RESEARCH **Open Access**

GPU-accelerated kinetic Hall thruster simulations in WarpX

Thomas A. Marks[1*] and Alex A. Gorodetsky[1]

*Correspondence:
marksta@umich.edu

[1] Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan, USA

Abstract
Two-dimensional (axial-azimuthal) simulations of a Hall thruster are performed using the open-source particle-in-cell code WarpX. The simulation conditions are chosen to match those of the axial-azimuthal benchmark first reported by Charoy et al. in 2019. A range of numerical and solver parameters is investigated in order to find those which yield the best performance. It is found that WarpX completes the benchmark case in 3.8 days on an Nvidia V100 GPU, and in as low as 1.5 days on a more recent Nvidia H100 GPU. Of the numerical parameters investigated, it is determined that the field-solve tolerance and particle resampling thresholds have the largest effect on the simulation wall time and that particle resampling may artificially widen electron velocity distribution functions, leading to unphysical heating. A semi-implicit scheme for the electrostatic field solve is tested and is found to produce results consistent to within 10% of the benchmark in less than twelve hours. The scaling properties of the electrostatic solver to multiple GPUs are also assessed on a uniform plasma test problem. The results of this work are discussed in the context of advancements in GPU hardware and the suitability of kinetic Hall thruster simulations for engineering applications.

Keywords: Hall thruster, Kinetic, Simulation, PIC, GPU

Introduction

Kinetic whole-device simulations of Hall thruster discharges have long been too expensive to use in engineering contexts. While capable of resolving the physics driving and growth and saturation of microinstabilities that lead to so-called "anomalous" electron transport [1], such simulations require grid spacings on the scale of the electron Debye length ($\sim 10^{-6}$m) and timesteps on the order of the electron plasma frequency ($\sim 10^{-12}$s). At the same time the simulation must run long enough to capture the long length- and time-scales over which the Hall thruster plasma achieves a quasi-steady state (around 10^{-1}m and 10^{-3}s, respectively) to be useful for simulations of real devices. These stringent requirements and the long run-times that result have led to kinetic simulations being considered inadequate for engineering purposes.

As a result, the community has adopted other approaches for incorporating turbulent effects into Hall thruster simulations. Most commonly, researchers model anomalous electron transport using Bohm diffusion modified with spatially-varying scaling coefficients. This approach has been successful at producing converged simulations that match experimental data [2], but the scaling coefficients are not generalizable across thrusters or operating conditions and must be tuned for each case [3]. To address this lack of generality, many authors have derived first-principles [4, 5] or data-driven [6] closure models of how the instability-induced electron transport scale with fluid plasma properties such as temperature, density, and velocity. To date, though, no such model has proved capable enough for general engineering and design applications [7].

In light of the shortcomings of these lower-fidelity approaches, and with recent advancements in computing hardware, there has been renewed interest in simulating Hall thrusters kinetically [8]. In particular, the advent of large-scale multi-GPU computing has recently produced unprecedented speed-ups in kinetic simulations of other plasma devices, such as laser plasmas [9], tokamaks [10] and plasma wakefield accelerators [11]. GPUs have also been successfully applied to low-temperature plasma applications [12], including the low-density plumes of electric propulsion systems [13]. These successes suggest that GPUs may accelerate kinetic Hall thruster simulations.

In this work, we apply WarpX, an open-source particle-in-cell (PIC) code, to the problem of Hall thruster simulation. WarpX was designed to scale to the largest supercomputing clusters [14], can run on CPUs and GPUs, and is highly extensible. As such, it serves as a good starting point for developing Hall thruster codes with similar scalability. We use WarpX to simulate the 2-D axial-azimuthal E×B benchmark of Charoy et al. [15], which models the plasma instabilities thought to drive anomalous electron transport in a simplified Hall thruster geometry. We find the following:

1. Using an explicit scheme, WarpX outperforms previously-published benchmark results on a single GPU, with completion times as low as 38 hours depending on the numerical parameters.
2. The results of the benchmark are insensitive to the precision of the Poisson solver.
3. Adopting a semi-implicit scheme for solving Poisson's equation reduces the computational cost of the benchmark simulation. We are able to complete the benchmark simulation in fewer than twelve hours, and expect that larger simulations would benefit even more strongly from the use of this scheme.
4. While WarpX's particle routines scale well as problems grow to multiple GPUs, the field solver lags behind. As demonstrated on a uniform plasma test problem, at larger problem sizes (32 GPUs), over 90% of the computational cost comes from MPI communication.

We note that paper is expanded and revised from a paper presented at the 2024 International Electric Propulsion Conference [16]. In that paper, we investigated the effect of numerical parameters beyond those investigated in the present work on the accuracy and computational cost of the benchmark simulation. These included the particle shape function and the frequency and presence of electron-ion collision checks. In that work, however, we did not investigate the scaling of the code to multiple GPUs, nor did we investigate the semi-implicit scheme mentioned above. We refer interested readers to that paper for more details.

This paper is organized as follows. In Methods, we describe the conditions of the benchmark, the capabilities of WarpX, and our modifications to WarpX to support Hall thruster simulations. We provide details of the numerical parameters and algorithms investigated in our simulations, describe our application of a semi-implicit numerical scheme, and set up an electrostatic uniform plasma scaling study. In Results, we then present the results of our studies. Finally, in the Conclusion, we conclude with some thoughts on the implications for our work on the use of particle-in-cell simulations in the engineering and design of Hall thrusters.

Methods

Benchmark simulation

In this work, we simulate Case 2a of the LANDMARK low-temperature benchmarking effort [17]. This case is designed to capture the main kinetic effects governing Hall thruster discharges—namely the onset and growth of drift-driven turbulence as well as plasma aceleration in the axial direction. In particular, we compare our efforts to results of several codes from throughout the community, reported by T. Charoy et al in 2019 [15]. In this section, we describe the benchmark conditions; the reader is referred to Ref. [15] for a more detailed description.

The benchmark is a two-dimensional axial-azimuthal particle-in cell simulation of a simplified Hall thruster geometry. We show the simulation domain in Fig. 1, where the axial (x) and azimuthal (y) dimensions have lengths of $L_x = 2.5\,\text{cm}$ and $L_y = 1.28\,\text{cm}$, respectively. Our simulations assume the azimuthal dimension is periodic and do not consider curvature effects. We apply an external radial magnetic field $B_z(x)$ throughout the domain, with the maximum field strength of 10 mT occuring at $x = 0.75\,\text{cm}$.

We employ a grid resolution of 512 cells in the axial direction ($\Delta x = 4.88 \times 10^{-5}\,\text{m}$) in the axial direction and 256 cells in the azimuthal direction ($\Delta y = 5 \times 10^{-5}\,\text{m}$), with a timestep of $5 \times 10^{-12}\,\text{s}$. These conditions are sufficient to resolve both the electron Debye length as well as the electron plasma frequency, which are critical to the stability and accuracy of the explicit particle-in-cell method.

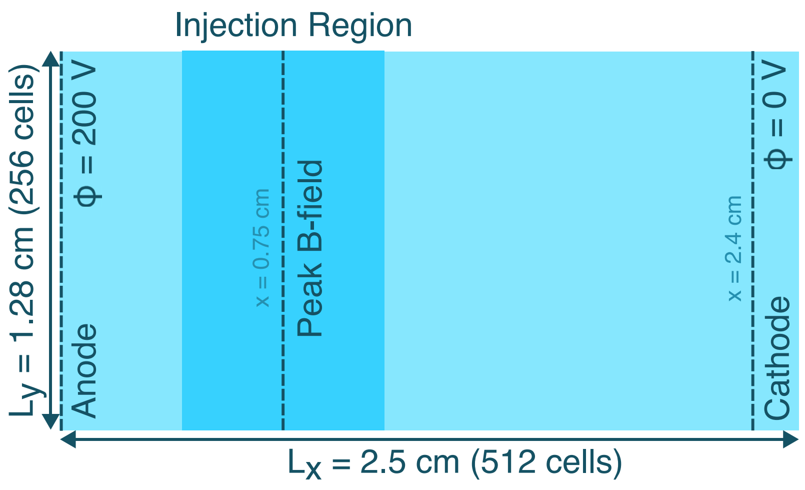


Fig. 1 The 2-D axial-azimuthal benchmark simulation domain

The benchmark does not model neutral atoms; we instead inject electron-ion pairs according to a ionization rate profile bounded by $x = 0.25\,\mathrm{cm}$ and $x = 1\,\mathrm{cm}$. This "injection region" is depicted in darker blue in Fig. 1. Injecting particles in this manner allows simulations to avoid resolving both ionization oscillations and start-up transients and achieve steady state in tens of microseconds. The temperatures of the newly-injected electrons and ions are 10 and 0.5 eV, respectively. Between the anode at $x = 0\,\mathrm{cm}$ and a "virtual cathode" at $x = 2.4\,\mathrm{cm}$, we apply a DC voltage of 200 V. To maintain current continuity, any net current that crosses the anode plane is re-injected as electron current at the cathode.

The simulation begins as a plasma with density $n_e = 5 \times 10^{16}\,\mathrm{m}^{-3}$, and electron and ion temperatures of 10 and 0.5 eV, respectively. We run the simulation for $20\,\mu\mathrm{s}$ and report plasma properties averaged over the last four microseconds of the run. The original benchmark considers three cases, differentiated by the weight of the computational macroparticles, and therefore by the number of particles in the simulation at startup. Case 1 starts with 150 particles per cell, Case 2 with 75, and Case 3 has 300. In this work, we simulate all three of these cases, but treat Case 2 as the baseline case for our subsequent studies. Table 1 contains a summary of the benchmark simulation parameters.

Table 1 Parameters of the benchmark simulations

Baseline parameters				
Axial domain length, L_x	2.5 cm			
Azimuthal domain length, L_y	1.28 cm			
Axial resolution, N_x	512			
Azimuthal resolution, N_y	256			
Time step, Δt	5×10^{-12} s			
Discharge voltage, U_0	200 V			
Maximum magnetic field, B_{max}	10^{-12} T			
Initial plasma density, n_0	$5 \times 10^{16}\,\mathrm{m}^{-3}$			
Initial electron temperature, $T_{e,0}$	10 eV			
Initial ion temperature, $T_{i,0}$	0.5 eV			
Simulation duration, t_{max}	20×10^{-6} s			
Averaging start time, t_{avg}	16×10^{-6} s			
Particle precision	Single			
Field-solve precision	Double			
Field gathering algorithm	Energy-conserving			
Additional parameters				
Initial particles per cell	75, 150, 300			
Resampling: max particles per cell	no resampling, 200, 250, 300			
Particle sort interval	10, 50, 100, 500, 1000			
Multigrid precision	10^{-2}, 10^{-3}, 10^{-5}, 10^{-6}			
Semi-implicit solver case	C	Δt	N_x	N_y
2x	8	1×10^{-11} s	256	128
4x	16	2×10^{-11} s	128	64
8x	8	4×10^{-11} s	64	32

Extending WarpX for Hall thruster simulations

WarpX is an open-source, time-dependent, relativistic, electrostatic and electromagnetic particle-in-cell code developed as part of the United States Department of Energy's Exascale Computing Project [14]. While the primary application of WarpX is simulating high-energy laser-plasma interactions [11], the generality of the code's algorithms makes it well-suited for a wide variety of plasma physics. The code is designed to scale well to very large problem sizes, on both CPU and GPU-dominated clusters [18].

As with most particle-in-cell codes, WarpX performs four main actions at each timestep when running an electrostatic simulation. These are:

1. **Gather fields to particles**: The electric and magnetic fields on the grid are interpolated to the particles using a prescribed kernel or "shape function". In this work, we use the energy-conserving field gathering scheme originally described by Lewis [19], which requires fields be stored on a staggered grid. Recent work has demonstrated that this scheme helps stabilize PIC simulations which under-resolve the Debye length and reduces the degree of numerical heating [20]. This is an alternative to the classic momentum-conserving scheme, which co-locates field quantities at nodes and can lead to aliasing and checkering artifacts in certain situations [21].
2. **Push particles**: The particles move to new positions based on their velocities and the timestep, and accelerate as a response to the fields gathered in the previous step. WarpX uses a relativistic extension of the well-known Boris scheme for particle advancement. This algorithm is second-order in time and energy-conserving, making it well-suited for capturing particle orbits around magnetic field lines. WarpX fuses this step with the field-gathering step above into a single kernel which can be efficiently executed on GPUs.
3. **Deposit charge**: The charge density ρ and current density $\mathbf{j}$ are computed on the grid from the particle positions, weights, and charges. This interpolation uses a shape function which matches that used to gather the fields onto the particles.
4. **Solve fields**: Given the charge density on the grid and appropriate boundary conditions, the code solves Poisson's equation (Eq. 1) to determine the electrostatic potential U and electric field $\mathbf{E}$:

$$-\nabla^2 U = \nabla \cdot \mathbf{E} = \rho/\epsilon_0, \tag{1}$$

   where ϵ_0 is the permittivity of free space, 8.854×10^{-12} F/m. The electric field is then gathered to the particles and the loop is repeated.

These steps on their own are not sufficient to simulate Hall thruster discharges, which include both ionization, which adds charged particles to the domain throughout the simulation and a cathode that injects sufficient electrons to neutralize the ion beam ejected by the thruster. We must therefore extend WarpX to handle these effects.

Extensions for benchmark simulations

Users running WarpX from its Python interface can specify callback functions which are triggered at specific points in the computational cycle and can access WarpX's internal data-structures. There are three parts of the benchmark not included in WarpX's core functionality and require new implementations. These are

1. The creation of particles in the injection region,
2. The injection of particles at the cathode to support current continuity, and
3. The zero-volt internal Dirichlet boundary condition at the cathode plane.

Here, we describe each of these components and their implementation into WarpX. We illustrate how each of these extensions slots into WarpX's main loop in Fig. 2.

1. Particle injection

This callback creates electron-ion pairs in the injection region according to an ionization profile at each timestep. Per the benchmark conditions [15], the ionization rate varies axially as

$$S(x) = \begin{cases} S_0 \cos\left(\pi \frac{x - x_m}{x_2 - x_1}\right) & x_1 \le x \le x_2 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In the above, $S_0 = 5.23 \times 10^{23}\,\mathrm{m}^{-3}\mathrm{s}^{-1}$ is the maximum ionization rate, $x_1 = 0.25\,\mathrm{cm}$, $x_2 = 1\,\mathrm{cm}$, and $x_m = (x_1 + x_2)/2 = 0.625\,\mathrm{cm}$. We compute the number of electron-ion pairs to be injected at each timestep by integrating this function over domain and multiplying by the timestep, giving

$$N_{inject,0} = \frac{2S_0}{\pi W_0} L_y (x_2 - x_1) \Delta t. \tag{3}$$

Here, W_0 is the base particle weight, or the number of real particles represented by a single computational macroparticle at simulation start-up, given by

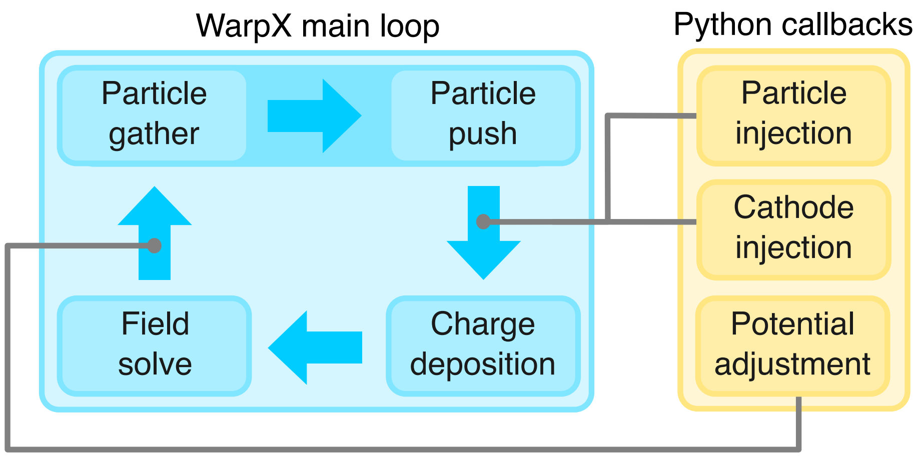


Fig. 2 WarpX's main computation loop, including the Python callbacks implemented to support the benchmark simulations

$$W_0 = \frac{n_0 L_x L_y}{N_{ppc,ini} N_x N_y}. \tag{4}$$

To implement this in WarpX, we define a callback that executes every timestep in the `particleinjection` position. This takes place after the particles have been pushed to new positions, but before the particles' charge is deposited onto the grid. We inject either $\lceil N_{inject,0} \rceil$ or $\lfloor N_{inject,0} \rfloor$ electron-ion pairs with respective probabilities P and $1 - P$, with P equal to the fractional part of N_{inject}. The axial and azimuthal positions of a newly-created electron-ion pair can be computed using inverse transform sampling and are given by [15]

$$x_i = x_m + \sin^{-1}(2r_1 - 1)\frac{x_2 - x_1}{\pi}, \tag{5}$$

$$y_i = r_2 L_y. \tag{6}$$

We sample r_1 and r_2 from uniform distributions on [0,1], the electron and ion velocities from 3-D Maxwellian distributions corresponding to their respective temperatures, and set the weights of both species to W_0. We then use WarpX's `add_particles` function to add the newly generated particles to their containers.

2. Cathode injection

To maintain current continuity, the benchmark prescribes that all net charge leaving the domain through the anode boundary returns as electron current at the cathode [15]. To implement this in WarpX, we use the code's `BoundaryScrapingDiagnostics` feature. This diagnostic allocates a buffer that logs the number and type of particles that leave the domain. At each timestep, we record the number of electrons (ΔN_e) and ions (ΔN_i) that have crossed the anode plane. We then inject $\Delta N_i - \Delta N_e$ electrons at the cathode plane ($x = x_e = 2.4\,\mathrm{cm}$). The new electrons have weight W_0, are distributed uniformly in the azimuthal dimension, and have velocities drawn from a stationary 3-D Maxwellian distribution function with temperature 10 eV. We clear the boundary buffer after each step to avoid an increasing memory footprint over time.

3. Potential adjustment

The benchmark adjusts the potential every timestep to enforce a 200 V drop between the anode and cathode lines [15]. To make this adjustment, we average the potential computed by WarpX ($U(x, y)$) along the cathode line, giving $\overline{U}_e = (\int_0^{L_y} U(x_e, y)dy)/L_y$. We then compute the corrected electrostatic potential $\phi(x, y)$:

$$\phi(x, y) = U(x, y) - \frac{x}{x_e}\overline{U}_e. \tag{7}$$

To implement this procedure in WarpX, we create a callback in the `afterEsolve` position, which ensures that the adjustment will be invoked directly after Poisson's equation has been solved. We then average the potential interpolated to the cathode line over the azimuthal direction to get $\overline{U}_e$ and apply the correction given by Eq. 7. Finally, we differentiate Eq. 7 to get a correction to the axial electric field:

$$E_x = -\frac{\partial \phi}{\partial x} = -\frac{\partial}{\partial x} U(x, y) + \frac{\overline{U}_e}{x_e} = \mathcal{E}_x + \frac{\overline{U}_e}{x_e}, \tag{8}$$

where $\mathcal{E}_x = -\partial U / \partial x$ is the uncorrected axial electric field.

Simulation outputs

Every 5000 iterations, we output the electric field vector, electrostatic potential, and charge densities of all species evaluated at the cell centers. We additionally compute the moments of the each species' velocity distribution function (VDF) at every cell. In particular, we calculate the zeroth (density), first (bulk velocity vector), second (pressure tensor), and contracted third (heat flux vector) moments. We also save the number of particles and total energy in the system at each timestep. Lastly, we obtain code profiling information from the TinyProfiler tool built into WarpX, allowing us to investigate the costs of different parts of the code.

Parameter investigation

In addition to demonstrating the feasibility of Hall thrusters in WarpX, we also investigate in this work the sensitivity of the simulation results and performance to several numerical options. We describe these below along with the range of investigated parameters.

Particle sorting interval When running on GPUs, WarpX periodically sorts particles so that particles that are physically near to one another are also nearby in memory [18]. This improves performance by improving memory locality during the deposition step, where particle quantities are interpolated to the grid. However, it also introduces a small performance overhead (about 80% of the cost of a particle push). To find the optimal setting, we vary the sorting interval between 5 and 1000 iterations.

Particle resampling parameters WarpX supports particle resampling, which is useful to ensure the simulation is adequately resolved and has an even distribution of computational macro-particles throughout the domain. This is particularly useful if the simulation features continuous particle injection, as ours does, which could lead to an unnecessary build-up of particles in the injection region and a corresponding reduction in simulation speed. In this work, we use the *leveling-thinning algorithm* developed by Muraviev et al. [22], which is WarpX's default resampling option. This algorithm down-samples (merges) particles while trying to maintain an accurate representation of their velocity distribution function. This resampling is performed per-species and is controlled by three parameters:

1. `resampling_algorithm_target_ratio`, which corresponds to the ratio of the number of particles before a resampling step to the number after a resampling step.
2. `resampling_min_ppc`, which is the threshold number of particles per cell below which a cell will not be resampled.

3. `resampling_trigger_max_avg_ppc`, which defines the maximum number of particles per cell, averaged over the whole domain, above which resampling is triggered.

In our simulations, we set the first parameter to the WarpX default of 1.5, the second to 75 particles per cell, and vary the maximum particles per cell between 200 and 300.

Field solve tolerance For electrostatic simulations, WarpX uses a multigrid method to solve Poisson's equation to obtain the electric field and potential. Starting from the fields solved at the previous timestep, this method iteratively reduces the error in the solution of Poisson's equation until it reaches a user-specified relative tolerance (hereafter, "multigrid precision"). We test tolerances between 10^{-2} and 10^{-6} in this paper.

Semi-implicit scheme Finally, we evaluate in this work the usefulness of a semi-implicit algorithm on accelerating kinetic Hall thruster simulations. This algorithm, developed by D. Barnes (see Appendix of Ref. [23]) and recently implemented in WarpX by Groenewald et al. [24], replaces the normal Poisson's equation (Eq. 1) for the potential with

$$\nabla \cdot \left(1 + \frac{1}{4} C \Delta t^2 \omega_{pe}^2\right) \nabla \phi = -e n_e / \epsilon_0, \tag{9}$$

where $C > 1$ is an adjustable constant and $\omega_{pe} = \sqrt{e^2 n_e / m_e \epsilon_0}$ is the plasma frequency. This modified Poisson's equation acts stabilize the electron plasma mode by artificially raising the electron plasma frequency. In combination with the energy-conserving gathering scheme, this choice allows larger cell sizes and timesteps than are typical in explicit PIC schemes while leaving modes uncoupled to the electron plasma mode unaffected. A recent study found that this scheme was able to accelerate simulations of an electrostatic Penning discharge by two orders of magnitude [24], which shows promise for its use in Hall thruster simulations. In this paper, we evaluate the semi-implicit scheme at three grid resolutions: **2x** ($\Delta t = 1 \times 10^{-11}$ s, $N_x = 256$, $N_y = 128$), **4x** ($\Delta t = 2 \times 10^{-11}$ s, $N_x = 128$, $N_y = 64$) and **8x** ($\Delta t = 4 \times 10^{-11}$ s, $N_x = 64$, $N_y = 32$, while leaving all other parameters unchanged.

Barnes' algorithm resembles the well-known Direct-Implicit (DI) scheme [25], which is known to suffer from numerical heating at large timesteps [20]. To address this, we set C to 8 for the **2x and 8x** cases and 16 for the **4x** and cases, as these values yielded the smallest deviation in the time-averaged electron temperature from the baseline simulation.

Summary of simulation parameters

Table 1 summarizes the parameters employed in this work. The top half of the table contains the benchmark parameters common to all simulations, discussed in the section describing the Benchmark simulation, while the bottom half shows the ranges of the variable numerical parameters discussed above. The parameters of the baseline case are underlined. For the semi-implicit simulations, we use an initial particle count of 75 particles per cell, a multigrid precision of 10^{-3}, a sorting interval of 500 iterations, and no resampling.

We perform all but one of our simulations on a single Nvidia H100 GPU, with 80 GB of onboard memory [26]. This is a new GPU, launched in 2023 designed primarily for machine learning workloads but with large general-purpose compute capability. To determine how much of WarpX's performance compared to the benchmark depends on the code architecture versus advancements in hardware in the intervening five years, we run a single simulation using the baseline case parameters on an Nvidia V100. The V100 GPU was released in 2017 and is thus contemporaneous with the computational hardware used in the 2019 benchmark. We employ single-precision arithmetic for the particles and double-precision arithmetic for the field solve. As GPUs typically have significantly more single-precision processing power than double-precision, mixing precisions in this way accelerates the simulation while maintaining an acceptable level of accuracy.

Scaling

Finally, we analyze the scaling of WarpX's electrostatic (ES) solver to multiple GPUs on both a single node and across several nodes. To that end, we perform 3-D simulations of a uniform plasma in a periodic box. This selection has several advantages over the Hall thruster benchmark simulation. First, these simulations converge quickly (typically a few minutes) as opposed to the tens of hours required for the 2-D axial-azimuthal simulations. Second, they have very simple physics, allowing us to focus entirely on the scalability of WarpX's numerics. Finally, they do not rely on WarpX's python extension interface. The callbacks we have implemented, while performant at the scale of the benchmark, may not scale well to very large problems.

As the scalability of the particle push, charge deposition, and field gathering, as well as that of the electromagnetic (EM) solver have all been assessed in the past [18], we focus here on the scaling of the grid-based electrostatic field solver. In Table 2, we show the parameters used for our scaling tests. tarting with a workload of 32 cells in each dimension (workload 1), we gradually increase the grid resolution, incrementally doubling the resolution in each dimension, until the computation cannot fit in memory.

Table 2 Parameters for uniform plasma scaling tests

	N_x	N_y	N_z	Workload
Minimum workload	32	32	32	$2^0(1)$
Maximum workload	1024	1024	512	$2^{14}(16384)$
Particles per cell				1
Minimum GPUs				1
Maximum GPUs				32
GPUs per node				8
Domain size:				40×10^{-6} m
Plasma density				1×10^{25} m^{-3}
Electron thermal velocity				0.01 c
Timestep size				10^{-14} s
Maximum iterations				250

Table 3 MPI domain decomposition and GPU allocation strategy for scaling tests

	Number of GPUs					
	1	**2**	**4**	**8**	**16**	**32**
Nodes	1	1	1	1	2	4
MPI ranks in *x*-direction	1	2	2	2	4	4
MPI ranks in *y*-direction	1	1	2	2	2	4
MPI ranks in *z*-direction	1	1	1	2	2	4

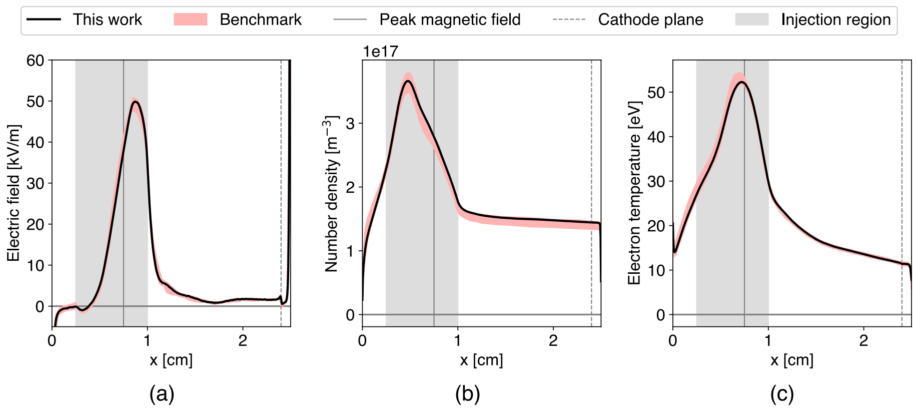


Fig. 3 **a** Electric field, **b** ion number density, and **c** electron temperature for the baseline simulation. The range of benchmark results from Case 2 of Charoy et al, 2019 [15] is indicated in pale red

The cluster we used has eight GPUs per node, so runs with 16 and 32 GPUs used two and three nodes, respectively. For a single GPU, workload 2^{13} is the largest able to run, with $(N_x, N_y, N_z) = (1024, 512, 512)$, while workload 2^{14} can run on two, four, and eight GPUs. For each workload and GPU count, we record the wall time after the simulation completed, as measured by WarpX. We assign one MPI rank per GPU and incrementally decompose the domain as described in Table 3.

Results

Benchmark simulations

In Table 5, we summarize the performance, in terms of wall time, of each of our simulations. This table includes only the results obtained using the explicit solver on the full-size grids. We first focus on the the results of our baseline simulation, highlighted in grey in this table, as well as the other cases from the 2019 benchmark. For the baseline simulation, we initialized the domain with 75 particles per cell, performed no resampling, sorted the particles every 500 iterations, used linear particle shape functions for the particles, and set the multigrid precision to 10^{-5}. In Fig. 3, we compare the results of this simulation to those of Case 2 of the 2019 benchmark. As that benchmark contained a number of different codes with slightly varying results, we show in light red the range of the benchmark results, rather than the result of any one code. It is apparent that our results lie well within the acceptable range of the benchmark, and our extensions to WarpX have been successful.

WarpX completed the full 20 μs simulation using the explicit solver in 1.81 days on the H100 GPU and 3.81 days on the V100 GPU. In comparison, typical wall times for this case ranged from 3 to 11 days in the 2019 benchmark, with the best result achieved by a code developed by the Princeton Plasma Physics Laboratory (PPPL, 2.5 days on 112 CPUs). The only GPU-based code that participated in the 2019 effort required 9 days to finish Case 2 on an Nvidia A100 GPU, a more powerful contemporary of the V100 [27, 28]. However, this code used an implicit particle pusher with a much larger computational overhead than the explicit pushing scheme employed by our WarpX simulations. As such, it is not directly comparable to our results.

We also report in Table 4 a summary of the time spent in by WarpX in different parts of the PIC cycle (see Fig. 2) in the first two cases. Cases 1 and 3—which have two and four times as many particles, respectively, as Case 2—took 58% and 179% longer than Case 2 to complete. In the baseline case, the computational time was split roughly evenly between particle-based and grid-based parts of the code. As the particle count increased in Cases 1 and 3 and grid size remained constant, the fraction of time WarpX spent in the particle routines increased by over 50%. We illustrate these data graphically in Fig. 4a. In Fig. 4b, we show a roofline plot of the performance on an H100 of the three main kernels of the PIC loop on the baseline case. We find that the gather-and-push and charge deposition kernels are memory-bound, limited primarily by the bandwidth of the DRAM and L2 caches. In contrast, the field solver is operating well-below the memory streaming limit. This may be due to the relatively small problem size.

In Table 5, we report the wall-clock time taken for each of the simulations in our numerical parameter investigation. We also report the time taken by an optimized explicit simulation, with numerical parameters chosen from the best results of each of our investigations. For this simulation we employed resampling at a 300 particle-per-cell resampling threshold, a sorting interval of 100 iterations, and a multigrid precision of 10^{-3}.

Effect of resampling

As expected, reducing the maximum allowable number of particles per cell improved performance, with a reduction in wall time of up to 25% at a threshold of 200 particles per cell. In contrast, a resampling threshold of 300 particles per cell had minimal effect on runtime (a decrease of about 1 hour), as the steady state particle count of the baseline simulation was 290 particles per cell of each species. Resampling was thus only triggered during the startup transient.

Despite the performance improvements made possible by resampling, we found that thresholds lower than the steady-state particle count adversely affected the accuracy of the simulation. In particular, while the time-averaged density and electric field profiles were very similar across all resampling thresholds, the electron temperature at the right boundary depended on the particular threshold chosen.

In Fig. 5, we show how the electron temperature changed as the maximum average particles per cell increased from 200 to 300 particles per cell. While the solution remains in good agreement with the benchmark result upstream of the location of maximum magnetic field, it deviates in the downstream half of the domain. At the right boundary, the electron temperature at a maximum resampling threshold of 200

Table 4 Performance for the baseline simulation using the explicit solver (Case 2, highlighted in grey) as well as Cases 1 and 3, on both H100 and V100 GPUs

Benchmark case	**Case 2**	Case 1	Case 3
Initial particles/cell	75	150	300
Final particles/cell	290	590	1200
Wall time (H100)	1.81 days	2.86 days	5.06 days
Wall time (V100)	3.81 days	N/A	N/A
Best benchmark sim [15]	2.5 days (112 CPU)	2.5 days (224 CPU)	2.5 days (448 CPU)
Fraction of time spent on PIC subroutines			
Particle gather and push	48.81 %	63.79%	73.11%
Field solve	40.86 %	24.21 %	16.63%
Charge deposition	8.43 %	10.72 %	9.63%
Python extensions	1.9 %	1.28 %	0.63%

The performance of the implicit solver is documented later, in Table 5

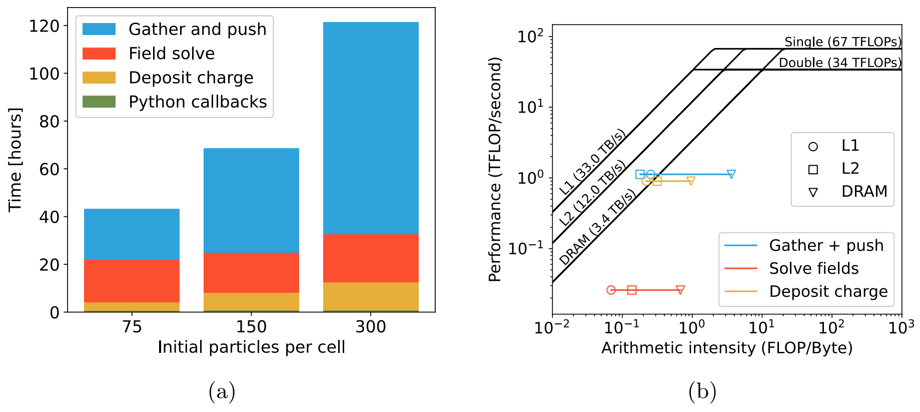


Fig. 4 **a** Performance of each main component of the PIC loop on each benchmark case. Data are the same as in Table 4. **b** Roofline plot of the performance of the three main kernels of WarpX's GPU solver on an H100 GPU for the baseline case (Case 2). L1 and L2 cache bandwidths are shown in addition to that of main memory. Additionally, we show the thresholds for single-precision ("Single") and double-precision ("Double") FLOPs

Table 5 Recorded wall times for each parameter set

Parameter value	Wall time
Initial particles per cell	
75	1.81 days
150	2.86 days
300	5.06 days
Resampling threshold	
No resampling	1.81 days
300	1.76 days
275	1.66 days
250	1.54 days
200	1.39 days
Particle sorting interval	
5 iterations	1.76 days
10 iterations	1.76 days
50 iterations	1.76 days
100 iterations	1.76 days
500 iterations	1.81 days
1000 iterations	1.80 days
Multigrid precision	
10^{-2}	1.59 days
10^{-3}	1.59 days
10^{-5}	1.81 days
10^{-6}	1.93 days
Optimized explicit simulation	**1.52 days (38 hours)**
Semi-implicit solver	
2x	11.5 hrs
4x	4 hrs
8x	2 hrs

The baseline case parameters are highlighted in grey and the results of the semi-implicit solver are indicated in yellow

particles per cell is 7 eV higher than that of the baseline case. To investigate why this might occur, we plot in Fig. 6 the ion density and electron temperature at the last timestep of the baseline case. This figure shows that in the region downstream of the peak magnetic field ($x > 0.75$ m), a long-wavelength mode develops where the electron temperature and density vary in phase with one another. In the high-density

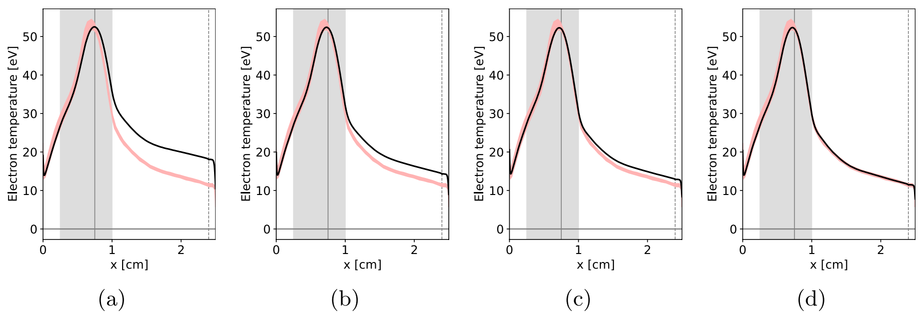


Fig. 5 Effect of resampling on electron temperature profiles, with resampling thresholds of **a** 200 particles per cell, **b** 250 particles per cell, **c** 275 particles per cell, and **d** 300 particles per cell. The light gray and red regions have the same meanings as in Fig. 3

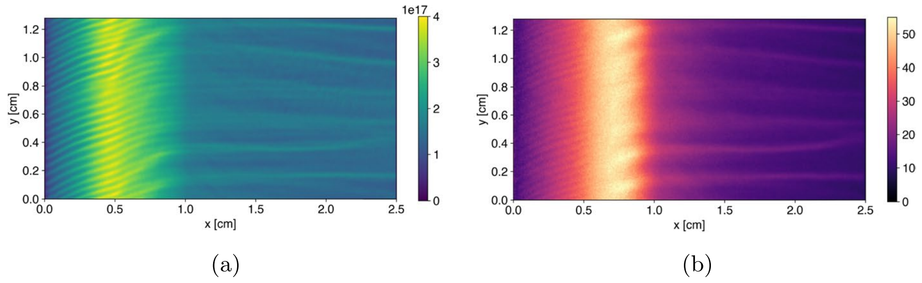


Fig. 6 **a** Ion density (m^{-3}) and **b** electron temperature (eV) of the baseline simulation at the last timestep ($t = 20\,\mu s$)

regions of this mode, the particle density may exceed the threshold for resampling. Resampling leaves the density (and therefore electric field, via Poisson's equation) unchanged, but may result in a small spread of the velocity distribution function, if the particle resolution in these high-density regions is not sufficiently high. This could then produce the higher electron temperatures observed in the right half of the domain. In Fig. 7, we show the electron velocity distribution function just upstream of the cathode line for resampling thresholds of 200 particles per cell, 250 particles per cell, and the baseline simulation (no resampling). As suggested by the larger electron temperatures, the VDFs are wider at lower resampling thresholds. This widening is limited to the axial and azimuthal axes, with the radial VDF remaining Maxwellian.

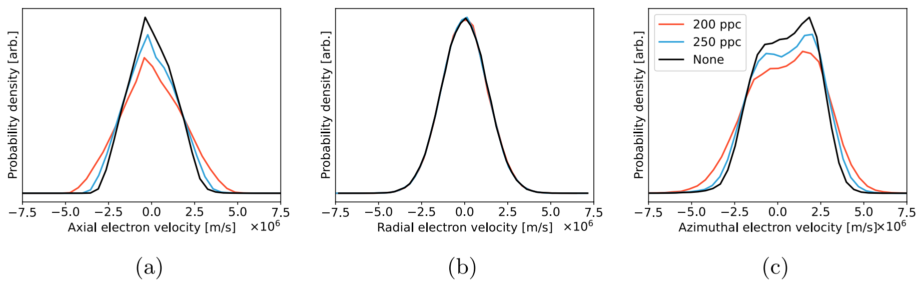


Fig. 7 Effect of maximum particles per cell (ppc) on **a** axial, **b** radial, and **c** azimuthal electron velocity distribution functions

In contrast, the axial and azimuthal distribution functions are non-Maxwellian, which may exacerbate numerical heating during resampling.

Given these results, it seems that resampling should be used with caution, at least in low-temperature plasmas. While resampling can improve performance significantly, it risks distorting the higher moments of the particles' velocity distribution function and altering the simulation in an unphysical manner if care is not taken to preserve these moments [29]. As such, the resampling threshold, if one is used, should be set to a value close to the expected steady-state particle count, so that resampling only occurs during transient events, provided that the steady-state solution is not dependent on these transients and that the resolution of these transients is not desired.

Sorting interval and multigrid precision

The results in Table 5 demonstrate that more frequent particle sorting typically results in better simulation performance, with a maximum speed-up of one hour and diminishing returns for sorting intervals below 100 iterations. We also found that that decreasing the tolerance of the field solver dramatically improved performance. This performance improvement was very coarse-grained—in the baseline case, the iterative algorithm used to solve the fields only needed to perform three iterations, on average, to converge on the requested relative tolerance of 10^{-5}. Increasing the tolerance to 10^{-3} decreased the number of iterations needed to two, and decreasing it to 10^{-6} increased the iteration count to four. Further increasing the tolerance to 10^{-2} had no effect, as two iterations were still required to converge to within this tolerance. We found that simulations performed at these higher tolerances were not qualitative or quantitatively different than those performed at lower tolerances and lay within the noise threshold of the PIC method and the run-to-run variance.

Semi-implicit solver

Performance

As shown in Table 5, the grid coarsening enabled by the use of the semi-implicit Poisson solver resulted in dramatic reductions in wall time. The 2x case completed in only 11.5 hours, a 3.5x speedup over the baseline case. The 4x and 8x cases finished even faster, at 4 and 2 hours, respectively. In Fig. 8, we show the results from the 2x, 4x, and 8x semi-implicit cases.

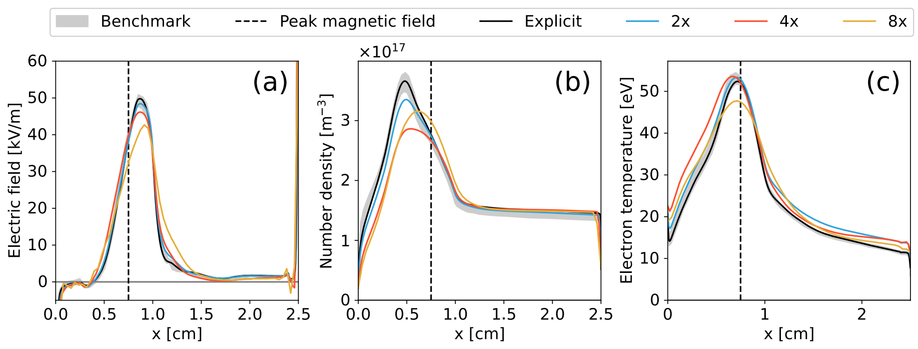


Fig. 8 **a** Axial electric field, **b** plasma density and **c** electron temperature for the baseline explicit case and the three semi-implicit cases

We find that the semi-implicit solver maintains good agreement with the benchmark up to 4x coarsening, particularly for the electron temperature. As the grid coarsens, the electric field profile broadens and the peak field decreases, from 50 kV in the baseline case to 43 kV in the 8x case. The peak plasma density also declines, from 3.6×10^{17} m^{-3} in the baseline case to 3×10^{17} m^{-3} in the 8x case. The electron temperature profile changes little between the baseline case and the 2x and 4x semi-implicit simulations, but the peak temperature decreases sharply in the 8x case, going from 52 eV to 44 eV.

In Fig. 9, we quantify the deviation in plasma density, electric field, and electron temperature from the benchmark for the chosen cases, as well as as a function of semi-implicit factor, timestep, and particle count. We calculate the deviation in a quantity y from the benchmark value y_b as

$$\text{Deviation} = \sqrt{\frac{\sum(\mathrm{y} - \mathrm{y_b})^2}{\sum \mathrm{y_b^2}}}, \tag{10}$$

where y has been interpolated to the same coordinates as y_b and values beyond the cathode line at $x = 2.4$ cm have been ignored.

In all cases, the deviation is less than 25% for the 4x and 8x cases and 10% in the 2x case, and tends to increases as the grid coarsens. Increasing the semi-implicit factor (C

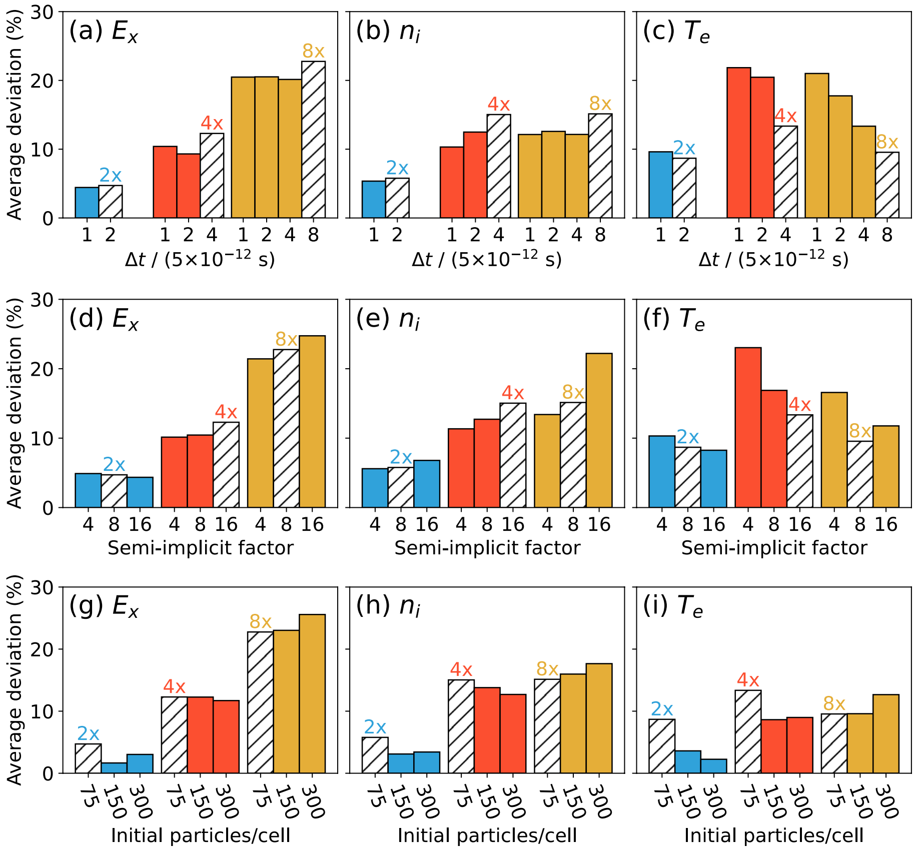


Fig. 9 Average deviation from the benchmark in plasma density, electric field, and electron temperature for different grid spacings, as a function of **a**-**c** timestep, **d**-**f** semi-implicit factor, and **g**-**i** particle count. The nominal 2x, 4x, and 8x cases from Table 1 are indicated with labels and as hatched bars

in Eq. 9) yields better agreement with the benchmark electron temperature at the cost of worse agreement with the plasma density and electric field. We attribute this to a reduction in numerical heating. Increasing timestep has little effect on the plasma density and electric field, but reduces the discrepancy in electron temperature by up to 50% in the 8x case. In the 2x case, increasing the particle count does appear to yield improved agreement, especially in electron temperature. This likely indicates a reduction in numerical heating. However, this trend is much less clear in the 4x and 8x cases.

In Fig. 10, we show how the axial ion velocity distribution function (IVDF) changes as the grid and time resolution are coarsened. We observe excellent agreement in the most probable velocity across all cases, with the ion acceleration profile becoming modestly more shallow as the grid coarsens. This makes sense given the reduction in peak electric field for the coarser cases observed in Fig. 8. Additionally, there is some disagreement in the near-anode region, particularly in the 8x case (Fig. 10d), as the grid spacing has grown larger than the anode sheath width in the baseline case. This leads to an expansion of the anode sheath, but does not affect the velocity outside of this region.

Plasma instabilities

To investigate the effect of the semi-implicit method and grid coarsening on the character of the global plasma instabilities, we show in Fig. 11 the 2D azimuthal electric field at the last timestep for each semi-implicit case, as well as for the explicit baseline solution. Consistent with the benchmark results in Ref. [15], we find that an instability develops early in the simulation and develops into a azimuthally-propagating wave with two modes—a short-wavelength mode near the anode and a long-wavelength mode in the right half of the domain. These basic characteristics of the explicit simulation persist in the semi-implicit simulations, but the wavelength of the short-wavelength mode gradually increases as the grid coarsens. We illustrate the changing dominant wavelength and

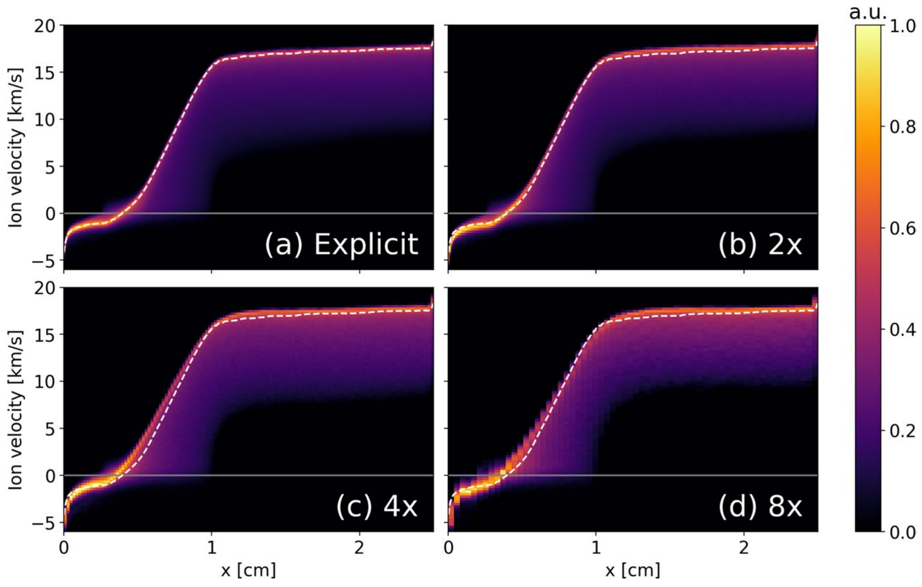


Fig. 10 Axial ion velocity distribution function for the explicit and semi-implicit cases. In each figure, we include for comparison the most probable velocity of the explicit case as a white dashed line

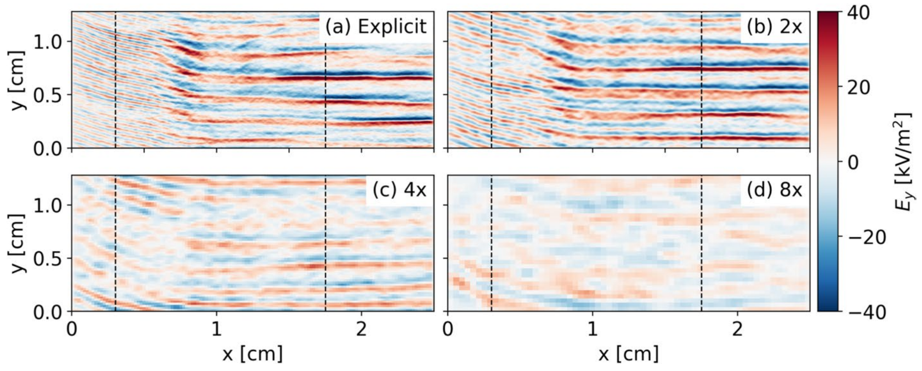


Fig. 11 Evolution of the azimuthal electric field as the grid is coarsened using the semi-implicit scheme. The vertical lines indicate $x = 0.3$ cm and $x = 1.75$ cm

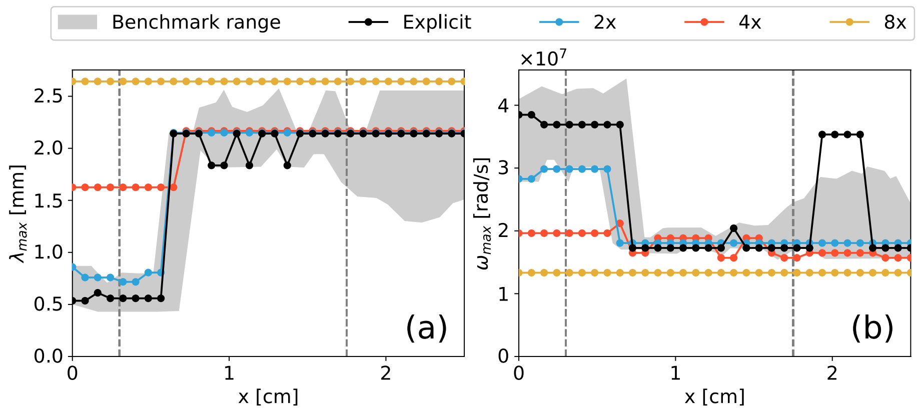


Fig. 12 Evolution of the **a** dominant azimuthal wavelength and **b** dominant azimuthal frequency as a function of axial location

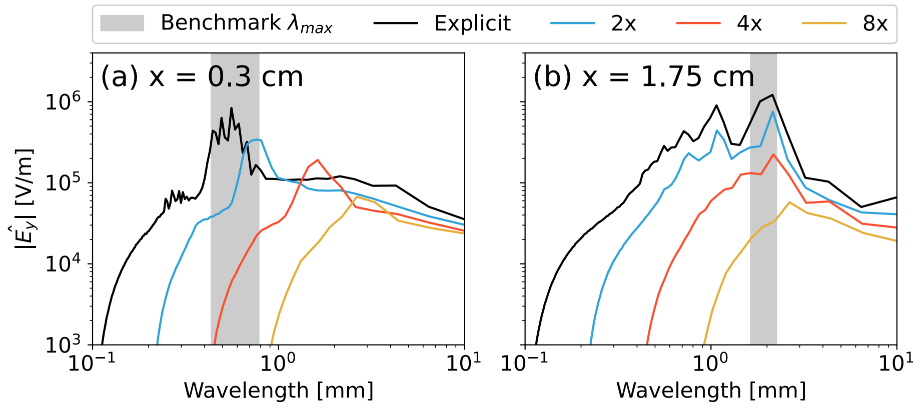


Fig. 13 Spectra of the azimuthal electric field wavelength at **a** x = 0.3 cm and **b** x = 1.75 cm. The dominant wavelengths from the 2019 benchmark are indicated in gray

frequency of these instabilities as one moves downstream in Fig. 12, and show the azimuthal wavelength spectra for each of the two modes in Fig. 13. The dominant frequencies and wavelengths agree with the benchmark for the 1x and 2x cases. However, the distinction between the upstream and downstream modes begins to diminish for the 4x case. In the 8x case, the discrepancy between these modes disappears entirely, and the same wavelength is dominant across the whole domain. In addition, the maximum azimuthal electric field declines by 40% in the 4x case, and 70% in the 8x case.

Given these changes in the character of the global instabilities, we were surprised that the time-averaged axial electric field remains consistent across all cases, with only a gradual reduction in peak field strength as the grid is coarsened. As the establishment and localization of this strongly-peaked electric field is one of the major consequences of anomalous electron transport in Hall thrusters, our results suggest that the magnitude of the transport may be relatively insensitive to some of the spectral characteristics of the azimuthal instability.

Convergence and numerical heating

In Fig. 14, we show the electron current emitted by the cathode and averaged electron temperature for each case. As in Ref. [15, 30], we use the former as a measure of both numerical convergence and anomalous transport level. The latter gives an indication of the degree of numerical heating/cooling, which must be monitored when using implicit PIC schemes. We find that in the 1x, 2x, and 4x cases, the electron current agrees well with the benchmark, but falls by a factor of 3 in the 8x case. This likely indicates a reduction in electron mobility, which makes sense given the reduction in azimuthal electric field amplitude seen in this case in Fig. 11.

Examining Fig. 14b, we observe a small increase in average electron temperature over the benchmark value of 2, 2.5, and 1 eV in the 2x, 4x, and 8x cases, respectively. This amount of numerical heating, while non-negligible, is small and may be difficult to distinguish from other sources of uncertainty in larger simulations with more self-consistent physics. As shown in Fig. 9i, the amount of numerical heating in the 2x and 4x cases may be reduced further by increasing the particle count.

Taken together, these results show that Barnes' semi-implicit method can significantly reduce the computational cost of Hall thruster simulations while preserving the bulk

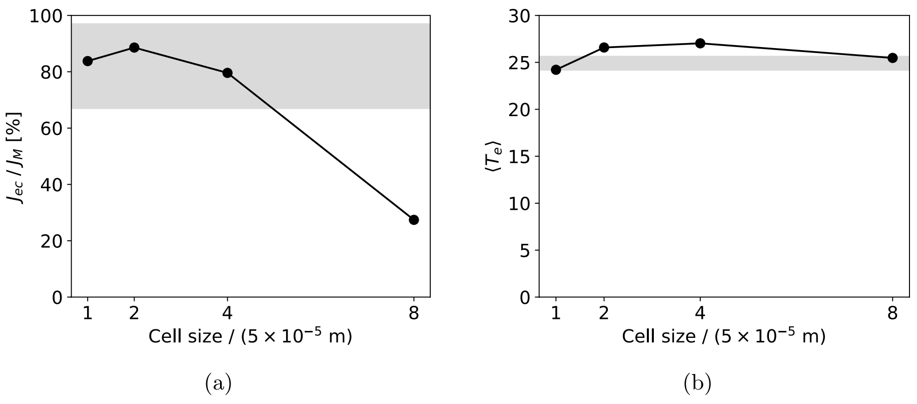


Fig. 14 **a** Cathode current as a fraction of the extracted ion current J_M for the 1x, 2x, 4x, and 8x cases. **b** The averaged electron temperature in the domain for the same cases. For both plots, the range of values in the 2019 benchmark is indicated in gray

plasma behavior. In particular, the 2x case yields averaged deviations of less or equal to than 10% across plasma density, electron temperature, and electric field and preserves the character of the azimuthal plasma instabilities. Despite the greater speedups seen in the 4x and 8x cases, their increased deviations from the benchmark of up to 25% and the observed modification of the wavelength spectra make them difficult to trust for predictive simulations.

We also believe that the potential speedups may be greater than those observed here. In the 2x case, the GPU needs to do eight times less work than in the explicit case, as it needs a quarter of the cells and half as many timesteps. Despite this, we only observed a speedup of 3.5 times over the baseline case. This is likely because the benchmark simulation is relatively small for this GPU, and some amount of its computational cost is purely overhead. This overhead remains roughly constant and begins to dominate as the workload is reduced. For larger simulations, the proportion of work dedicated to this overhead decreases, and the potential speedups seen from using the semi-implicit method increase. Three-dimensional simulations should see even larger speedups, as the grid can be coarsened along an additional dimension.

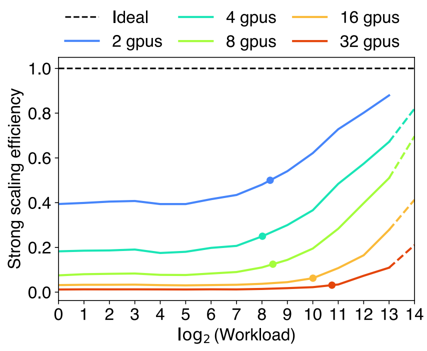
Scaling analysis of electrostatic solver

One good measure of the how well a parallel code scales to multiple processors is the *strong scaling efficiency*, given by

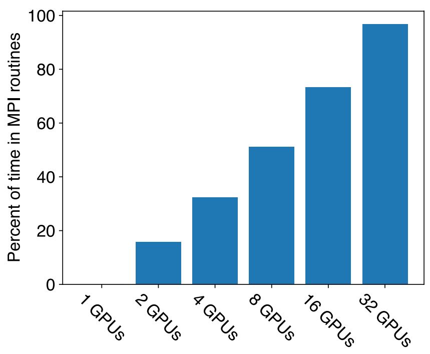
$$\text{Strong scaling eff.}(N) = \frac{t(1)}{Nt(N)}, \tag{11}$$

where N is the number of processors and $t(N)$ is the execution (wall) time when running the code with N processors. The factor $t(1)/t(N)$ gives the *speedup*, which for a perfectly parallelizable code is equal to N. The strong scaling efficiency thus measures the fraction of the ideal speedup that we observe in practice.

In Fig. 15a, we show how the strong scaling efficiency computed at each workload varies as a function of GPU count. For all workloads up to 2^{13}, we measured the strong scaling efficiency with respect to the single-GPU case. For the final case (workload 2^{14}) we



(a)



(b)

Fig. 15 **a** Strong scaling efficiency for uniform plasma simulations. Markers indicate the workload at which speedup is equal to one. Dashed lines represent strong scaling efficiency measured with respect to the 2-GPU case, i.e. $2t(2)/Nt(N)$. **b** Fraction of wall time spent in inter-process communication for workload 2^{13}

instead measured it with respect to the 2-GPU case, as the problem could no longer fit in the memory of a single GPU. We find that while speedups in the electrostatic solver can be obtained using multiple H100 GPUs, this speedup only becomes relevant for large workloads. For instance, when running the largest uniform plasma simulation that can fit in memory on a single H100 (workload 2^{13}), the speedup is only 1.75. The culprit for this less-than-ideal scaling behavior is the increased overhead of MPI communication as the number of GPUs increases.

In Fig. 15b, we plot the fraction of time spent on MPI communication for each GPU count for workload 2^{13}. We find that the MPI overhead increases with the number of GPUs, up to 95% in the 32 GPU case. The reason for this increased overhead lies with Poisson's equation. Solving Poisson's equation requires communication across MPI ranks many times per iteration as smoothing is applied at each level of the multigrid algorithm. In contrast, updating the electromagnetic fields can be done largely locally using finite differencing and minimal inter-process communication.

There are a few caveats to these results, however. Scaling studies of this kind are sensitive to the configuration of the systems used to run them, and it is possible that our results are particular to the specific cluster we used. Additionally, while we found that the electrostatic solver in particular scaled sub-ideally, we were able to verify that the particle parts of the code and the electromagnetic solver exhibited excellent scaling, as previously reported in Ref. [18]. Given that, it might be possible that for certain very large problems, the electromagnetic solver could outperform the electrostatic solver for Hall thruster simulations, even though the EM solver must take significantly smaller timesteps than the ES solver.

Finally, we note that while WarpX's field solver may not exhibit perfect scaling, other Poisson solvers for electrostatic PIC applications may be able to perform better. In particular, the GPU ion thruster plume code CHAOS [13] uses a conjugate-gradient method to solve Poisson's equation. This solver, when implemented with careful regard for the layout of the problem in GPU memory and the amount of MPI communication required, is reported to yield good strong scaling efficiency up to 128 GPUs.

Role of GPU hardware

The performance on the older Nvidia V100 (3.81 days compared to 1.81 days on the H100 GPU) is still significantly faster than all but two of the codes in the 2019 benchmark. This indicates that at least some of the improvements seen in the explicit simulations this work are due to WarpX's code architecture and the inherent advantages of a GPU for PIC simulations. However, the simulation on the H100 was nearly twice as fast with no change in algorithm or configuration. This result highlights the important role of increasingly powerful GPU hardware in accelerating kinetic simulations of low-temperature plasma devices like Hall thrusters.

Despite WarpX's good benchmark performance across hardware generations, one major gap in its abilities is lack of support for the reduced-precision and tensor computations needed to fully exploit newer AI-focused GPUs like the H100 [26]. In particular, while the H100 PCI-e GPU has a capacity of 34 and 67 teraFLOPS (10^{12} floating point operations per second), respectively, for non-tensor double- and single-precision floating point operations, respectively, it can support up to 756 teraFLOPS for TF32 tensor operations and 1513 FLOPS for FP16 operations. New algorithms for PIC that can effectively make use of these reduced-precision operations may lead to even more dramatic performance improvements. As the demand for increasingly powerful GPUs for machine learning and artificial intelligence applications increases, it is likely that even greater speed-ups in particle-in-cell simulations of Hall thrusters will be made possible, provided the codes can make efficient use of the new hardware.

Kinetic simulations in an engineering context

Combining increases in hardware capabilities with new algorithms for reducing noise and improving the parallel efficiency of PIC simulations may bring kinetic Hall thruster simulations down in cost enough to be useful in an engineering contexts. The main challenge remaining is the long simulation times needed to adequately resolve the dynamics of real thrusters. Including real ionization, rather than a fixed source term, introduces breathing mode oscillations which have frequencies on the order of 10 kHz [1]. As such, simulations that capture these oscillations must run for timescales of ~ 1 ms, about 50 times longer than the simulation durations in this work. Additionally, recent 3-D particle-in-cell simulations of Hall thrusters have demonstrated that many important aspects of the instabilities governing anomalous transport are not well-resolved by 2-D axial azimuthal simulations [8]. Even accounting for significant improvements in hardware, these constraints mean kinetic, whole-device Hall thruster simulations may still require wall times measuring in the months. However, when accounting for the time needed to build a thruster, and collect the data necessary to calibrate current non-predictive engineering models of Hall thrusters [3], it is still possible that kinetic simulations may soon become usable in engineering applications.

Conclusion

In this work, we have demonstrated the applicability of the open source particle-in-cell code WarpX for kinetic Hall thruster simulations. To do this, we simulated the well-known 2-D axial azimuthal benchmark of Charoy et al., and found that the results we obtained agreed satisfactorily with those previously published. Next, we investigated the impact of a variety of numerical parameters on the simulation performance and physics. We found that while resampling particles in regions of high densities has the potential to significantly speed up simulations, the method employed here appears to produce unphysical particle heating when the resampling threshold is too low. Therefore, this technique must be applied with care. In comparison to the effect of resampling, the precision of the multigrid Poisson solver had little impact on the physical output of the simulation. The performance implications of the particle sorting interval were relatively minor, with slightly better performance seen at shorter sorting intervals. However, reducing the Poisson solver relative tolerance from 10^{-5} to 10^{-3} accelerated the simulation by about 13%, or 5 hours, with no visible impact on solution quality. We then demonstrated the impact of recent improvements in GPU hardware by performing one simulation on an older Nvidia V100 GPU. This simulation took over twice as long to complete as our baseline simulation, which used a newer Nvidia H100.

We then assessed a semi-implicit formulation of Poisson's equation recently implemented into WarpX. Our simulations using this scheme used far fewer grid cells than our baseline case and longer timesteps. The results of these simulations showed good agreement with the benchmark when using cell sizes and timesteps up to four times longer than the baseline case. In this case, the simulation completed in just four hours. Simulations performed at even coarser grid resolutions were stable and showed qualitative, albeit worsened, agreement with the benchmark. These results show that this solver is capable of producing dramatic reductions in computational cost and time with only minor losses in fidelity.

Finally, we assessed the scalability of the WarpX's electrostatic solver across multiple GPUs on a uniform plasma test case. We found that increasing the GPU count does yield speedups for problems that saturate a single GPU, but the parallel efficiency was less than desired. We attribute this inefficiency to the large amount of MPI data-transfer incurred by the multigrid Poisson solver, and found that the other parts of the code scaled well to arbitrary computational resources.

As WarpX is an open source code, this work provides a common baseline for researchers to compare to and expand upon. Our results highlight the potential of advancements in GPU hardware for accelerating kinetic simulations of Hall thrusters and similar low-temperature plasma devices and suggests that such simulations could soon be viable for use in certain engineering contexts.

Acknowledgements
This work was funded by Los Alamos National Laboratories under the project "Algorithm/Software/Hardware Co-design for High Energy Density applications" at the University of Michigan, and used computing resources provided by an AFOSR DURIP under Program Manager Dr. Fariba Fahroo and grant number FA9550-23-1-006. The authors acknowledge additional computational resources and support provided by Advanced Research Computing, a division of Information and Technology Services at the University of Michigan. Lastly, we would like to acknowledge the WarpX developers and community for their assistance at many stages in this work. In particular, we would like to highlight Roelof Groenewald for his work implementing the semi-implicit electrostatic solver into WarpX.

Authors' contributions
T.M. helped develop the ideas, programmed and ran the simulations, analyzed the data, and wrote the manuscript. A.G. secured funding, helped develop the ideas, and reviewed the manuscript.

Data availability
WarpX is an open-source code, available on Github at https://github.com/BLAST-WarpX/warpx. Our simulations were run on version 24.05 through 25.02. The scripts used to run and analyze the simulations in this work are available on Github at https://github.com/archermarx/warpx-hall.

Declarations

Competing interests
The authors declare no competing interests.

Received: 20 December 2024 Accepted: 28 April 2025
Published online: 09 May 2025

References
1. Boeuf JP (2017) Tutorial: Physics and modeling of Hall thrusters. J Appl Phys 121:011101. https://doi.org/10.1063/1.4972269
2. Mikellides IG, Ortega AL (2019) Challenges in the development and verification of first-principles models in Hall-effect thruster simulations that are based on anomalous resistivity and generalized Ohm's law. Plasma Sources Sci Technol 28:48. https://doi.org/10.1088/1361-6595/aae63b
3. Marks TA, Jorns BA (2023) Challenges with the self-consistent implementation of closure models for anomalous electron transport in fluid simulations of H all thrusters. Plasma Sources Sci Technol 32(4):045016. https://doi.org/10.1088/1361-6595/accd18
4. Cappelli MA, Young CV, Cha E, Fernandez E (2015) A zero-equation turbulence model for two-dimensional hybrid Hall thruster simulations. Phys Plasmas 22. https://doi.org/10.1063/1.4935891
5. Lafleur T, Baalrud SD, Chabert P (2016) Theory for the anomalous electron transport in Hall effect thrusters. II. Kinetic model. Phys Plasmas 23:11101. https://doi.org/10.1063/1.4948496

6. Jorns B (2018) Predictive, data-driven model for the anomalous electron collision frequency in a Hall effect thruster. Plasma Sources Sci Technol 27:104007. https://doi.org/10.1088/1361-6595/aae472
7. Marks TA, Jorns BA (2023) Evaluation of algebraic models of anomalous transport in a multi-fluid Hall thruster code. J Appl Phys 134(15):153301. https://doi.org/10.1063/5.0171824
8. Villafana W, Cuenot B, Vermorel O (2023) 3-D particle-in-cell study of the electron drift instability in a Hall thruster using unstructured grids. Phys Plasmas 30:033503. https://doi.org/10.1063/5.0133963
9. Vay JL, Huebl A, Almgren A, Amorim LD, Bell J, Fedeli L, Ge L, Gott K, Grote DP, Hogan M, Jambunathan R, Lehe R, Myers A, Ng C, Rowan M, Shapoval O, Thévenet M, Vincenti H, Yang E, Zaïm N, Zhang W, Zhao Y, Zoni E (2021) Modeling of a chain of three plasma accelerator stages with the warpx electromagnetic pic code on gpus. Phys Plasmas 28(2):023105. https://doi.org/10.1063/5.0028512
10. Farber R (2022) WDMApp - the first simulation software in fusion history to couple tokamak core to edge physics. https://www.exascaleproject.org/highlight/wdmapp/. Accessed 6 Jan 2025.
11. Fedeli L, Huebl A, Boillod-Cerneux F, Clark T, Gott K, Hillairet C, Jaure S, Leblanc A, Lehe R, Myers A, Piechurski C, Sato M, Zaim N, Zhang W, Vay JL, Vincenti H (2022) Pushing the frontier in the design of laser-based electron accelerators with groundbreaking mesh-refined particle-in-cell simulations on exascale-class supercomputers. In: SC22: International Conference for High Performance Computing, Networking, Storage and Analysis. pp 1–12. https://doi.org/10.1109/SC41404.2022.00008
12. Fierro A, Dickens J, Neuber A (2014) Graphics processing unit accelerated three-dimensional model for the simulation of pulsed low-temperature plasmas. Phys Plasmas 21(12):123504. https://doi.org/10.1063/1.4903330
13. Jambunathan R, Levin DA (2018) CHAOS: An octree-based PIC-DSMC code for modeling of electron kinetic properties in a plasma plume using MPI-CUDA parallelization. J Comput Phys 373:571–604. https://doi.org/10.1016/j.jcp.2018.07.005
14. Vay JL, Almgren A, Bell J, Ge L, Grote D, Hogan M, Kononenko O, Lehe R, Myers A, Ng C, Park J, Ryne R, Shapoval O, Thévenet M, Zhang W (2018) Warp-X: A new exascale computing platform for beam-plasma simulations. Nucl Instrum Methods Phys Res Sect A Accelerators Spectrometers Detectors Assoc Equip 909:476–479. https://doi.org/10.1016/j.nima.2018.01.035
15. Charoy T, Boeuf JP, Bourdon A, Carlsson JA, Chabert P, Cuenot B, Eremin D, Garrigues L, Hara K, Kaganovich ID, Powis AT, Smolyakov A, Sydorenko D, Tavant A, Vermorel O, Villafana W (2019) 2-D axial-azimuthal particle-in-cell benchmark for low-temperature partially magnetized plasmas. Plasma Sources Sci Technol 28(10):105010. https://doi.org/10.1088/1361-6595/ab46c5
16. Marks TA, Gorodetsky AA (2024) Hall thruster simulations in WarpX. In: 38th International Electric Propulsion Conference, Toulouse, France. IEPC paper #409. https://doi.org/10.7302/23491
17. Boeuf JP, Smolyakov A (2019) Landmark plasma test cases. https://jpb911.wixsite.com/landmark/test-cases. Accessed 2 Dec 2024
18. Myers A, Almgren A, Amorim L, Bell J, Fedeli L, Ge L, Gott K, Grote D, Hogan M, Huebl A, Jambunathan R, Lehe R, Ng C, Rowan M, Shapoval O, Thévenet M, Vay JL, Vincenti H, Yang E, Zaïm N, Zhang W, Zhao Y, Zoni E (2021) Porting WarpX to GPU-accelerated platforms. Parallel Comput 108:102833. https://doi.org/10.1016/j.parco.2021.102833
19. Lewis H (1970) Energy-conserving numerical approximations for vlasov plasmas. J Comput Phys 6(1):136–141. https://doi.org/10.1016/0021-9991(70)90012-4
20. Sun H, Banerjee S, Sharma S, Powis AT, Khrabrov AV, Sydorenko D, Chen J, Kaganovich ID (2023) Direct implicit and explicit energy-conserving particle-in-cell methods for modeling of capacitively coupled plasma devices. Phys Plasmas 30(10):103509. https://doi.org/10.1063/5.0160853
21. Tyushev M, Papahn Zadeh M, Chopra NS, Raitses Y, Romadanov I, Likhanskii A, Fubiani G, Garrigues L, Groenewald R, Smolyakov A (2025) Mode transitions and spoke structures in e×b penning discharge. Phys Plasmas 32(1):013511. https://doi.org/10.1063/5.0238577
22. Muraviev A, Bashinov A, Efimenko E, Volokitin V, Meyerov I, Gonoskov A (2021) Strategies for particle resampling in PIC simulations. Comput Phys Commun 262:107826. https://doi.org/10.1016/j.cpc.2021.107826
23. Barnes DC (2021) Improved C^1 shape functions for simplex meshes. J Comput Phys 424:109852. https://doi.org/10.1016/j.jcp.2020.109852
24. Groenewald R, Barnes DC, Tyushev M, Zhang W, Huebl A, Necas A, Smolyakov A, Vay JL, Tajima T, Dettrick S (2024) New semi-implicit electrostatic particle-in-cell method to extend scope of the exascale WarpX code. In: The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC24). https://sc24.supercomputing.org/proceedings/poster/poster_pages/post233.html. Accessed 6 Jan 2025.
25. Langdon A, Cohen BI, Friedman A (1983) Direct implicit large time-step particle simulation of plasmas. J Comput Phys 51(1):107–138. https://doi.org/10.1016/0021-9991(83)90083-9
26. Nvidia Corporation (2024) NVIDIA H100 Tensor Core GPU Datasheet. Nvidia Corporation. https://resources.nvidia.com/en-us-tensor-core/nvidia-tensor-core-gpu-datasheet. Accessed 18 Jun 2024
27. Nvidia Corporation (2020) NVIDIA V100 Tensor Core GPU Datasheet. Nvidia Corporation. https://images.nvidia.com/content/technologies/volta/pdf/volta-v100-datasheet-update-us-1165301-r5.pdf. Accessed 18 Jun 2024
28. Nvidia Corporation (2024) NVIDIA A100 Tensor Core GPU Datasheet. Nvidia Corporation. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-nvidia-us-2188504-web.pdf. Accessed 18 Jun 2024
29. Faghihi D, Carey V, Michoski C, Hager R, Janhunen S, Chang C, Moser R (2020) Moment preserving constrained resampling with applications to particle-in-cell methods. J Comput Phys 409:109317. https://doi.org/10.1016/j.jcp.2020.109317
30. Boeuf JP, Garrigues L (2018) E × B electron drift instability in Hall thrusters: Particle-in-cell simulations vs. theory. Phys Plasmas 25:061204. https://doi.org/10.1063/1.5017033