are extracted along the y–z plane passing through the center of the domain. The spatial variation of the ion charge density, normalized by the initial charge density, ρ_o, is shown in Fig. 12 for plasma times $t\omega_{peo} = 200$, 600, 1500, and 3000. With the progression of time, the leading edge of the ion beam, or the beam-front, advances in the streamwise direction. The beam-front at $t\omega_{peo} = 200$, 600, 1500, and 3000 is located at $z = 0.06$, 0.16, 0.32, and 0.45 m, respectively, as shown by the dotted line in Figs. 12(a) to 12(d). The extent of the ion beam, from the thruster exit to the beam-front, can be divided into two regions, namely, the core region where $\rho_i > 2\rho_o$, and the front region, where the ion charge density gradually decreases from $2\rho_o$ to $0.2\rho_o$ at the beam-front. At $t\omega_{peo} = 600$ and 1500, the core region extends from the thruster exit to $z = 0.1$ and 0.2 m, respectively, which is approximately two-thirds



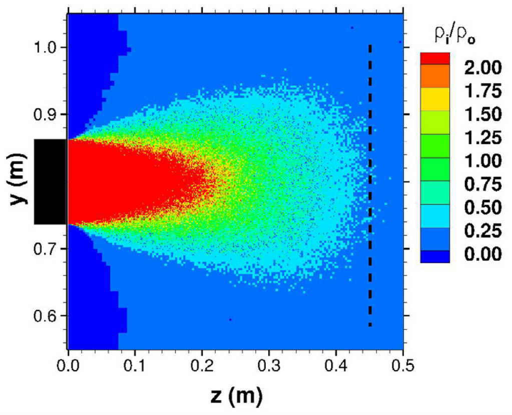
(a) $t\omega_{peo}$=200



(b) $t\omega_{peo}$=600



(c) $t\omega_{peo}$=1500



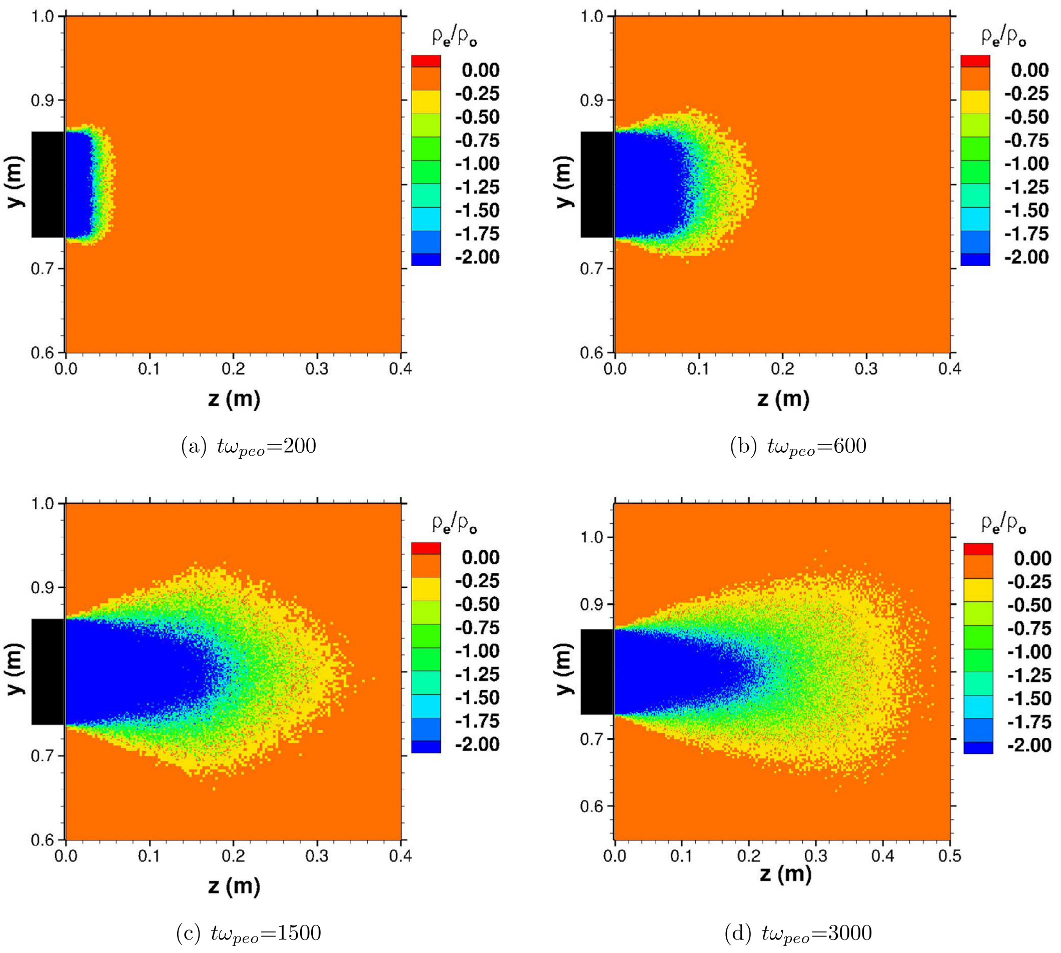
(d) $t\omega_{peo}$=3000

Fig. 12. Transient ion charge density for case 1 along the y–z plane extracted at the center of the domain, normalized by ρ_o. Dashed line indicates the notional location of the beam-front.

of the beam-front distance of 0.16 and 0.32 m, as shown in Figs. 12(b) and 12(c), respectively. Note that, in the absence of acceleration, the beam-front would be at $z = tv_{ibeam} = 0.1011$ m, instead of 0.16 m. This indicates that the average ion velocity may be higher than the initial beam velocity in the front region of the ion beam. At $t\omega_{peo} = 3000$, the ion plume expands further and causes the ion charge density to decrease, as shown in Fig. 12(d). As a result, the fraction of the core-region is now about half the extent of the beam, unlike the two-third fraction observed at previous times.

Similarly, the evolution of the electron charge density, normalized by ρ_o, is shown in Fig. 13 at plasma times $t\omega_{peo} = 200$, 600, 1500, and 3000. At the outset, because the electrons are emitted with a high thermal velocity they overshoot the ion beam, resulting in a net positive charge within the plume. The electrons introduced in subsequent timesteps are then electrostatically trapped by the positively charged plume indicated by the increase in the electron number density within the plume compared to its initial value of ρ_o. This electrostatic coupling between the electrons and ions forces the electrons to collectively travel with the ion beam, as observed from the similarity in the ion and electron charge density variations shown in Figs. 12 and 13, respectively. In particular, from Figs. 13(a)–13(c), we observe that the electron charge density core region, with $\rho_e < -2\rho_o$, extends up to two-thirds of the beam, similar to that observed for the ion charge density distribution. At $t\omega_{peo} = 3000$, shown in Fig. 13(d), the magnitude of the electron charge density decreases beyond $z = 0.22$ m, which is half the distance to the beam-front at $z = 0.45$ m, similar to the ion charge density variation shown earlier in Fig. 12(d), demonstrating that the electrons collectively travel with the ion beam. In addition, a small number of electrons with higher thermal velocity component over-shoot the beam-front, thereby exerting an attractive force on the ions, causing the ion particles at the beam-front to accelerate further and move with streamwise velocity higher than the initial v_{ibeam}.

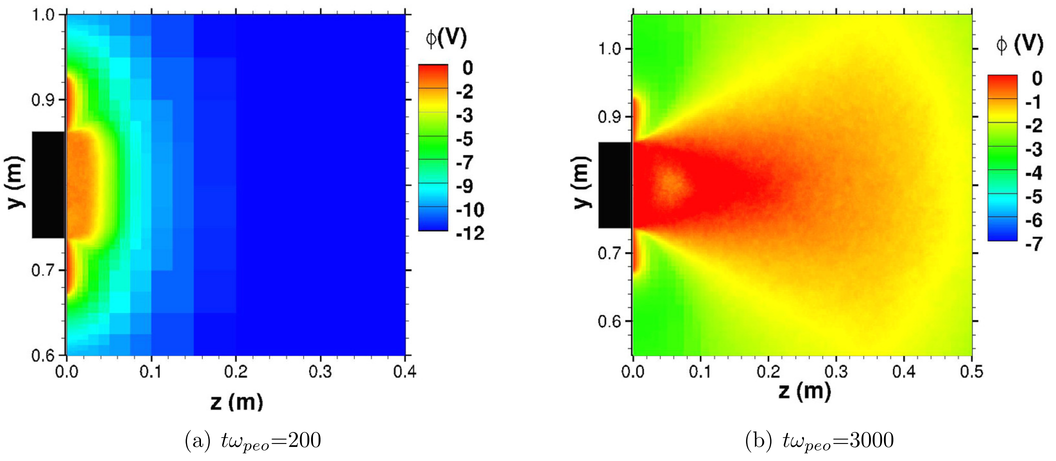
The spatial variation of electric potential, ϕ, at plasma times $t\omega_{peo} = 200$ and 3000 is shown in Figs. 14(a) and 14(b), respectively. The equivalence in the ion and electron charge densities due to electrostatic coupling within the beam tends to create a quasi-neutral plume with potential consistently within the range of 0 to 1 V potential inside the plume core region and $\phi = -2$ V at the beam-front and radial edges of the plume. Such a quasi-neutral plume was also computed by Wang et al. [58,59], using a uniform grid approach for 2D mesothermal proton plasma simulations with a co-located electron–ion



(a) $t\omega_{peo}$=200 (b) $t\omega_{peo}$=600

(c) $t\omega_{peo}$=1500 (d) $t\omega_{peo}$=3000

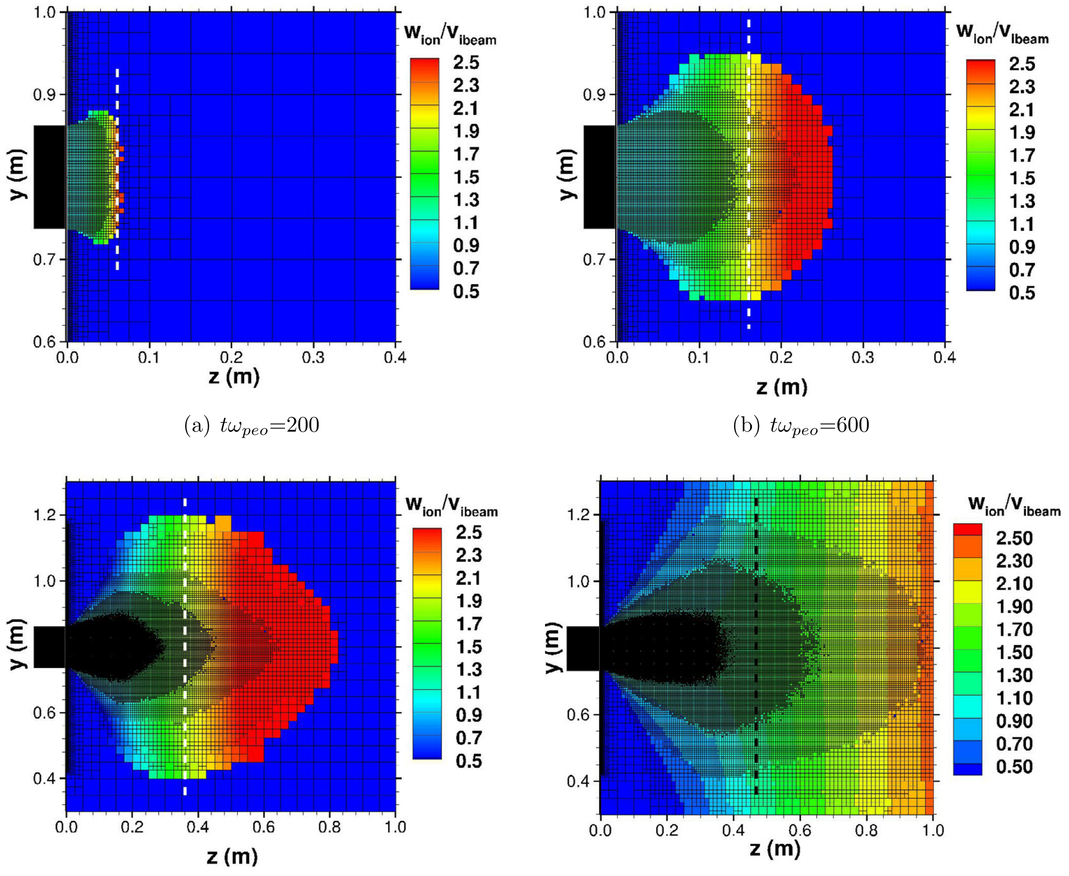
Fig. 13. Transient electron charge density for case 1 along the y–z plane extracted at the center of the domain, normalized by en_{eo}.



(a) $t\omega_{peo}$=200 (b) $t\omega_{peo}$=3000

Fig. 14. Spatial distribution of electric potential along the y–z plane, extracted at the center of the domain.

source. The potential in the region surrounding the ion source is 0 V due to the Dirichlet boundary condition implemented here, as discussed previously in Sec. 4. Outside of the plume, in the far-field, the potential decreases to -6 V near the computational boundary at $t\omega_{peo} = 3000$ or $t = 16.8$ μs, caused by the electrons that have escaped from the trapping due to their higher thermal velocity.



(a) $t\omega_{peo}$=200 (b) $t\omega_{peo}$=600

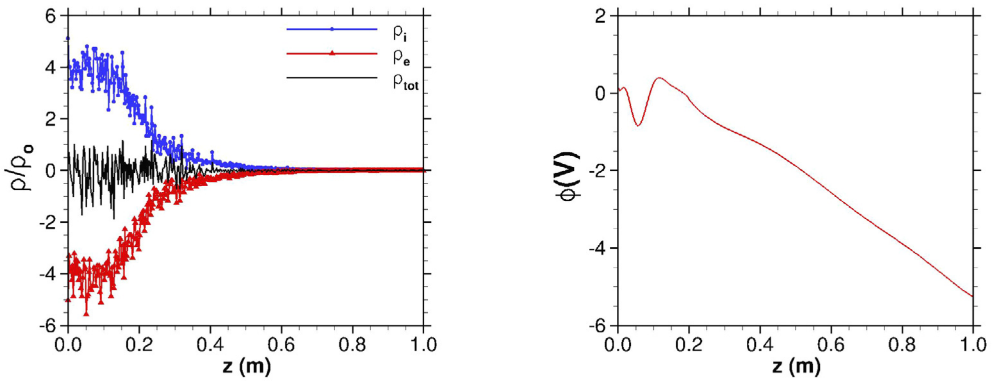
(c) $t\omega_{peo}$=1500 (d) $t\omega_{peo}$=3000

Fig. 15. Spatial distribution of streamwise ion velocity normalized by the initial beam velocity, $v_{ibeam} = 30{,}000$ m/s, along the y–z plane at the center of the domain. The dotted line indicates the location of the ion beam-front.

The instantaneous streamwise ion velocity, w_{ion}, normalized by the initial beam velocity of $v_{ibeam} = 30{,}000$ m/s, for plasma times $t\omega_{peo} = 200$, 600, 1500, and 3000 are shown in Figs. 15(a) to 15(d), respectively, along with an evolving 2:1 octree structure. The dynamic reconstruction and re-partitioning of the E-FOT allows for accurately capturing the evolving local Debye length variations, efficiently satisfying the numerical criteria at all time instances. The smallest leaf node size is 1.5625 mm at the core of the plume, and in the far-field, the leaf node size increases to 0.25 m. As inferred from the ion charge density distribution, the normalized ion velocity within the core region of the plume, with maximum leaf node refinement, is close to one, and the ion velocity increases to $1.8v_{ibeam}$ at the beam-front, as shown in Figs. 15(b) to 15(d). Due to the electrostatic attractive force exerted by the high thermal velocity electrons that overshoot the beam-front, some ions escape from the plume with velocity equal to $2.5v_{ibeam}$. Note that, the charge density of these accelerated ions in the region downstream of the beam-front is found to be much less that $0.1\rho_o$, as shown previously in Fig. 12.

The instantaneous electron, ion, and total charge distributions, extracted along the center-line $(0.8, 0.8, z)$ of the y–z plane, at plasma time $t\omega_{peo} = 3000$, normalized by ρ_o, are shown in Fig. 16(a). The ion and electron charge density distributions along the center-line are equal and opposite, suggesting that the plume has achieved quasi-neutrality. This is also supported by the total charge distribution that fluctuates about neutrality along the axis of the plume. Since the co-located proton plume undergoes expansion, as seen from the ion charge distribution in Fig. 12(d), the magnitude of the ion charge density decreases from the initial charge density of $4\rho_o$ near the thruster exit to $0.2\rho_o$ at the beam-front at $z = 0.45$ m. Corresponding to the quasi-neutral charge density variation, the electric potential, shown in Fig. 16(b) is found to be close to zero at the thruster exit, and decreases to -2 V at the beam-front located at $z = 0.45$ m. The dip in the potential from the thruster exit to $z = 0.12$ m is due to the entrapment of electrons in the ion beam.

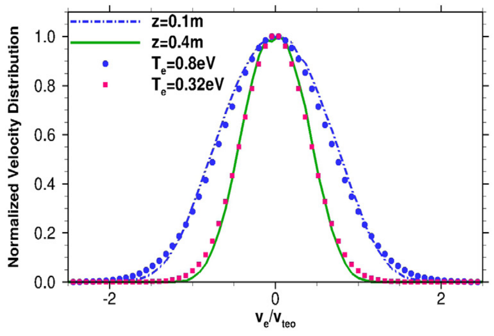
The electron kinetic behavior is analyzed by sampling the electron velocity distribution functions (EVDF) at locations $z = 0.005$, 0.1, and 0.4 m, within 0.05 m radius from the plume-axis $(0.8, 0.8, z)$, for a plasma time of $t\omega_{peo} = 3000$. In Fig. 17(a), we compare the y-EVDF, v_e/v_{teo}, at $z = 0.1$ and 0.4 m with the analytical Maxwell Boltzmann (MB) distributions. The analytical MB distributions are computed for a T_e and velocity shift to generate a distribution similar to that of the sampled distributions. The sampled y-EVDF at $z = 0.1$ and 0.4 m, are found to agree well with the MB distributions for $T_{ey} = 0.8$ and 0.32 eV, respectively. A drop in T_{ex} and T_{ey}, the respective x- and y-component of electron temperature, is



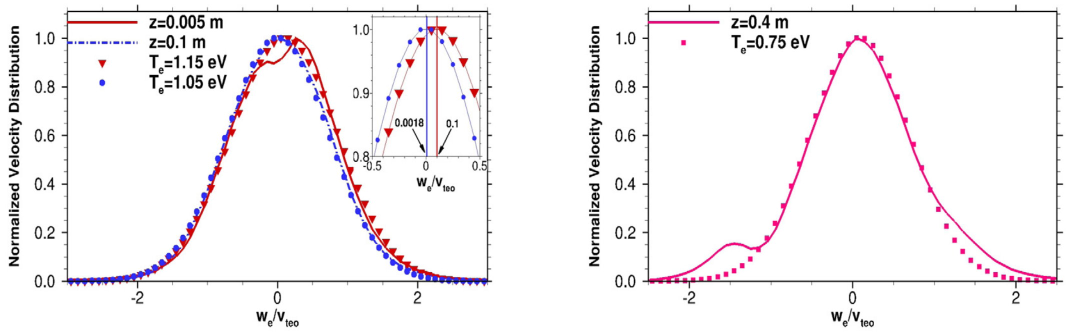
(a) Ion, electron, and total charge distribution normalized by ρ_o=e.n_{eo}, where n_{eo} is given in Table 2.

(b) Instantaneous electric potential.

Fig. 16. Variation of charge distribution and electric potential along the plume center-line, (0.8, 0.8, z), at plasma time $t\omega_{peo} = 3000$.



(a) y-velocity component at z=0.1, (blue), and 0.4 m, (green), respectively.



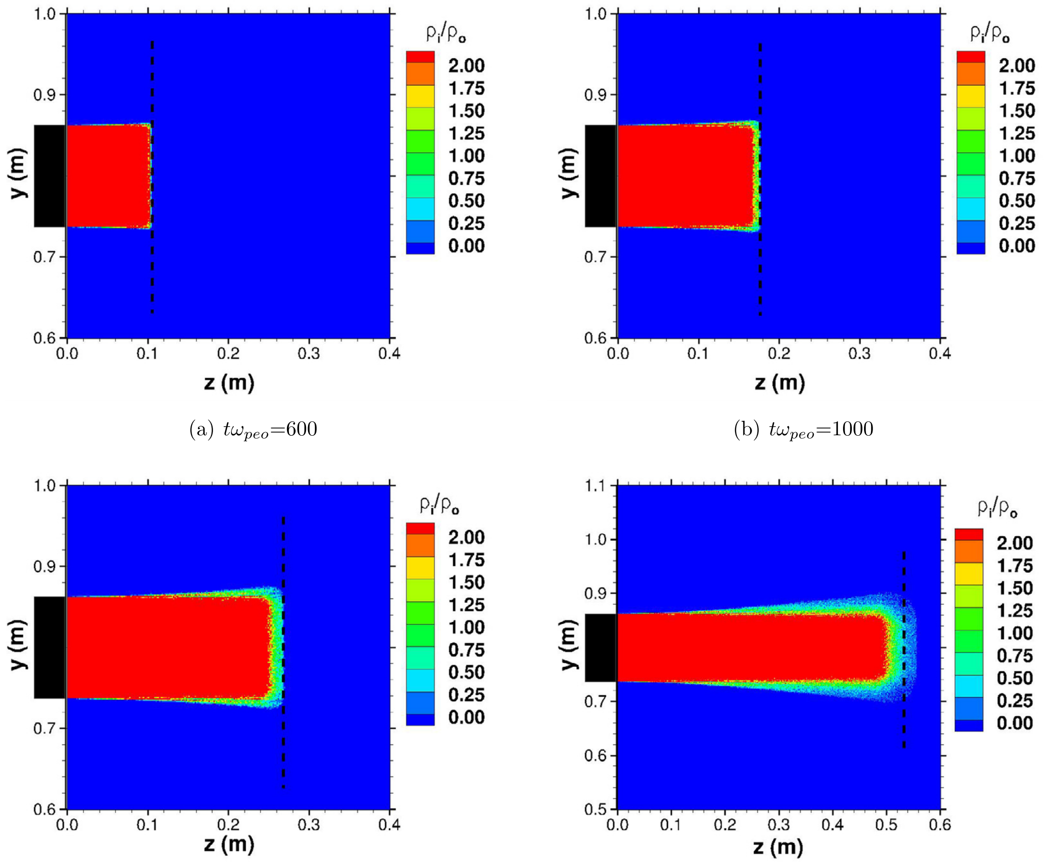
(b) z-velocity component at z=0.005 and 0.1 m, respectively.

(c) z-velocity component at z=0.4 m.

Fig. 17. Comparison of case 1 electron velocity distribution sampled from the simulation (the solid lines), at plasma time $t\omega_{peo} = 3000$, with the analytical distribution (dotted symbols), at locations $z = 0.005$, 0.1, and 0.4 m. The temperature used to obtain the analytical distribution is specified in the legend.

observed downstream from the electron source due to the plume expansion. Note that, the x- and y-EVDF were found to be equal at all the sampled locations, which is a consequence of the axial symmetry of the plume. However, the z-EVDF were not equal to the cross-stream velocity distributions.

The variation in the streamwise temperature, T_{ez}, and bulk velocity, w_e, from the thruster exit to the beam-front is analyzed by comparing the z-EVDF shown in Figs. 17(b) and 17(c). Near the thruster exit, at $z = 0.005$ m, the electrons that were emitted with zero bulk velocity undergo acceleration resulting in an average $w_e = 60$ km/s, as indicated by the MB peak value of $w_e/v_{teo} = 0.1$, shown in Fig. 17(b). This transition from the initial stationary EVDF to an accelerated EVDF results in a kink at the peak of the z-EVDF, which further indicates that the sampled distribution near the thruster exit is not strictly Maxwellian. At $z = 0.1$ m, however, the average w_e decreases to 1 km/s, i.e., $w_e/v_{teo} = 0.0018$, and



(a) $t\omega_{peo}$=600

(b) $t\omega_{peo}$=1000

(c) $t\omega_{peo}$=1500

(d) $t\omega_{peo}$=3000

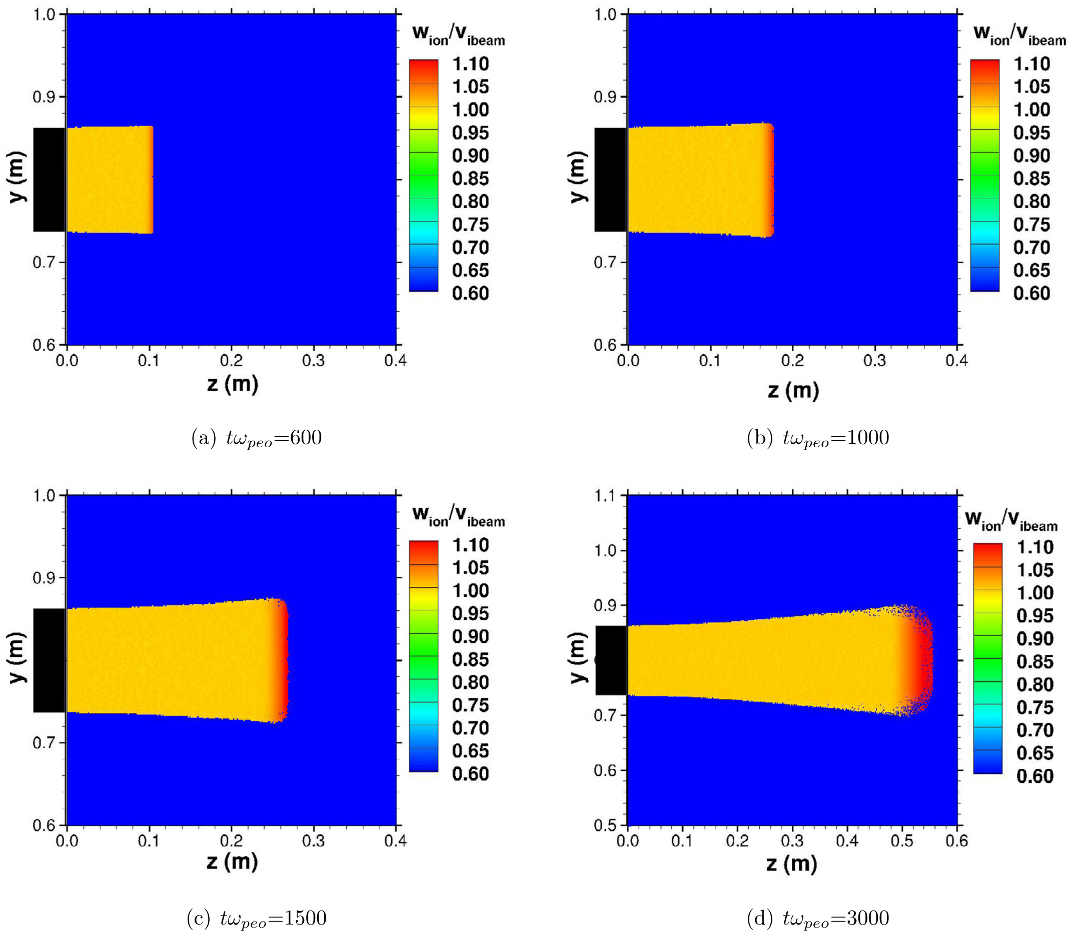
Fig. 18. Transient ion charge density variation, ρ_i, normalized by the initial charge density, ρ_o. The dotted line shows the location of the ion beam-front.

agrees well with an MB distribution corresponding to 1.05 eV, as shown in Fig. 17(b). The electrons accelerated at the thruster-exit are trapped at $z = 0.1$ m by the ion beam, causing some of the electrons to reverse their direction which in turn decreases the net bulk velocity to 1 km/s, as shown by the peak value in Fig. 17(b). This electron trapping indicated by the decrease in bulk velocity is consistent with the dip in the potential observed previously at $z = 0.1$ m in Fig. 16(b). Near the beam-front, at $z = 0.4$ m, the sampled z-EVDF is in close agreement with the MB distribution at $T_{ez} = 0.75$ eV, but it has a secondary peak at $w_e/v_{teo} = -1$, as shown in Fig. 17(c). This enhanced tail in the negative direction is due to the electrostatic attraction of the beam-front electrons towards the plume which prevents them from escaping. The average $w_e = 54$ km/s in this region, which is equal to the ion bulk velocity of $1.8v_{ibeam}$ at the beam-front, confirming that the trapped electrons propagate along with ion beam.

The difference in the y and z-EVDFs, shown in Fig. 17, indicates that the electron temperature for the expanding plume is anisotropic. In particular, T_{ez} is found to be higher than T_{ey} because of the increase in thermal energy caused by the streamwise electron trapping as opposed to the temperature decrease associated with the radial expansion of the plume. In addition, the decrease in electron temperature from the thruster exit to the beam-front for an expanding plume suggests that the use of a single T_e in the Boltzmann relation [1] would not predict the electric field accurately, even for a quasi-neutral plume. Even so, the secondary peak in the EVDF at the beam-front can only be captured by a particle method. Finally, the observed electron trapping, ion–electron coupling mechanism to achieve a quasi-neutral state, as well as the anisotropy of the electron temperature obtained from our 3D octree-based PIC simulations are in qualitative agreement with previous studies [58,59] for the proton plasma plume.

5.2. *Effect of ion mass on mesothermal plume characteristics – case 2*

For case 2, changing the plume ion species from the lighter proton ions to the heavier xenon ions results in a more confined beam. Since the xenon ions are two orders of magnitude heavier than the proton mass, their thermal velocity is ten times smaller than that of the proton mass ions, which leads to the smaller radial spread for case 2 compared to case 1. The evolution of the instantaneous xenon ion charge density for case 2 at plasma times $t\omega_{peo} = 600$, 1000, 1500, and 3000, is shown in Figs. 18(a) to 18(d), respectively. As observed from the location of the dotted-line in Fig. 18, the



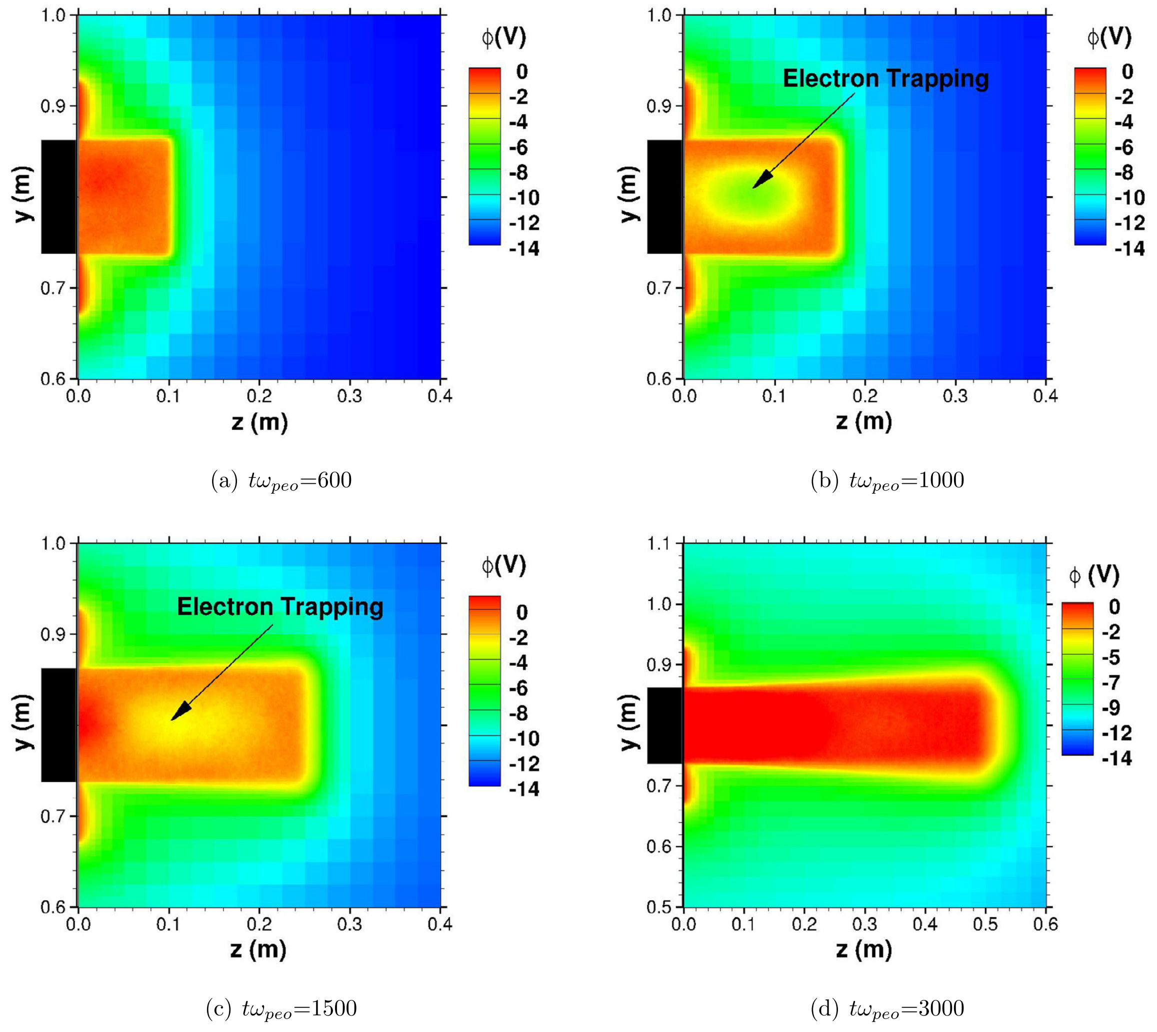
(a) $t\omega_{peo}$=600 (b) $t\omega_{peo}$=1000 (c) $t\omega_{peo}$=1500 (d) $t\omega_{peo}$=3000

Fig. 19. Transient streamwise ion velocity variation, w_{ion}, normalized by the beam velocity, $v_{ibeam} = 30{,}000$ m/s.

beam-front is located at $z = 0.1$, 0.176, 0.27, and 0.54 m, at times, $t = 3.37$, 5.62, 8.43, and 16.85 µs, respectively. We can infer that the streamwise ion velocity, w_{ion}, is approximately 30,000 m/s within the beam, using the expression, $z_{bf} = tw_{ion}$, where the beam-front location, z_{bf}, is obtained from the location of the dotted line, shown in Figs. 18(a) to 18(d), at the respective times, t. It can also be observed that $\rho_i > 2\rho_o$ within most of the plume, and sharply decreases to zero at the beam-front, contrary to the gradual decrease observed in the front one-third region of the lighter ion plume of case 1. This is again attributed to the heavier mass of the xenon ions which do not accelerate as quickly compared to the lighter proton ions, for the same attractive force exerted by the induced electric field. Similar to case 1, the electrons initially overshoot the beam-front leaving behind a positively charged beam, which in turn, traps the electrons introduced in the subsequent timesteps. The magnitude of the electron charge density increases to $4\rho_o$ within the plume, and its spatial variation is found to be similar to that of the ion charge density variation due to the electrostatic electron–ion coupling observed even for the proton plasma.

The variation of the streamwise ion velocity component, w_{ion}, normalized by the beam velocity, v_{ibeam}, is shown in Fig. 19 for plasma times $t\omega_{peo} = 600$, 1000, 1500, and 3000. Consistent with the evolution of the ion charge density, it is observed that the xenon ion velocity is equal to the initial beam velocity of 30,000 m/s within the plume up to the beam-front. At the beam-front, however, due to the attractive force exerted by the electrons that overshoot the beam-front, the maximum ion velocity is found to be $1.1v_{ibeam}$, which is 40% smaller than the $1.8v_{ibeam}$ ion velocity observed at the beam-front for the lighter proton ions in case 1.

Fig. 20 shows the instantaneous electric potential, ϕ, for the evolving xenon plume at plasma times, $t\omega_{peo} = 600$, 1000, 1500, and 3000. Similar to case 1, the electrons emitted at the beginning overshoot the beam-front due to their high thermal velocity, and the resulting positively charged plume electrostatically traps the electrons emitted in the subsequent timesteps. This trapping of electrons increases their number density, such that, the potential within the plume is uniform and quasi-neutral between 0 to 1 V, as shown in Fig. 20(a). But, since the ion beam at $t\omega_{peo} = 1000$ does not expand as much as in case 1, the trapped electrons are confined within a smaller radial width, resulting in an oval-shaped negative potential zone within the plume, as shown in Fig. 20(b). These trapped electrons travel along with the ion beam and eventually mix within the heavy xenon ion beam, such that, at $t\omega_{peo} = 1500$, the negative potential zone within the beam



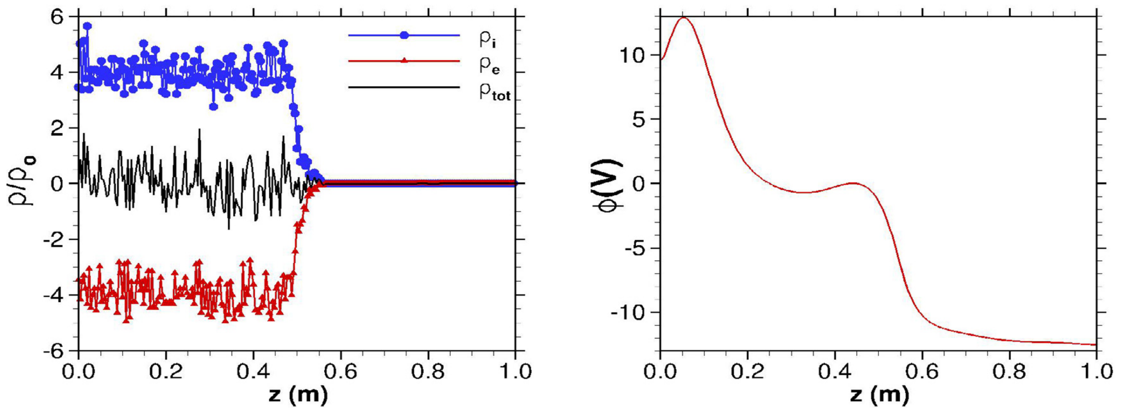
(a) $t\omega_{peo}$=600 (b) $t\omega_{peo}$=1000 (c) $t\omega_{peo}$=1500 (d) $t\omega_{peo}$=3000

Fig. 20. Spatial distribution of the instantaneous electric potential obtained for case 2 along the y–z center-plane.

diffuses, as shown in Fig. 20(c), and additionally, the potential near the thruster reaches quasi-neutrality. The electrons emitted in the subsequent timesteps are not trapped in this quasi-neutral region, and as a result, at $t\omega_{peo} = 3000$, the near-thruster potential is greater than 0 V as shown in Fig. 20(d). At the radial edges and the beam-front, the potential is -3 V, showing that the electrostatic forces cause the electrons to envelop around the ion beam.

The variation of the charge distributions along the plume axis at plasma time $t\omega_{peo} = 3000$, i.e., at $t = 16.8$ μs, is shown in Fig. 21(a). Similar to case 1, the amplitude of the ion and electron charge densities are equal and opposite along the plume axis, and the total charge distribution fluctuates about neutrality, demonstrating that the plume has achieved a quasi-neutral state. The ion charge distribution is observed to rapidly decrease from $4\rho_o$ at $z = 0.47$ m to zero at the beam-front, $z = 0.54$ m, as shown in Fig. 21(a). The corresponding electric potential variation along the plume axis is shown in Fig. 21(b). It can be seen that the potential reaches a maximum of 13 V at $z = 0.05$ m which is a factor of two higher than the maximum ϕ of 5 V observed for case 1 in Fig. 16(b). The decrease in the potential to 0 V at $z = 0.46$ m is due to the trapped electrons within the ion beam. Downstream from the beam-front, for $z > 0.54$ m, the potential gradually decreases to -12.5 V at the boundary due to the electrons that overshoot the beam. Fig. 21(b) shows that the maximum plume potential is 25.5 V higher than the boundary potential.

To analyze the effect of ion mass on the electron kinetic properties, the instantaneous EVDF were sampled at $z = 0.005$, 0.1, and 0.54 m, within a radius of 0.05 m about the plume axis, at $t\omega_{peo} = 3000$, i.e., at $t = 16.8$ μs, as shown in Fig. 22. In Fig. 22(a), the y-velocity distributions sampled at $z = 0.1$ and 0.54 m are compared with analytical MB distributions, obtained using methods similar to that discussed for case 1. From the agreement between the sampled and analytical distributions, we can state that T_{ey} decreases from 1.5 at $z = 0.1$ to 1.1 eV at $z = 0.54$ m due to the expansion of the beam. The T_{ex} and T_{ey} values obtained for case 2 are higher than those from case 1, shown in Fig. 17(a) because the electrons are trapped within a more confined beam-like Xenon ion plume as opposed to the proton plume that has a larger



(a) Variation of ion, electron, and total charge distribution normalized by ρ_o=en_{eo}.

(b) Variation of electric potential along the center-line

Fig. 21. Variation of charge distribution and electric potential along the plume center-line at plasma time $t\omega_{peo} = 3000$ for case 2.

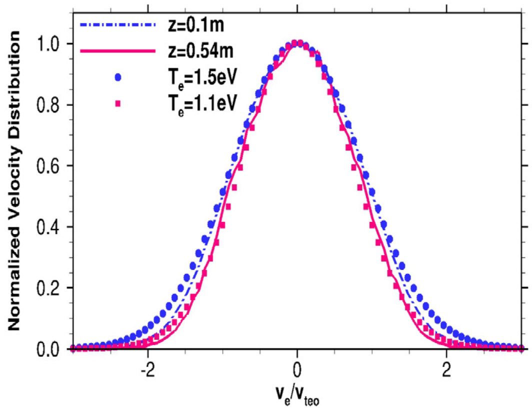
radial spread. This electron trapping within a confined region increases the thermal spread, and therefore the temperature. However, similar to case 1, the electron temperature for case 2 is also anisotropic, meaning that the T_{ez} is not equal to T_{ex} and T_{ey}.

The variation in the local z-EVDF for case 2 from the thruster exit to the beam-front are shown in Figs. 22(b) and 22(c). Close to the thruster exit, at $z = 0.005$ m, the z-EVDF is in close agreement with an MB distribution of $T_{ez} = 2$ eV and average $w_e = 88.9$ km/s, as shown by the peak value of $w_e/v_{teo} = 0.15$ in Fig. 22(b). The electrons that were initialized with a zero bulk velocity at the thruster exit are now accelerated to 88.9 km/s so that they can collectively propagate with the ion beam. This transition in the bulk velocity is captured by the kink in the EVDF near the thruster exit, similar to the near-thruster z-EVDF observed for case 1, in Fig. 17(b). As we move further downstream, at $z = 0.1$ m, the comparison of the sampled z-EVDF with the analytical MB distribution shows that $T_{ez} = 2$ eV is unchanged, but the bulk velocity decreases to 30,000 m/s, as shown by the distribution peak value of $w_e/v_{teo} = 0.05$ in Fig. 22(c). The decrease in the bulk velocity is due to the electron trapping in this region, which is also consistent with the decrease in the electric potential shown previously in Fig. 21(b). Incidentally, the bulk electron velocity of 30,000 m/s in this region is equal to the local streamwise ion velocity, indicating that the trapped electrons collectively travel with the ion beam. At the beam-front, $z = 0.54$ m, the bulk velocity of the electrons is maintained at 30,000 m/s, but the temperature decreases to $T_{ez} = 1.5$ eV as observed from Fig. 22(c) due to expansion at the beam-front. The beam-front T_{ez} is higher than $T_{ey} = 1.1$ eV at the same location, showing that the degree of anisotropy in temperature is $T_{ez}/T_{ex} = 1.36$ for this case. Compared to case 1, the T_{ez} for case 2 at the beam-front is higher by a factor of two, and unlike the secondary enhanced tail observed for the velocity distribution in Fig. 17(c) at the proton plasma beam-front, the case 2 distribution has no evidence of a secondary peak.

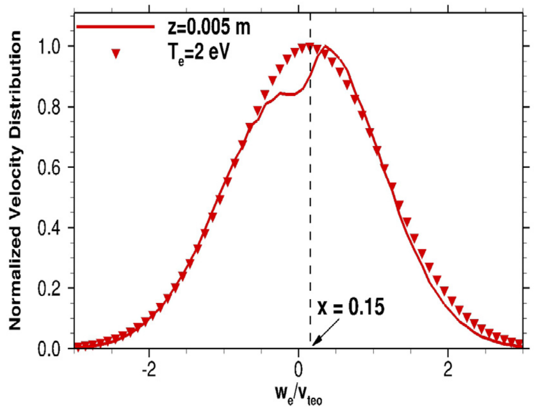
5.3. Neutralization from a shifted electron source – case 3

In this section, we discuss the effect of electron source location on plume dynamics and electron kinetic behavior. For case 3, the ion source is centered at (0.8, 0.8, 0.0) m, and the electron source is shifted along the y-direction, with its center at (0.8, 0.925, 0), highlighted by the black and gray blocks, respectively, at the inlet plane in Fig. 23(a). For all the subsequent figures in this subsection, the black and gray blocks at the inlet plane indicate the ion and electron source location, respectively. The transient evolution of the electron and ion charge density, normalized by $\rho_o = en_{eo}$, is shown in Figs. 23 and 24, respectively, at plasma times $t\omega_{peo} = 600$, 1000, 1500, and 3000.

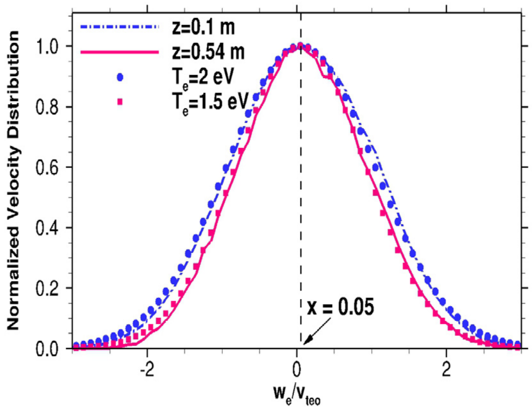
The shift in the electron source location affects the mechanism by which the electrons are trapped within the heavy xenon ion beam. As soon as the electrons are emitted at the shifted hollow cathode exit, the electrostatic forces attract these lighter electrons towards the heavier xenon ion beam. This is evident from the finite electron charge density near and below the ion source region, at $y \le 0.88$ m in Fig. 23(a). Due to this initial force exerted by the induced electric field, the electrons accelerate with a high negative y-velocity towards the ion beam, which is shown in Fig. 24, and overshoot its radial edge resulting in a positively charged plume center. At $t\omega_{peo} = 1000$, the off-shoot of electrons observed in Fig. 23(b) near the bottom edge of the beam at $z = 0.18$ m suggests that the electrons overshoot the ion beam edge, shown in Fig. 24(b). These electrons that overshoot are pulled back towards the plume reversing their direction, as observed from the envelop of electron charge density surrounding the ion charge density at $t\omega_{peo} = 1500$, shown in Figs. 23(c) and 24(c), respectively. As the plume evolves further, the ion beam traps the electrons within the beam, which in turn damps the cross-stream oscillations, so that the confined electrons propagate with the ion beam, as shown in Figs. 23(d) and 24(d) for $t\omega_{peo} = 3000$. In contrast, no such cross-stream electron oscillations were observed for case 2, which had a co-located electron and ion source.



(a) y-velocity component at z=0.1, (blue), and 0.54 m, (pink). For the analytical distribution, T_e=1.5 and 1.1 eV for z=0.1 and 0.54 m, respectively.



(b) z-velocity component at z=0.005, and analytical distribution at T_e=2 eV.

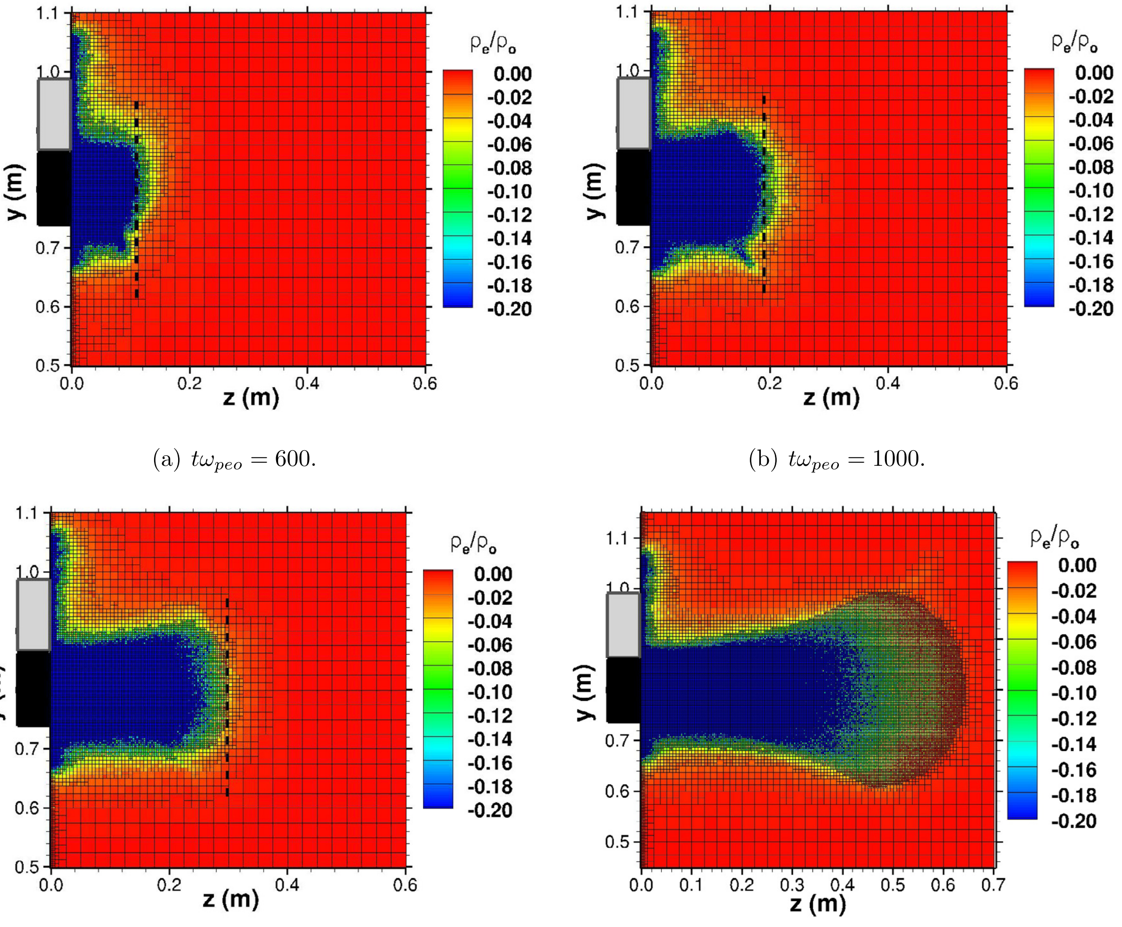


(c) z-velocity component at z=0.1, (blue), and 0.54 m (pink). For the analytical distribution, T_e=2.0 and 1.5 eV for z=0.1 and 0.54 m, respectively.

Fig. 22. Comparison of case 2 electron velocity distribution sampled from the simulation (the solid lines), at plasma time $t\omega_{peo} = 3000$, with the analytical MB distribution (dotted symbols), at locations $z = 0.005$, 0.1, and 0.54 m.

We observe striations in the plume beyond $z = 0.36$ m in the electron and ion charge density distributions shown in Figs. 23(d) and 24(d), respectively. Although such striations were observed in the modeling of plasma streamers [9–11], in this work, we attribute their formation to numerical noise since these contour plots show instantaneous results from the transient plume calculations. Additionally, the octree cell size in these regions is over-refined compared to the local Debye length with fewer than 20 particles, which may further contribute to noise, especially, for instantaneous results. The exact nature of these numerical striations as the plume reaches steady-state will be investigated in future work.

In comparison to case 2, the ion beam for case 3 has a wider radial expansion and longer streamwise extent. The maximum radial width of the case 3 ion beam at $t\omega_{peo} = 1500$ and 3000 is 0.2 and 0.4 m, as observed from Figs. 24(c) and 24(d), respectively. In contrast, the maximum radial width of the case 2 ion beam was 0.16 m at $t\omega_{peo} = 3000$, as shown previously in Fig. 18(d). This is because, in case 3, the same electrostatic forces that attract the oscillating electrons towards the plume-axis also cause the ions to be repelled away from the plume axis, consequently results in a wider ion beam compared to that observed for case 2. The dotted lines shown in Figs. 24(a) to 24(d), indicate that the ion beam-front is located at $z = 0.1$, 0.19, 0.3 and 0.64 m at plasma times $t\omega_{peo} = 600$, 1000, 1500, and 3000, respectively. The beam-front for the shifted electron case at $t\omega_{peo} = 3000$, shown in Fig. 24(d) lies 0.1 m further downstream from the beam-front location observed at $z = 0.54$ m for case 2, shown in Fig. 18(d). This is due to the acceleration induced by the electrons that overshoot the beam-front in case 3, evidence of which can be seen from the finite number density of electrons downstream



(a) $t\omega_{peo} = 600$.

(b) $t\omega_{peo} = 1000$.

(c) $t\omega_{peo} = 1500$.

(d) $t\omega_{peo} = 3000$.

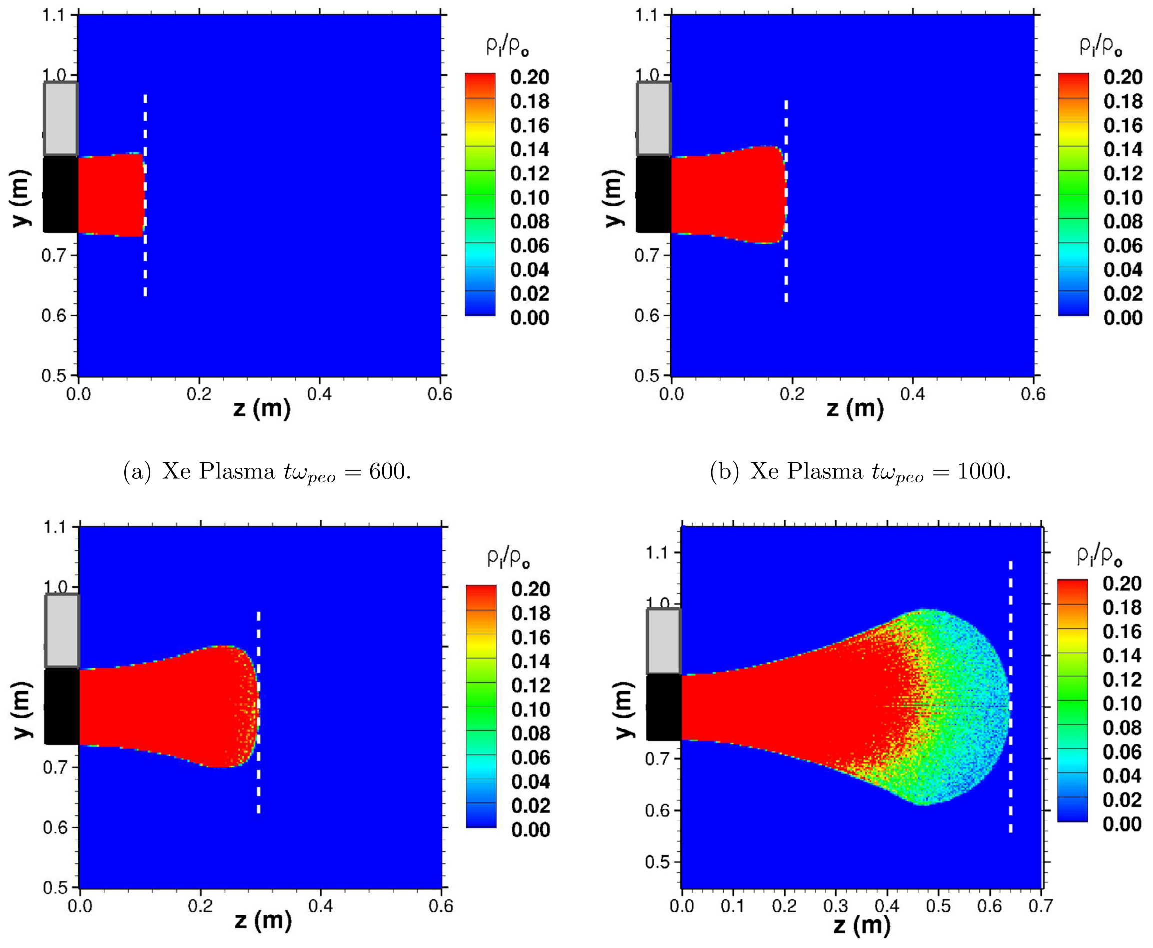
Fig. 23. Transient electron charge density for case 3, along the y–z plane extracted at the center of the domain, normalized by ρ_o.

of the dotted-line, observed at all time instances as shown in Fig. 23. Additionally, the shift in the electron source breaks the axial symmetry in the ion charge and electron charge density variation within the plume, as shown in Fig. 24(d).

The instantaneous variation in electric potential for case 3 at plasma times $t\omega_{pe} = 600$, 1000, 1500, and 3000, are shown in Fig. 25. At an initial plasma time of $t\omega_{pe} = 600$, the maximum potential within the beam near the thruster exit is 50 V, as shown in Fig. 25(a). This confirms that, even though the electrons are attracted towards the beam, the electron charge density is higher at the radial edges compared to the beam-center, resulting in a positive potential at the plume-center. At times, $t\omega_{peo} = 1000$ and 1500, the maximum potential decreases by 10 V indicating that more electrons are trapped within the beam compared to earlier plasma times. Downstream from the thruster exit, the potential decreases along the plume axis towards the beam-front. As the plume expands further and the corresponding ion charge density decreases below $0.2\rho_o$, the beam-front potential at $t\omega_{peo} = 3000$, shown in Fig. 25(d), is 80 V lower than the thruster exit potential of 40 V. Note that due to the Dirichlet boundary condition, the potential surrounding the electron and ion sources is zero, as observed in Figs. 25(a)–25(d).

The instantaneous streamwise ion velocities for case 3 at plasma times, $t\omega_{peo} = 600$, 1000, 1500, and 3000 are shown in Fig. 26. The heavy xenon ions are observed to decelerate to $0.9v_{ibeam}$ at the top y-edge of the beam, due to the electrostatic force exerted by the electrons that are emitted from the shifted source. As previously discussed, due to the electron oscillations in the y-direction, the electrons overshoot and envelop around the radial edge of the beam, consequently decelerating the ion streamwise velocity even at the lower y-edge of the beam as shown in Figs. 26(c) and 26(d) at times $t\omega_{peo} = 1500$ and 3000, respectively. Similarly, the electrons that overshoot the beam-front, as discussed previously in Fig. 23, accelerate the ions in the streamwise direction to $1.2v_{ibeam}$ at the beam-front as shown in Figs. 26(b) to 26(d). Due to the deceleration in the streamwise ion velocity at the radial edges and the acceleration at the beam-front, the leading edge of the plume is curved for case 3, as shown in Fig. 26(d), compared to the flat-shaped beam-front shown in Fig. 19(d) for case 2.

The charge densities extracted along the plume axis, $(0.8, 0.8, z)$, for case 3 at $t\omega_{pe} = 3000$ are shown in Fig. 27(a). The ion charge density gradually decreases from ρ_o at the thruster exit to zero at the beam-front, contrary to the constant



(a) Xe Plasma $t\omega_{peo} = 600$.

(b) Xe Plasma $t\omega_{peo} = 1000$.

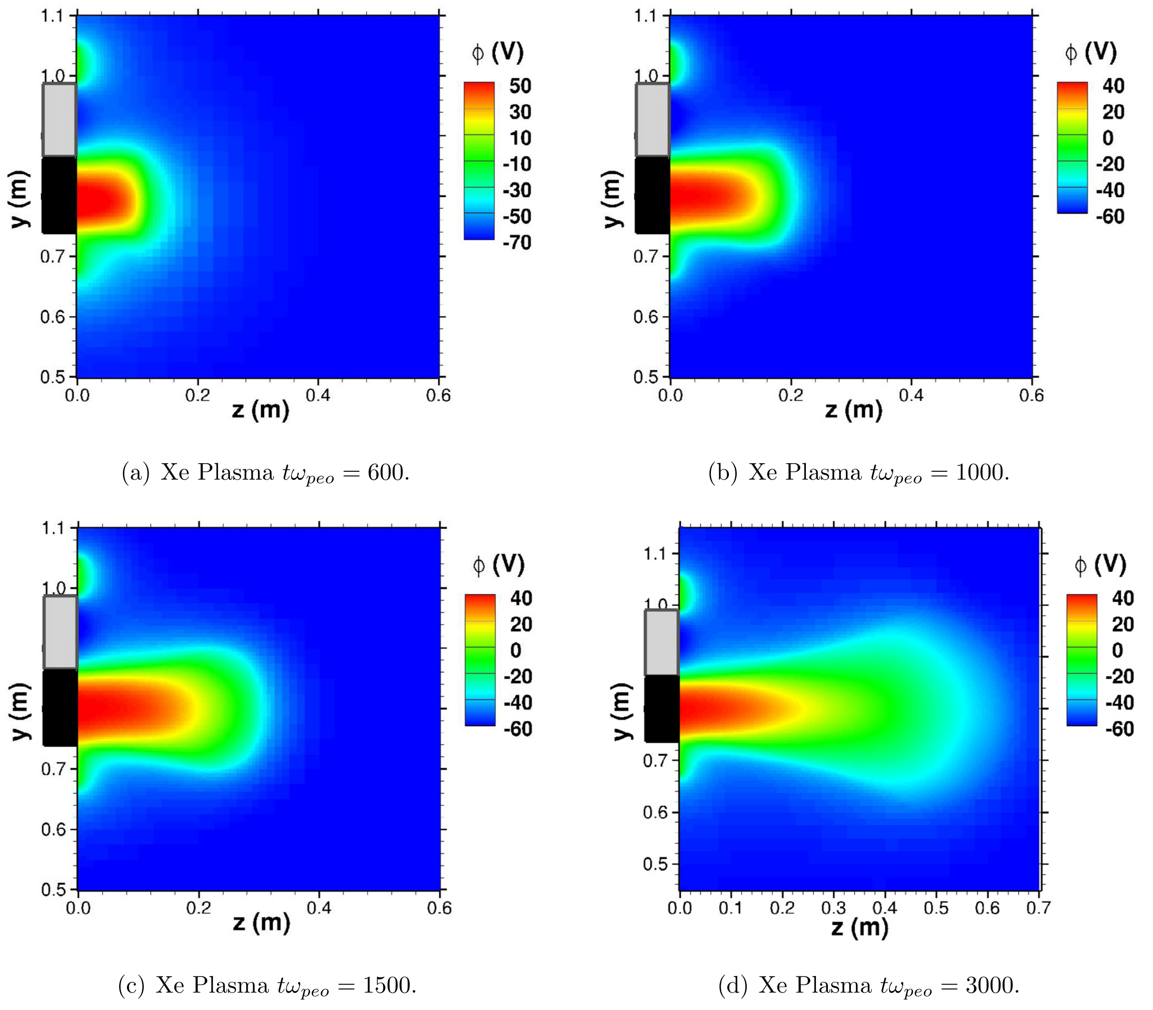
(c) Xe Plasma $t\omega_{peo} = 1500$.

(d) Xe Plasma $t\omega_{peo} = 3000$.

Fig. 24. Transient ion charge density for case 3, along the y–z plane extracted at the center of the domain, normalized by ρ_o.

ion charge density variation followed by a rapid decrease at the beam-front observed for case 2, shown in Fig. 21(a). In addition, the total charge distribution is positive prior to $z = 0.36$ m, beyond which the charge fluctuates about neutrality for case 3. Note that unlike case 2, the linear charge density profiles of ρ_i and ρ_e are not symmetric, as shown in Fig. 21(a). The instantaneous potential variation along the plume axis is shown in Fig. 27(b), where the potential decreases from 40 V at the thruster exit to −60 V towards the boundary. The potential is zero only at $z = 0.36$ as shown by the dotted line in Fig. 27(b), consistent with the total charge density variation dropping to zero at the same location, as shown in Fig. 27(a). Because the electron charge density in case 3 was higher at the beam-edges compared to the center, the potential variation along the plume axis is more gradual for case 3.

The previous discussion has focused primarily on the macroscopic properties of the evolving plume for case 3, however, examination of electron behavior at very early times provides insight into the physics of plume neutralization. These electron and ion kinetic properties can be understood from phase-space plots and EVDFs only from fully PIC simulations. The evolution of the charged particles in phase-space at early times provides a clear visualization of electron oscillations in the cross-stream y-direction, along which the electron source is shifted. Fig. 28 shows the electron and ion distribution in phase-space, where the cross-stream velocity, v_e and v_i, normalized by v_{teo}, is plotted on the y-axis and the position of the charged particles in the streamwise direction is plotted on the z-axis, at $t\omega_{peo} = 25$, 50, 62.3, and 75. Since the ion velocity is much smaller than the electron velocity for a mesothermal plume, a zoomed-in view of the ion phase-space is shown in all the figures. Initially, at $t\omega_{peo} = 25$, the electrons exhibit a high negative velocity, indicating that the electrostatic forces attract the electrons towards the ion beam, as shown in Fig. 28(a), whereas, in contrast, the cross-stream velocities of the heavy xenon ions are symmetric about zero, with a range of $v_i/v_{teo} = \pm 0.001$. Due to their high negative y-velocity, the attracted electrons overshoot the radial edge of the ion beam, leaving the beam with a net positive charge. As discussed previously, this positively charged beam subsequently attracts the electrons back towards the beam resulting in a second electron population with a high positive velocity, at $t\omega_{peo} = 50$, as shown in Fig. 28(b). The two counter-streaming electron



(a) Xe Plasma $t\omega_{peo} = 600$.

(b) Xe Plasma $t\omega_{peo} = 1000$.

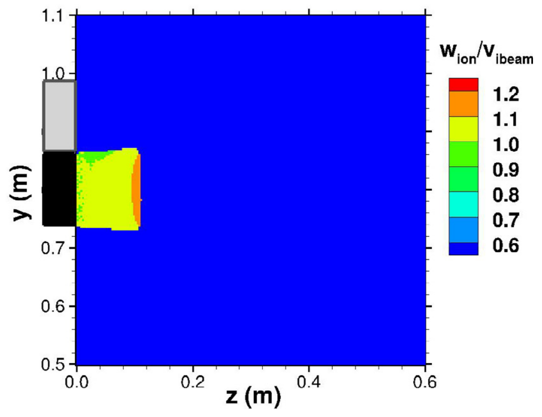
(c) Xe Plasma $t\omega_{peo} = 1500$.

(d) Xe Plasma $t\omega_{peo} = 3000$.

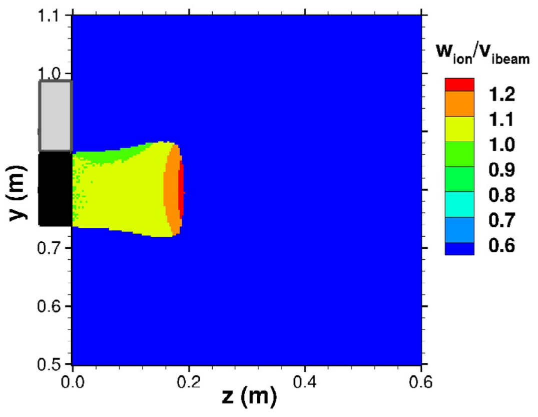
Fig. 25. Transient electric potential variation for case 3, along the y–z plane extracted at the center of the domain.

populations with high positive and negative y-velocity components and the hole in the phase-space are a precursor to a two-stream instability [73]. However the hole in the phase-space does not grow in time, as observed from the electron phase-space plots at $t\omega_{peo} = 62.3$ and 75, shown in Figs. 28(c) and 28(d), respectively. This is because the ion beam traps the electrons within the plume, resulting in the mixing of the two counter-streaming populations, such that, the electron oscillations are damped and a finite number of electrons with zero cross-stream velocity exist within the beam. At these early times, the electrons that overshoot the ion beam-front due to the high streamwise thermal velocity, still exhibit a gap in the phase-space. This further confirms that the electron trapping mechanism within the heavy xenon ion beam is the main cause for the eventual mixing of the counter-streaming electrons, and the resulting decay of any instability characteristics.

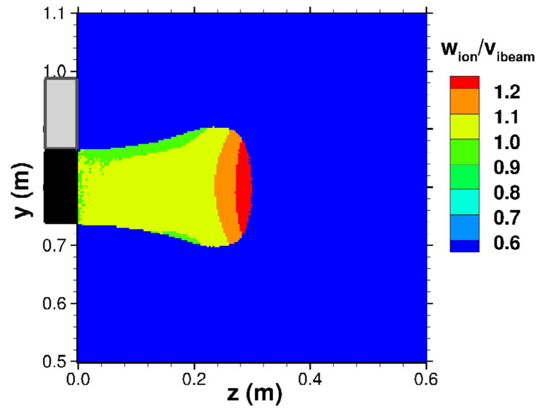
To study the evolution of the electron oscillations in the cross-stream direction and their eventual damping, we analyze the y-EVDF at $t\omega_{peo} = 100$, 200, 400, 1000, 1500, and 3000, as shown in Fig. 29(a). Note that, the electrons are sampled within a radius of 0.05 m around the plume axis at $z = 0.005$ m. At $t\omega_{peo} = 100$ and 200, the y-velocity distribution peak is at $v_e/v_{teo} = -10$ with a secondary peak at $v_e/v_{teo} = 8$ and 10, respectively. This shows that at early times, the electron velocity distribution is bi-modal, consistent with the phase-space characteristics observed for the electrons in Fig. 28. At $t\omega_{peo} = 400$, the spacing between the two peak locations decreases, and the normalized probability of electrons with zero cross-stream velocity increases to 0.4, supporting the hypothesis that the ion beam is trapping the electrons, which in turn, damps the electron oscillations. As the plume evolves to $t\omega_{peo} = 1000$, 1500, and 3000, the peak of the y-EVDF lies at $v_e/v_{teo} = 0$, with a smaller peak at $v_e/v_{teo} = -8$ showing that most electrons do not oscillate, and the electrons emitted from the shifted source are immediately attracted towards the plume. Downstream from the thruster-exit, the local y-EVDF sampled at $z = 0.1$ m within a radius of 0.05 m from the beam axis for plasma times $t\omega_{peo} = 400$, 1000, 1500, and 3000 is shown in Fig. 29(b). Compared to the plateau-like distribution observed at $t\omega_{peo} = 1000$, where the probability is almost uniform for $-8 < v_e/v_{teo} < 8$, the distribution at $t\omega_{peo} = 1500$ and 3000 shows a peak at $v_e/v_{teo} = 0$. This suggests that



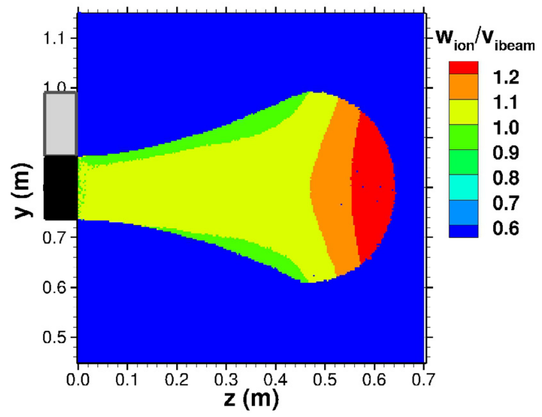
(a) Xe Plasma $t\omega_{peo} = 600$.



(b) Xe Plasma $t\omega_{peo} = 1000$.

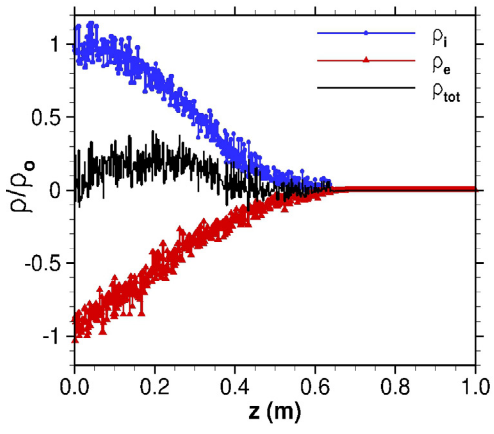


(c) Xe Plasma $t\omega_{peo} = 1500$.

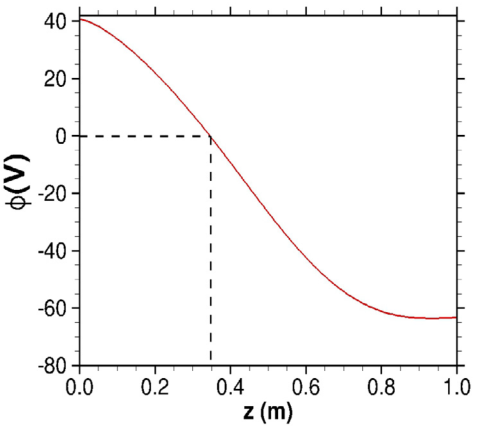


(d) Xe Plasma $t\omega_{peo} = 3000$.

Fig. 26. Transient streamwise ion velocity for case 3, along the y–z plane extracted at the center of the domain, normalized by v_{ibeam}.

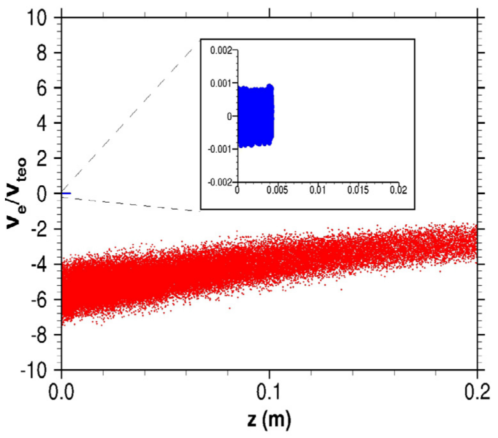


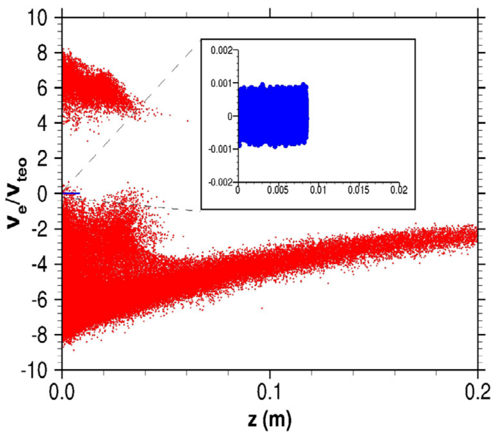
(a) Variation of ion, electron, and total charge distribution normalized by ρ_o=en_{eo}.

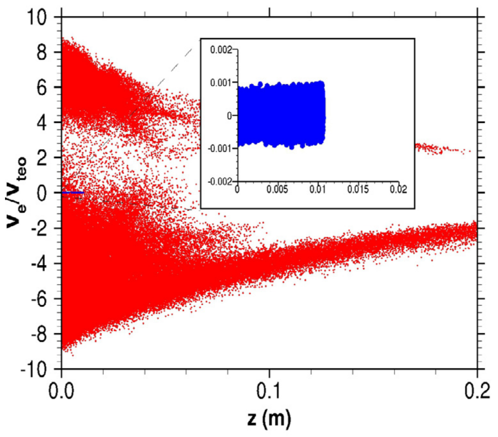


(b) Variation of electric potential along the centerline

Fig. 27. Variation of charge distribution and electric potential along the plume center-line at plasma time $t\omega_{peo} = 3000$ for case 3.


(a) $t\omega_{peo}$=25, or t=1,400 ns


(b) $t\omega_{peo}$=50, or t=2,800 ns


(c) $t\omega_{peo}$=62.3, or t=3,500 ns

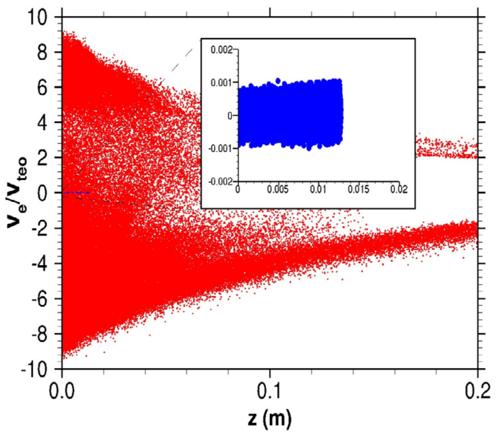
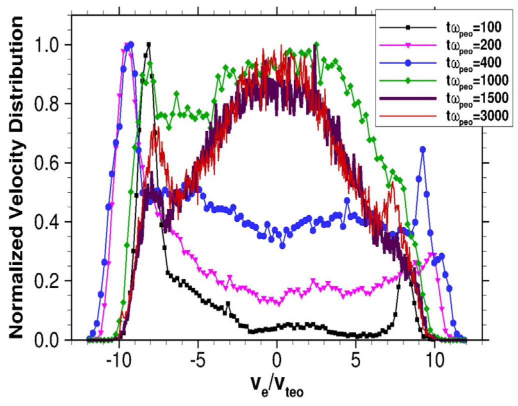
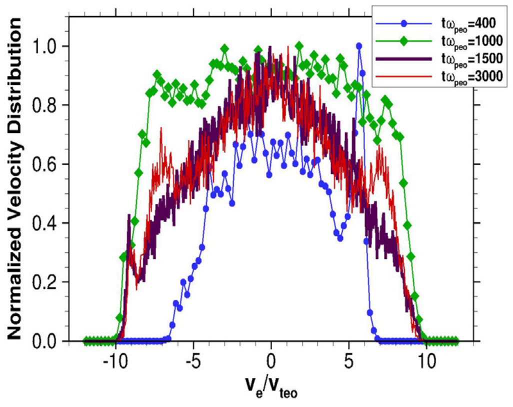

(d) $t\omega_{peo}$=75, or t=4,200 ns

Fig. 28. Phase-space plot of the electrons (red) and ions (blue) showing the y-velocity component normalized by v_{teo} along the y-axis, and the position in the streamwise direction along the z-axis.



(a) y-velocity, v_e/v_{teo} at z=0.005 m



(b) y-velocity, v_e/v_{teo} at z=0.1 m

Fig. 29. Evolution of the cross-stream velocity-component, v_e, at $z = 0.005$ and 0.1 for case 3.

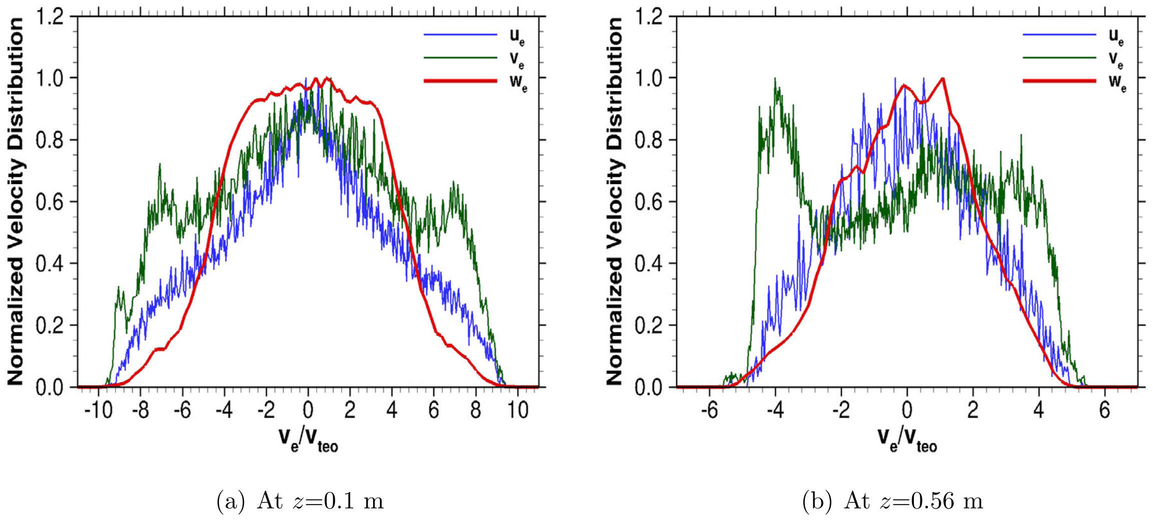


Fig. 30. Normalized u_e (blue), v_e (green), and w_e (red) distribution functions at $z = 0.1$ and 0.56 m, at plasma time $t\omega_{peo} = 3000$.

as the trapped electron mix within the xenon ion beam, the bi-modal EVDF transitions to a single peak non-Maxwellian distribution, thereby damping the cross-stream electron oscillations.

The normalized u_e, v_e, and w_e EVDFs obtained at $z = 0.1$ and 0.56 m at $t\omega_{peo} = 3000$ are shown in Figs. 30(a) and 30(b), respectively. The EVDFs in all three directions are found to be non-Maxwellian, contrary to the Maxwellian distributions observed for case 2 with a co-located electron and ion source. The u_e distribution has a single peak at both locations in comparison to the v_e distribution which has secondary peaks at $\pm 7v_{teo}$ at $z = 0.1$ m (Fig. 30(a)) and at $-4v_{teo}$ (Fig. 30(b)) at $z = 0.56$ m. This is because the electron source is shifted in the y-direction causing the secondary peak in the EVDF primarily in that direction.

6. Conclusions

A coupled PIC-DSMC framework for plasma simulations using heterogeneous CPU–GPU architecture is described in this work. The PIC and DSMC modules in the CHAOS solver construct separate linearized Z-ordered forest of trees that respectively satisfy the Debye length and local mean free path criteria and they are partitioned using different computational weights. The parallelization strategies to model the Particle-In-Cell approach by solving Poisson's equation on the linearized octree is found to scale efficiently with increase in the number of GPUs. Strong scaling studies showed that equal partitioning of leaf nodes among the GPUs does not necessarily lead to load balancing, especially for simulations that use fewer number of GPUs. However, with an increase in the number of GPUs, the scaling is near ideal.

Three mesothermal collisionless plume simulations were performed. In the first case, the electron and ion source was co-located and the ion mass was equal to that of a proton. The transient plume dynamics and electric potential showed that the plume achieved a quasi-neutral state by trapping electrons within the plume. The electron temperature was anisotropic, i.e., the cross-stream temperature components were not equal to the streamwise temperature component. In the second case, instead of a proton mass, xenon ions were introduced from the ion source to model the propellant used in real ion thrusters. The higher ion mass resulted in a more confined plume compared to the proton plasma which showed a larger radial spread. Since the beam-like xenon plume trapped the electrons within a more confined region, the electron temperatures for the xenon ion case were found to be higher than those obtained in case 1. For both the cases with co-located ion and electron sources, axial symmetry in the electron and ion properties was observed. Finally, in the third case, the electron source was shifted from the xenon ion source, to study the effect of electron source location on the plume dynamics. A bi-modal electron velocity distribution was observed to occur at early times indicating that the electrons that are emitted from the shifted source oscillate in the cross-stream direction due the electrostatic forces exerted by the beam. With the progression of time, the confined xenon beam traps these electrons and the cross-stream electron velocity distribution transitions to one centered at zero. The shift in the electron source also breaks the symmetry observed for the co-located cases, such that, the velocity distributions in all three directions are unequal. Furthermore, the electron velocity distributions in all three directions were non-Maxwellian, unlike the Maxwellian distributions observed for the co-located cases.

The electron kinetics can only be studied using a particle approach, such as PIC, and to improve computational run-time for such methods, hybrid CPU–GPU parallelization was exploited. Although an electrostatic plasma is modeled in this paper, the 2:1 octree framework can be extended to include the Maxwell's equations for obtaining the magnetic field. Finally, the

Poisson solver for the 2:1 E-FOT can also be used for applications that require an electron-fluid approach, by reducing the mass, momentum and energy conservation equations to a Poisson-like form [21].

Acknowledgements

We are grateful for the funding support provided by AFOSR through the Grant AF FA9550-16-1-0193. This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. We also acknowledge Xsede for the start-up allocation on XStream. We gratefully acknowledge Prof. Vincent Le Chenadec for the valuable discussions on the Poisson Solver implementation. We thank Nakul Nuwal for going through some sections of this manuscript and giving his feedback, which helped to improve the draft.

Appendix A. Code snippet to compute locational key of face neighbor

```
//*******************************************************************//
//** Consider a leaf node with id : leafId                         **//
//*******************************************************************//
//* Variable Definition :                                          **//
//* leafId : Id of the leaf under consideration                    **//
//* LeafLevel : Array of leaf level for all the leaf nodes         **//
//* xyz_bit : Binary of x-min,y-min, & z-min position of leaf node **//
//* mod_xyzbit : Binary of x-min,y-min,z-min of face neighbor      **//
//* dx_bit : Binary form of leaf node size , dx                    **//
//* FaceNbrLocKey : Location key of face neighbor                  **//
//* FaceNbrLeafId : LeafId of face neighbor                        **//
//-------------------------------------------------------------------||
uint64_t Morton3(uint64_t x, uint64_t y, uint64_t z)
{
 return ((Partby2(x)<<2) + (Partby2(y)<<1) + (Partby2(z))) ;
 //Partby2 code snippet is given in Ref~cite{RJDAL_CnF}
}
uint64_t Reverse_PartBy2(uint64_t n)
{
     n &= 0x1249249249249249 ;
     n = (n ^ (n >> 2 )) & 0x10c30c30c30c30c3;
     n = (n ^ (n >> 4 )) & 0x100f00f00f00f00f;
     n = (n ^ (n >> 8 )) & 0x001f0000ff0000ff;
     n = (n ^ (n >> 16)) & 0x001f00000000ffff;
     n = (n ^ (n >> 32)) & 0x00000000001fffff;
     return n;
}
ComputeFaceNbr(int leafId,int MaxLevel,int *LeafLevel)
{
   // Determine Morton Id of leaf node with id : leafId //
   xyz_bit = new uint64_t[3];
   xyz_bit[0] = Reverse_PartBy2(uint64_t(MortonId[leafId])>>2);
   xyz_bit[1] = Reverse_PartBy2(uint64_t(MortonId[leafId])>>1);
   xyz_bit[2] = Reverse_PartBy2(uint64_t(MortonId[leafId]));
   // mod_xyzbit is the Morton Id of the face Nbr //
   uint64_t *mod_xyzbit = new uint64_t[3];
   mod_xyzbit[0] = xyz_bit[0];
   mod_xyzbit[1] = xyz_bit[1];
   mod_xyzbit[2] = xyz_bit[2];
   uint64_t dx_bit = uint64_t(1)<<uint64_t(MaxLevel-LeafLevel[leafId]);
   currentLeafLevel = LeafLevel[leafId];
   if(i_face<3)
   {
      i_dim = i_face;
      Nbrs_face = i_face + 3;
      mod_xyzbit[i_dim] -=1 ;
   }
   else
   {
      i_dim = i_face - 3;
      Nbrs_face = i_face - 3;
      mod_xyzbit[i_dim] = mod_xyzbit[i_dim] + dx_bit;
   }
   FaceNbrLocKey = Morton3(mod_xyzbit[0],mod_xyzbit[1],mod_xyzbit[2]);
   FaceNbrLeafId = LeafLocationArray[FaceNbrLocKey];
}
```

Appendix B. Code snippet to show dot product computation using MPI-CUDA

```


//*****************************************************************//
//** Consider vector d and q                                    **//
//** We have to find the dot product  gamma = d . q             **//
//** Only a chunk of the vectors are stored on the GPU          **//
//** So, compute dot product in two steps :                     **//
//** Local step : compute gamma_loc = dev_dvec . dev_qvel       **//
//** Global step : Compute gamma = sum of gamma_loc of all procs **//
//*****************************************************************//
//* Variable Definition :                                       **//
//* NumLeafNodes : number of leaf nodes on the GPU sub-domain   **//
//---------------------------------------------------------------//

// Include the following in the CUDA file //
#include<cublas_v2.h>




//Initialize

cublasHandle_t cublasHandle = 0;
cublasStatus_t cublasStatus;
cublasStatus = cublasCreate(&cublasHandle);
double gamma_loc = 0.0;
double gamma = 0.0;
// Call cublas to compute dot product of vector of doubles (Ddot)
cublasDdot(cublasHandle,NumLeafNodes,dev_dvec,1,dev_qvec,1,&gamma_loc);
cudaDeviceSynchronize();

//Call Communication Subroutine to execute the following mpi_reduce command
CommunicationDomain->AllReduceForDotProduct(gamma_loc,gamma);


// The following subroutine is defined in the CommunicationDomain class
CommunicationDomain::AllReduceForDotProduct(&loc_sum,&glo_sum)
{
// Use MPI_Reduce to compute sum of all the partial sums
   MPI_Allreduce(&loc_sum,&glo_sum,1,MPI_DOUBLE,MPI_SUM,MPI_COMM_WORLD);
}
```

References

[1] D.M. Goebel, I. Katz, Fundamentals of Electric Propulsion: Ion and Hall Thrusters, vol. 1, John Wiley & Sons, 2008.
[2] B. Korkut, Z. Li, D. Levin, et al., 3-D simulation of ion thruster plumes using octree adaptive mesh refinement, IEEE Trans. Plasma Sci. 43 (5) (2015) 1706–1721.
[3] M. Carruth Jr., A review of studies on ion thruster beam and charge-exchange plasmas, in: 16th International Electric Propulsion Conference, 1982, p. 1944.
[4] K.G. Xu, M.L. Walker, Effect of external cathode azimuthal position on Hall-effect thruster plume and diagnostics, J. Propuls. Power 30 (2) (2014) 506–513.
[5] I.D. Boyd, M.W. Crofton, Modeling the plasma plume of a hollow cathode, J. Appl. Phys. 95 (7) (2004) 3285–3296.
[6] K.-I. Nishikawa, P. Hardee, G. Richardson, R. Preece, H. Sol, G. Fishman, Particle acceleration in relativistic jets due to Weibel instability, Astrophys. J. 595 (1) (2003) 555.
[7] E. Alves, T. Grismayer, R. Fonseca, L. Silva, Electron-scale shear instabilities: magnetic field generation and particle acceleration in astrophysical jets, New J. Phys. 16 (3) (2014) 035007.
[8] J. Deca, A. Divin, G. Lapenta, B. Lembège, S. Markidis, M. Horányi, Electromagnetic particle-in-cell simulations of the solar wind interaction with lunar magnetic anomalies, Phys. Rev. Lett. 112 (Apr. 2014) 151102.
[9] D. Levko, M. Pachuilo, L.L. Raja, Particle-in-cell modeling of streamer branching in CO_2 gas, J. Phys. D, Appl. Phys. 50 (35) (2017) 354004.
[10] C. Montijn, W. Hundsdorfer, U. Ebert, An adaptive grid refinement strategy for the simulation of negative streamers, J. Comput. Phys. 219 (2) (2006) 801–835.
[11] V. Kolobov, R. Arslanbekov, Towards adaptive kinetic-fluid simulations of weakly ionized plasmas, J. Comput. Phys. 231 (3) (2012) 839–869.
[12] P.K. Chu, J. Chen, L. Wang, N. Huang, Plasma–surface modification of biomaterials, Mater. Sci. Eng., R Rep. 36 (5) (2002) 143–206.
[13] T. Desmet, R. Morent, N.D. Geyter, C. Leys, E. Schacht, P. Dubruel, Nonthermal plasma technology as a versatile strategy for polymeric biomaterials surface modification: a review, Biomacromolecules 10 (9) (2009) 2351–2378.
[14] C.K. Birdsall, Particle-in-cell charged-particle simulations, plus Monte Carlo collisions with neutral atoms, PIC-MCC, IEEE Trans. Plasma Sci. 19 (2) (1991) 65–85.
[15] 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 Phys. Control. Fusion 57 (11) (2015) 113001.
[16] A.A. Philippov, A. Spitkovsky, B. Cerutti, Ab initio pulsar magnetosphere: three-dimensional particle-in-cell simulations of oblique pulsars, Astrophys. J. Lett. 801 (1) (2015) L19.

[17] S.J. Choi, M.J. Kushner, A particle-in-cell simulation of dust charging and shielding in low pressure glow discharges, IEEE Trans. Plasma Sci. 22 (2) (Apr 1994) 138–150.
[18] G. Lapenta, DEMOCRITUS: an adaptive particle in cell (PIC) code for object–plasma interactions, J. Comput. Phys. 230 (12) (2011) 4679–4695.
[19] J. Boeuf, Characteristics of a dusty nonthermal plasma from a particle-in-cell Monte Carlo simulation, Phys. Rev. A 46 (12) (1992) 7910.
[20] I.D. Boyd, R.A. Dressler, Far field modeling of the plasma plume of a Hall thruster, J. Appl. Phys. 92 (4) (2002) 1764–1774.
[21] I.D. Boyd, J.T. Yim, Modeling of the near field plume of a Hall thruster, J. Appl. Phys. 95 (9) (2004) 4575–4584.
[22] F. Taccogna, S. Longo, M. Capitelli, Particle-in-cell with Monte Carlo simulation of SPT-100 exhaust plumes, J. Spacecr. Rockets 39 (3) (2002) 409–419.
[23] R.S. Roy, D. Hastings, N. Gatsonis, Ion-thruster plume modeling for backflow contamination, J. Spacecr. Rockets 33 (4) (1996) 525–534.
[24] J. Wang, D. Brinza, M. Young, Three-dimensional particle simulations of ion propulsion plasma environment for deep space 1, J. Spacecr. Rockets 38 (3) (2001) 433–440.
[25] C. Cai, Numerical studies on plasma plume flows from a cluster of electric propulsion devices, Aerosp. Sci. Technol. 41 (2015) 134–143.
[26] G. Lapenta, Particle simulations of space weather, J. Comput. Phys. 231 (3) (2012) 795–821.
[27] R.W. Hockney, J.W. Eastwood, Computer Simulation Using Particles, CRC Press, 1988.
[28] B.I. Cohen, A.B. Langdon, D.W. Hewett, R.J. Procassini, Performance and optimization of direct implicit particle simulation, J. Comput. Phys. 81 (1) (1989) 151–168.
[29] G. Lapenta, Exactly energy conserving semi-implicit particle in cell formulation, J. Comput. Phys. 334 (2017) 349–366.
[30] G. Chen, L. Chacón, D.C. Barnes, An energy- and charge-conserving, implicit, electrostatic particle-in-cell algorithm, J. Comput. Phys. 230 (18) (2011) 7018–7036.
[31] S. Markidis, G. Lapenta, The energy conserving particle-in-cell method, J. Comput. Phys. 230 (18) (2011) 7037–7052.
[32] 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, et al., OSIRIS: a three-dimensional, fully relativistic particle in cell code for modeling plasma based accelerators, in: International Conference on Computational Science, Springer, 2002, pp. 342–351.
[33] S. Zabelok, R. Arslanbekov, V. Kolobov, Adaptive kinetic-fluid solvers for heterogeneous computing architectures, J. Comput. Phys. 303 (2015).
[34] J.-L. Vay, P. Colella, J. Kwan, P. McCorquodale, D. Serafini, A. Friedman, D. Grote, G. Westenskow, J.-C. Adam, A. Heron, et al., Application of adaptive mesh refinement to particle-in-cell simulations of plasmas and beams, Phys. Plasmas 11 (5) (2004) 2928–2934.
[35] R.S. Martin, J.-L. Cambier, Octree particle management for DSMC and PIC simulations, J. Comput. Phys. 327 (Suppl. C) (2016) 943–966.
[36] K. Fujimoto, A new electromagnetic particle-in-cell model with adaptive mesh refinement for high-performance parallel computation, J. Comput. Phys. 230 (23) (2011) 8508–8526.
[37] M.E. Innocenti, G. Lapenta, S. Markidis, A. Beck, A. Vapirev, A multi level multi domain method for particle in cell plasma simulations, J. Comput. Phys. 238 (2013) 115–140.
[38] J. Brackbill, An adaptive grid with directional control, J. Comput. Phys. 108 (1) (1993) 38–50.
[39] J. Barnes, P. Hut, A Hierarchical O (N log N) Force-Calculation Algorithm, Nature Publishing Group, 1986.
[40] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, et al., PETSc Users Manual Revision 3.5, Tech. rep., Argonne National Laboratory (ANL), 2014.
[41] R.S. Sampath, S.S. Adavani, H. Sundar, I. Lashuk, G. Biros, Dendro: parallel algorithms for multigrid and AMR methods on 2:1 balanced octrees, in: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, IEEE Press, 2008, p. 18.
[42] W. Bangerth, R. Hartmann, G. Kanschat, Deal.II—a general-purpose object-oriented finite element library, ACM Trans. Math. Softw. 33 (4) (2007) 24.
[43] A. Gholami, D. Malhotra, H. Sundar, G. Biros, FFT, FMM, or multigrid? A comparative study of state-of-the-art Poisson solvers for uniform and nonuniform grids in the unit cube, SIAM J. Sci. Comput. 38 (3) (2016) C280–C306.
[44] P. Ricker, A direct multigrid Poisson solver for oct-tree adaptive meshes, Astrophys. J. Suppl. Ser. 176 (1) (2008) 293.
[45] H. Sundar, G. Biros, C. Burstedde, J. Rudi, O. Ghattas, G. Stadler, Parallel geometric-algebraic multigrid on unstructured forests of octrees, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, IEEE Computer Society Press, 2012, p. 43.
[46] F. Losasso, F. Gibou, R. Fedkiw, Simulating water and smoke with an octree data structure, ACM Trans. Graph. 23 (2004) 457–462.
[47] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries, J. Comput. Phys. 190 (2) (2003) 572–600.
[48] A. Fierro, J. Dickens, A. Neuber, Graphics processing unit accelerated three-dimensional model for the simulation of pulsed low-temperature plasmas, Phys. Plasmas 21 (12) (2014) 123504.
[49] H. Burau, R. Widera, W. Honig, G. Juckeland, A. Debus, T. Kluge, U. Schramm, T.E. Cowan, R. Sauerbrey, M. Bussmann, PIConGPU: a fully relativistic particle-in-cell code for a GPU cluster, IEEE Trans. Plasma Sci. 38 (10) (2010) 2831–2839.
[50] X. Kong, M.C. Huang, C. Ren, V.K. Decyk, Particle-in-cell simulations with charge-conserving current deposition on graphic processing units, J. Comput. Phys. 230 (4) (2011) 1676–1685.
[51] V.K. Decyk, T.V. Singh, Particle-in-cell algorithms for emerging computer architectures, Comput. Phys. Commun. 185 (3) (2014) 708–719.
[52] M. Bussmann, H. Burau, T.E. Cowan, A. Debus, A. Huebl, G. Juckeland, T. Kluge, W.E. Nagel, R. Pausch, F. Schmitt, et al., Radiative signatures of the relativistic Kelvin–Helmholtz instability, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, ACM, 2013, p. 5.
[53] S. Bastrakov, R. Donchenko, A. Gonoskov, E. Efimenko, A. Malyshev, I. Meyerov, I. Surmin, Particle-in-cell plasma simulation on heterogeneous cluster systems, J. Comput. Sci. 3 (6) (2012) 474–479.
[54] G. Bird, Molecular Gas Dynamics and the Direct Simulation of Gas Flows, Oxford Engineering Science Series, Clarendon Press, 1994.
[55] R. Jambunathan, D.A. Levin, Advanced parallelization strategies using hybrid MPI-CUDA octree DSMC method for modeling flow through porous media, Comput. Fluids 149 (2017) 70–87.
[56] G. Stantchev, W. Dorland, N. Gumerov, Fast parallel particle-to-grid interpolation for plasma PIC simulations on the GPU, J. Parallel Distrib. Comput. 68 (10) (2008) 1339–1349.
[57] C. Capon, M. Brown, C. White, T. Scanlon, R. Boyce, pdFOAM: a PIC-DSMC code for near-Earth plasma–body interactions, Comput. Fluids 149 (2017) 160–171.
[58] J. Wang, O. Chang, Y. Cao, Electron–ion coupling in mesothermal plasma beam emission: full particle PIC simulations, IEEE Trans. Plasma Sci. 40 (2) (2012) 230–236.
[59] Y. Hu, J. Wang, Electron properties in collisionless mesothermal plasma expansion: fully kinetic simulations, IEEE Trans. Plasma Sci. 43 (9) (2015) 2832–2838.
[60] Y. Hu, J. Wang, Fully kinetic simulations of collisionless, mesothermal plasma emission: macroscopic plume structure and microscopic electron characteristics, Phys. Plasmas 24 (3) (2017) 033510.
[61] H. Usui, A. Hashimoto, Y. Miyake, Electron behavior in ion beam neutralization in electric propulsion: full particle-in-cell simulation, J. Phys. Conf. Ser. 454 (1) (2013) 012017.
[62] H. Sundar, R.S. Sampath, G. Biros, Bottom-up construction and 2:1 balance refinement of linear octrees in parallel, SIAM J. Sci. Comput. 30 (5) (2008) 2675–2708.
[63] A. Jones, P. Jimack, An adaptive multigrid tool for elliptic and parabolic systems, Int. J. Numer. Methods Fluids 47 (10–11) (2005) 1123–1128.
[64] T. Tu, D.R. O'Hallaron, O. Ghattas, Scalable parallel octree meshing for terascale applications, in: Supercomputing 2005, Proceedings of the ACM/IEEE SC 2005 Conference, IEEE, 2005, p. 4.

[65] C. Burstedde, L.C. Wilcox, O. Ghattas, Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM J. Sci. Comput. 33 (3) (2011) 1103–1133.

[66] J. Ferziger, M. Peric, Computational Methods for Fluid Dynamics, Springer, Berlin, Heidelberg, 2012.

[67] W.L. Briggs, V.E. Henson, S.F. McCormick, A Multigrid Tutorial, SIAM, 2000.

[68] J.R. Shewchuk, et al., An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, Carnegie Melon University, Pittsburgh, PA, USA, 1994.

[69] J. Blandón, J. Grisales, H. Riascos, Electrostatic plasma simulation by particle-in-cell method using ANACONDA package, J. Phys. Conf. Ser. 850 (2017) 012007.

[70] J. Duras, K. Matyash, D. Tskhakaya, O. Kalentev, R. Schneider, Self-force in 1D electrostatic particle-in-cell codes for non-equidistant grids, Contrib. Plasma Phys. 54 (8) (2014) 697–711.

[71] S.N. Averkin, N.A. Gatsonis, A parallel electrostatic particle-in-cell method on unstructured tetrahedral grids for large-scale bounded collisionless plasma simulations, J. Comput. Phys. 363 (2018) 178–199.

[72] R.J. LeVeque, Numerical Methods for Conservation Laws, Birkhäuser Verlag, 1992.

[73] T. Stringer, Electrostatic instabilities in current-carrying and counterstreaming plasmas, J. Nucl. Energy, Part C, Plasma Phys. Accel. Thermonucl. Res. 6 (3) (1964) 267.