Hall thruster simulations in WarpX

IEPC-2024-409

Presented at the 38th International Electric Propulsion Conference, Toulouse, France
June 23-28, 2024

Thomas Marks* and Alex A. Gorodetsky†
University of Michigan, Ann Arbor, Michigan, 48105, United States of America

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 LANDMARK axial-azimuthal benchmark reported by Charoy et. al. in 2019. Additionally, a range of numerical and solver parameters are investigated in order to find those which yield the best performance. By extending the code via its python interface, it is found that WarpX can simulate the benchmark case in 3.8 days on an Nvidia V100 hailing from the same era as the original benchmark, and just in 1.8 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, but that particle resampling may artificially widen electron velocity distribution functions, leading to unphysical heating. Using the results of the parameter investigation, an optimized simulation is then performed which completes the benchmark in just 36 hours on a single GPU. 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.

I. Introduction

Kinetic whole-device simulations of Hall thruster discharges have long been too expensive for use in engineering contexts. While capable of resolving the physics driving the growth and growth and saturation of micro-instabilities 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, any Hall thruster 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 engineering simulations of real devices. These stringent requirements and the prohibitively-long run-times that result have led to such 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, anomalous electron transport has been incorporated into simulations via the inclusion of ad-hoc transport coefficients. This approach has been very successful at producing converged simulations that match experimental data.[2] However, the transport coefficients must be tuned per-thruster or even per-operating condition, and are not typically generalizable across devices or operating regimes.[3] Other authors have attempted to remedy this problem by developing closure models of how the micro-turbulence should scale with bulk fluid plasma properties, i.e. temperature, density, and velocity. However, to data, no such model has proved capable enough to be recommended for general use in engineering applications.[4]

In light of the shortcomings of these lower-fidelity approaches, and with recent advancements in computing hardware, there has been renewed interest in applying kinetic simulation to the problem of whole-device Hall thruster simulation. For example, Vilafana, Cuenot, and Vermorel presented 3D, particle-in-cell simulations of a simplified Hall thruster on an unstructured mesh[5]. Additionally, large-scale multi-GPU computing has

*Postdoctoral Research Fellow, Department of Aerospace Engineering, marksta@umich.edu
†Associate Professor, Department of Aerospace Engineering

recently produced unprecedented speed-ups in kinetic simulations of other plasma devices, such as tokamaks [6] and plasma wakefield accelerators. [7] While significant differences exist between these systems and Hall thruster plasmas, the challenges of whole device simulation—namely that the a large range of length- and time- scales must be resolved—are similar. The recent success of massively-parallel kinetic simulations in these devices s suggests that a similar approach may be able to provide dramatic speed-ups in kinetic simulations of Hall thrusters.

In this work, we apply WarpX, a highly-optimized open-source, massively-parallel particle-in-cell code, to the problem of Hall thruster simulation. WarpX was designed to scale to the largest supercomputing clusters. [8] 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 2D axial azimuthal benchmark of Charoy et al [9], which concerns the growth of the electron cyclotron drift instability thought to be a key driver of anomalous electron transport in Hall thrusters in a simplified Hall thruster geometry. While this benchmark simulation lacks some important physics present in real Hall thrusters, such as ionization and neutral dynamics, it captures enough of the physics driving anomalous transport to be useful as a starting point in our efforts to develop scalable, GPU-enabled Hall thruster simulations. We demonstrate both WarpX's ability to efficiently solve the problem on hardware contemporaneous to that of the original benchmark, as well as the large speedups made possible by recent developments in GPU hardware.

This paper is organized as follows. In Section II, we describe the conditions of the benchmark, the capabilities of the WarpX code, and our modifications to WarpX to support Hall thruster simulations. We then provide details of the numerical parameters investigated in our simulations. In Section III, we present the results of our study. We demonstrate that our simulations in WarpX are capable of matching the benchmark results at a significantly-reduced computational cost compared to previous efforts at this benchmark, and assess the impact of different simulation options on performance. We discuss these results in the context of past kinetic simulation efforts and improvements in GPU hardware. Finally, in Section IV, 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.

II. Methods

A. Benchmark simulation

In this section we describe our target benchmark simulation — Case 2a of the LANDMARK low-temperature plasma benchmark effort. [10] This case is designed to capture the main kinetic effects governing Hall thruster discharges – namely the onset and growth of drift-driven turbulence. As such, it is a good test case for kinetic codes targeting Hall thruster applications. In particular, we compare our efforts to results of several codes from throughout the community, reported by T. Charoy et al in 2019 [9]. In this section, we briefly summarize the conditions of this benchmark. For a more detailed description of the benchmark conditions and its results, the reader is referred to the original paper.

The benchmark is a two-dimensional axial-azimuthal particle-in cell simulation of a simplified Hall thruster geometry. The simulation domain is shown in Figure 1, where the axial (x) and azimuthal (y) dimensions have lengths of $L_x = 2.5$ cm and $L_y = 1.28$ cm, respectively. The azimuthal dimension is assumed periodic and no curvature effects are considered. A magnetic field $B_z(x)$ with a piecewise-Gaussian profile is applied to the domain. This field points purely in the radial (z) direction, and the maximum magnetic field strength of 10 mT occurs at $x = 0.75$ cm.

We divide the computational domain into 512 cells in the axial direction and 256 cells in the azimuthal direction. At this grid resolution, the corresponding axial and azimuthal grid spacings are $\Delta x = 4.88 \times 10^{-5}$ m and $\Delta y = 5 \times 10^{-5}$ m, respectively. We employ a simulation timestep of 5×10^{-12} seconds. 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 particle-in-cell method.

The benchmark does not include the dynamics of neutral atoms; instead, we inject electron-ion airs according to a specified ionization rate profile within the interval bounded by $x = 0.25$ cm and $x = 1$ cm. This "injection region" is depicted in darker blue in Figure 1. Injecting particles in this manner allows simulations to avoid resolving both ionization oscillations and start-up transients. As a result, the simulation settles to a steady state within 20 microseconds of simulation time, instead of the 500 microseconds or more required if these effects were included. The temperatures of the newly-injected electrons and ions are 10 and 0.5 eV, respectively. Between the anode at $x = 0$ and a "virtual cathode" at $x = 2.4$ cm, we apply a DC

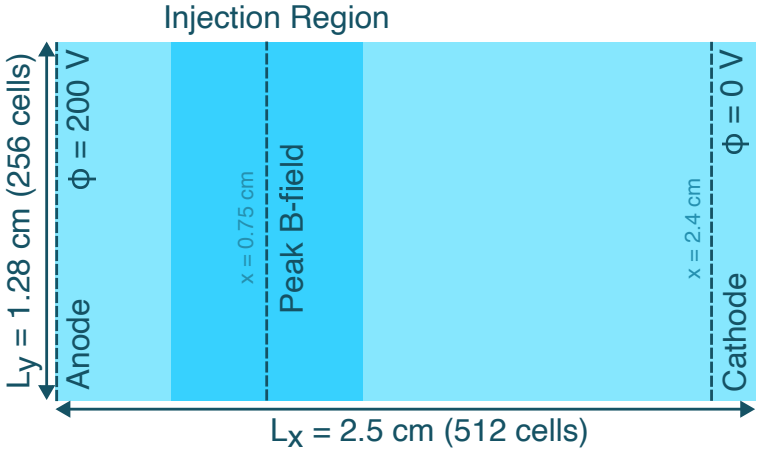

Figure 1: The 2D axial-azimuthal benchmark simulation domain.

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 original benchmark considers three cases, differentiated by the weight of the computational macroparticles, and therefore by the number of particles in the simulation–Case 1, initialized with 150 particles per cell, Case 2, with 75, and Case 3, with 300 initial particles per cell. In this work, we simulate all three of these cases, but treat the second of these cases as our baseline case when performing our numerical parameter investigation.

The initial condition of the simulation is a uniform plasma with a plasma density of $n_e = 5 \times 10^{16}\,\mathrm{m}^{-3}$, with electron and ion temperatures of 10 and 0.5 eV, respectively. We run the simulation for 20 µs, which corresponds to four million timesteps, and report plasma properties averaged over the last four microseconds of the run. Finally, we summarize the benchmark parameters in Table 1.

B. 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.[8] 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[12]-dominated clusters.

WarpX is based on AMReX, a high-performance adaptive mesh-refinement (AMR) framework.[13] While we do not employ AMR in this work, it may prove enabling for future Hall thruster simulations, as regions of the discharge with small Debye lengths, i.e. those with high densities and/or low temperatures, could be resolved without requiring unduly-small cell sizes throughout the domain.

In this work, we employ single-precision arithmetic for particle advancement 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 gives a significant speed-up while maintaining an acceptable level of accuracy.

1. Summary of WarpX's solution procedure

As with most particle-in-cell codes, WarpX performs four main actions at each timestep when running in electrostatic mode. 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".

2. **Push particles**: The particles are moved to new positions based on their velocities and the timestep, while their velocities are updated using the fields gathered in the previous step. By default, and in our simulation, WarpX uses a relativistic extension of the well-known Boris scheme for particle advancement.

This algorithm is explicit and second-order in time, and has the advantage of exactly preserving particle orbits around magnetic field lines.

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, $8.854 \times 10^{-12}\,\mathrm{F\,m^{-1}}$. The electric field is then gathered to the particles and the loop is repeated.

Internally, WarpX fuses the first two steps (gather and push) into a single step which can be efficiently executed. Taken together, the steps listed above comprise a Monte Carlo approach[14] to the solution of the time-dependent Vlasov-Poisson system of equations for a collisionless plasma.[15] However, these steps on their own are not sufficient to simulate a Hall thruster. Hall thrusters include 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 and establish current continuity. To handle Hall thruster simulations, WarpX must be extended to handle these additional effects.

2. Extensions for benchmark simulations

WarpX provides the ability for end-users to extend its physics without altering its internal structure. 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. These callbacks can both view and modify the state of the simulation, whether on CPU or GPU. This feature enables us to implement Hall thruster-related functionality without modifying the source code. There are three parts of the benchmark which are not included in WarpX's core functionality and require new implementations: the creation of particles in the injection region, the injection of particles at the cathode to support current continuity, and the zero-volt internal Dirichlet boundary condition at the cathode plane. Here, we briefly describe each of these components, and give some details about their implementation into WarpX. We illustrate how each of these extensions slots into WarpX's main loop in Figure 2. Additionally, we provide a link to our implementation at the end of the paper.

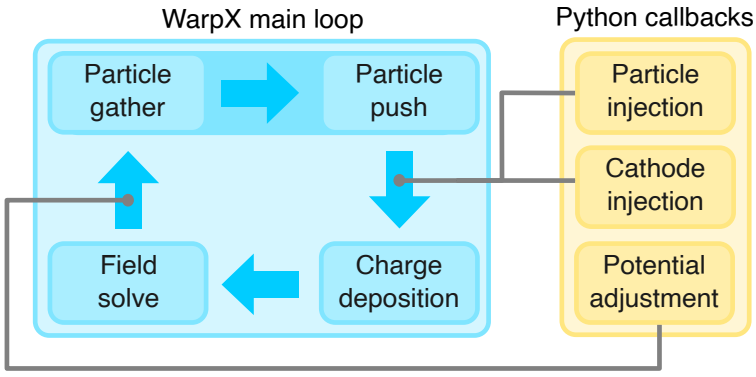

Figure 2: WarpX's main computation loop, including the Python callbacks implemented to support the benchmark simulations.

Particle injection

At every timestep, electron-ion pairs are created in the injection region according to a prescribed ionization profile. Per the benchmark conditions[9], the ionization rate is given as a function of axial position as

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

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

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

Here, W_0 is the base particle weight, or the number of real particles represented by a single computational macro-particle at simulation startup. This is computed as

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

With the number of initial particles per cell, $N_{ppc,ini}$, equal to 75, this gives a base weight of $W_0 = 1.627 \times 10^6$ real particles per macro-particle. Plugging this into Eq. 3, we find that we need to inject 98.195 particles per timestep of each species.

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. Inside of this callback, we first obtain the containers holding the electrons and the ions. We then sample a uniform random number between zero and one to determine whether to inject 98 or 99 particles, as follows

$$N_{inject}(t) = \lfloor N_{inject,0} \rfloor + \begin{cases} 1 & r_0 < (N_{inject,0} - \lfloor N_{inject} \rfloor), \;\; r_0 \sim \mathcal{U}(0,1) \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The ionization rate profile (Eq. 2) gives the probability distribution of a new particle being created at a given axial location. We can sample this distribution by applying an inverse-CDF transform. The axial and azimuthal positions of a new electron-ion pair are thus given by [9]

$$x_i = x_m + \sin^{-1}(2r_1 - 1)\frac{x_2 - x_1}{\pi} \tag{6}$$

$$y_i = r_2 L_y, \tag{7}$$

where both r_1 and r_2 are samples from uniform distributions on the interval [0, 1]. The particle velocities are Maxwellian, drawn from 3-D normal distributions with zero mean and standard deviations equal to the thermal speed, $v_{t,j} = \sqrt{T_j/m_j}$ for $j \in \{i, e\}$. The weights of the newly-injected particles are equal to W_0. Once the particle positions, velocities, and weights have been generated, we use WarpX's `add_particles` function to add the newly-generated particles to their containers.

Cathode injection

To maintain current continuity, the benchmark prescribes that all net charge leaving the domain through the anode boundary is re-injected as electron current at the cathode. To do this, we make use of WarpX's `BoundaryScrapingDiagnostics` feature. This allocates a buffer into which particles that leave the domain through specified boundaries are logged. We record all particles that leave through the anode boundary at $x = 0$. Then, in a callback function placed in the `particleinjection` position, we count how many electrons and ions are in the buffer. We then inject a number of electrons equal to the difference between the number of ions and number of electrons in that buffer. Mathematically, the flux of electrons injected at the cathode each timestep, $\Gamma_{e,c}$, is given by

$$\Gamma_{e,c} = \Gamma_{e,a} - \Gamma_{i,a}, \tag{8}$$

where $\Gamma_{e,a}$ and $\Gamma_{i,a}$ are the numbers of electrons and ions, respectively, that had left the anode boundary that timestep and are now present in the boundary buffer. We inject the electrons at $x = x_e = 2.4$ cm, with a uniform distribution in the azimuthal dimension, with velocities sampled from a zero-mean full 3D

Maxwellian distribution with a temperature of 10 eV. The newly-injected cathode electrons have weight W_0. After the electrons have been added to the particle container, we clear the boundary buffer to avoid an increasing memory footprint as the simulation proceeds.

Potential adjustment
In the LANDMARK benchmark, the electrostatic potential is adjusted at every timestep so that the potential drop between the anode at $x = 0$ and the cathode at $x_e = 2.4$ cm is 200 V. To accomplish this adjustment, after the particles have been deposited on the grid, we first solve the fields with a Dirichlet boundary of 0 V at $x = L_x$, giving a potential $U(x, y)$, and then we average the potential at the cathode line, giving $\overline{U}_e = (\int_0^{L_y} U(x_e, y)dy)/L_y$. Next, we compute the corrected electrostatic potential at this timestep, $\phi(x, y)$, using[9]:

$$\phi(x, y) = U(x, y) - \frac{x}{x_e}\overline{U}_e. \tag{9}$$

To implement this procedure in WarpX, we create a callback, `adjust_potential`, in the `afterEsolve` position. This position ensures that the callback will be invoked only after Poisson's equation has been solved at that timestep. Next, we obtain both the axial electric field and the potential from WarpX. As the location of the cathode line does not directly correspond to a cell center, we interpolate the potential from the cells on either side of the cathode line to x_e. We can then average this interpolated potential over the azimuthal potential to get $\overline{U}_e$, and then apply the correction given by Eq. 9 to the potential. Finally, we differentiate Eq. 9 to get a correction to the axial electric field:

$$E_x = -\frac{\partial\phi}{\partial x} = -\frac{\partial}{\partial x}U(x, y) + \frac{\overline{U}_e}{x_e} = \mathcal{E}_x + \frac{\overline{U}_e}{x_e}. \tag{10}$$

Here, we have made use of the fact that the uncorrected axial electric field, $\mathcal{E}_x$, is simply equal to the derivative of U in the $x-$direction. Following a similar procedure for the azimuthal direction, it can be shown that $E_y = \mathcal{E}_y$, so the azimuthal electric field does not need to be corrected.

C. Simulation outputs

We configure WarpX to output grid-based diagnostics every 5000 iterations. These diagnostics include the electric field vector, electrostatic potential, and charge densities of all species. Additionally, due to the large number of particles in the domain, it proved infeasible to save information about the simulation macro-particles directly Instead, at every output step, we compute the moments of the each species' velocity distribution function at every cell. In particular, we calculate the zeroth (density), first (bulk velocity vector), second (pressure tensor) and contracted third (heat flux vector) moments. In addition to these outputs, we also save the number of particles at each output timestep and record the time taken by each simulation step. Lastly, we obtaie code profiling information via AMReX's TinyProfiler tool built into WarpX. This allows us to investigate the relative cost of the different parts of the PIC computational cycle.

D. Numerical 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. These are described below, along with the range of investigated parameters.

Particle shape function
As discussed in Section II.B.1, an important part of the particle-in-cell method is the selection of a suitable particle shape function. Typically, these shape functions are multi-dimensional B-splines.[16] When gathering grid quantities to the particles, or depositing charge and current onto the grid, the order of these shape functions determine how many nearby grid cells influence and are influenced by a given particle. Higher-order shape functions include more grid cells, and thus potentially reduce the sampling noise inherent to the PIC method,[16] at an increased computational cost compared to lower-order functions. Here, we consider linear, quadratic, and cubic shape functions.

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. This aids 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 which has the potential to slow down the simulation if sorting is called too frequently. To find the optimal setting, we varied 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 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.[17], 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_trigger_max_avg_ppc`, which defines the maximum number of particles per cell, averaged over the whole domain, above which resampling is triggered.

3. `resampling_min_ppc`, which is the threshold number of particles per cell below which a cell will not be resampled.

In our simulations, we left the first parameter at the WarpX default of 1.5 as we had no reason, a priori, to use a different value, and varied the other two. We vary the maximum average particle count per cell between 200 and 300 and the minimum particle count between 75 and 200.

Field solve tolerance
For electrostatic simulations, WarpX uses the *Multi-Level Multi-Grid* (MLMG) method to solve Poisson's equation to obtain the electric field. Starting from the fields solved at the previous timestep, this method iteratively reduces the error in the solution of Poisson's equation until a user-specified relative tolerance (hereafter, "multigrid precision") is reached. We test tolerances between 10^{-2} (i.e. 1% of the initial error) and 10^{-6} (0.0001% of the initial error) in this paper.

Coulomb collision interval
While the original benchmark does not include binary collisions, WarpX supports binary collisions using the *Direct Simulation Monte Carlo* (DSMC) method.[18] Considering a binary collision between species A and species B, this method works by first pairing up all available members of species A with others of species B, splitting particles when necessary to ensure that every particle of species A has a partner particle of species B. The collision probability for each pair is then computed, considering the relative velocities of the partners and the collision cross-section. Finally, if the collision probability is less than a uniform random number in the interval [0, 1], a collision occurs and the particle velocities are adjusted according the type of collision. For charged particle collisions, WarpX uses a relativistic version of this algorithm developed by Perez et al.[19]

In Hall thrusters, it is expected that "anomalous" scattering events will dominate classical ones across most of the discharge. Nevertheless, in some areas of the discharge, particularly near the location of the peak magnetic field, the anomalous collision frequency may drop below the classical value.[3] In other regions of the discharge, classical collisions may modify instabilities giving rise to anomalous transport by providing additional mechanisms for energy transport, and randomizing particle velocities. Additionally, in more advanced codes which include neutrals, both electron-neutral collisions and ionization collisions will require the use of a DSMC-like algorithm. To assess the cost of checking for collision events on WarpX's computational cycle, we therefore perform some additional simulations which incorporate a selection of classical collisions. As there are no neutrals in our simulations, we consider only electron-electron (e-e), electron-ion (e-i) and ion-ion (i-i) collisions. The relevance of these collisions to the physics of the present simulations can be determined by computing the effective electron-ion collision frequency for a plasma with density n_e and temperature T_e, which is given in SI units by[20]

$$\nu_{ei} = \frac{e^{\frac{5}{2}} n_e \ln \lambda_{ei}}{(4\pi\epsilon_0)^2 m_e^{1/2} T_e^{3/2}} \tag{11}$$

where $\ln \lambda_{ei} = 24 - \ln\left(10^{-3} n_e^{1/2} T_e^{-1}\right)$ is the Coulomb logarithm for $T_e > 10$ eV.[20] Picking $n_e \approx 2 \times 10^{17}$ m^3 and $T_e \approx 20$ eV as representative values, we find that $\lambda_{ei} \approx 10$ and the corresponding electron-ion collision frequency is $\nu_{ei} \approx 1.94 \times 10^4$ s^{-1}. This is a very low collision frequency (approximately four collisions per electron across the simulation duration), and it is therefore unlikely that Coulomb collisions will have a large impact on the simulation physics. In real thrusters, the classical collision frequency (considering electron-ion and and electron-neutral collisions) is on the order of $10^5 - 10^6$ s^{-1},[2] and will have a larger impact on the physics. Despite the low frequency of the included collisions, the majority of the computational cost of the DSMC algorithm involves checking for collision events,[18] so we include them despite their limited effect on the physics of the simulation. Given the expected low collision frequency relative to the simulation timestep, we super-cycle the collision computations, applying them only after several global timesteps have elapsed. We vary the number of global timesteps per collision check (the "collision interval") between 10 and 10,000. These intervals can resolve collision frequencies between 10^7 and 10^{10} s^{-1} and should therefore serve as an adequate test of the relative impact of collision checks on a full-scale Hall thruster simulation.

E. 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 Sec. II.A, while the bottom half shows the ranges of the variable numerical parameters discussed above. The parameters of the baseline case are underlined.

Table 1: Simulation parameters employed in this work.

Benchmark 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^{-2} T
Initial plasma density, n_0	5×10^{16} 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
Additional parameters	
Initial particles per cell	<u>75</u>, 150, 300
Particle shape function	<u>linear</u>, quadratic, cubic
Resampling: max particles per cell	<u>no resampling</u>, 200, 250, 300
Resampling: min particles per cell	<u>no resampling</u>, 75, 100, 150, 200
Particle sort interval	10, 50, 100, <u>500</u>, 1000
Multigrid precision	10^{-2}, 10^{-3}, <u>10^{-5}</u>, 10^{-6}
Coulomb collision interval	10, 1000, 10000, <u>no collisions</u>

We perform all but one of our simulations on a single Nvidia H100 GPU, with 80 GB of onboard memory.[21]

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 hails from the same era as the computational hardware used in the 2019 benchmark.

III. Results and discussion

A. Baseline simulation

In Table 2, we summarize the performance, in terms of wall time, of each of our simulations. 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 shape functions for the particles, a multigrid precision of 10^{-5}, and did not include collisions. In Figure 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, so our modifications to WarpX appear to have been successful.

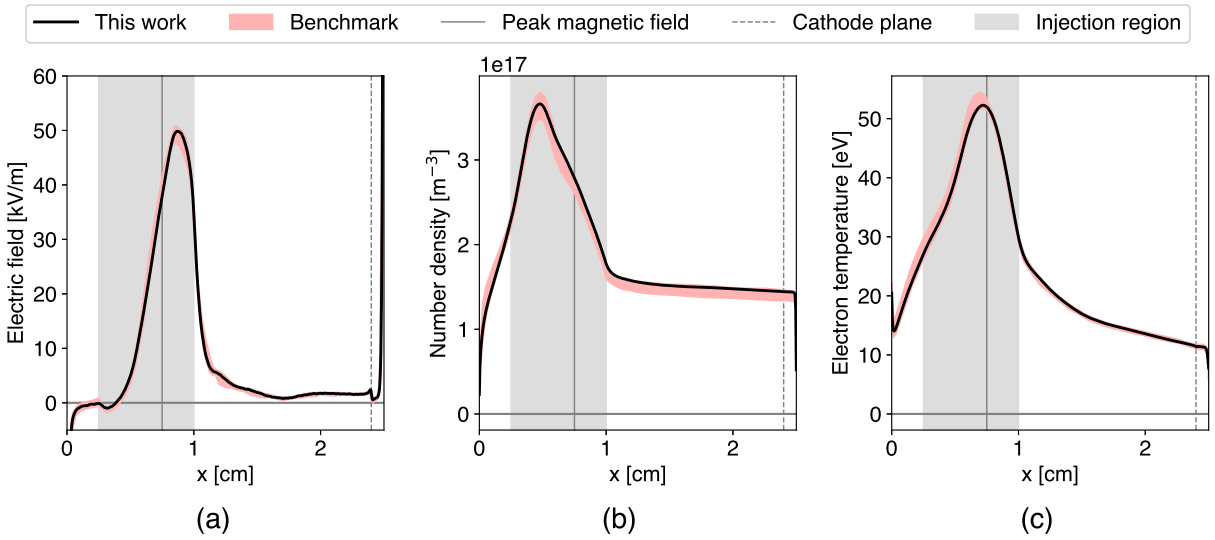

Figure 3: (a) Electric field, (b) ion number density, and (c) electron temperature for the baseline simulation. The range of benchmark results from Case 2 of Charoy et al, 2019[9] is indicated in pale red.

WarpX's completed the full 20 µs simulation duration 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 (2.5 days on 112 CPUs). The only GPU-based code which participated in the 2019 effort required 9 days to finish Case 2 on an Nvidia A100 GPU, a more powerful contemporary of the V100.[22,23] However, this code used an implicit particle pusher, which had a much larger computational overhead than the explicit scheme employed by our WarpX simulations. As such, it is not directly comparable to our results.

In Table 2, we compare the performance of our WarpX simulations on all three 2019 benchmark cases. We also report a summary of the time spent in by WarpX in different parts of the PIC cycle (see Figure 2) in the first two cases. Due to a cluster issue, the Case 3 simulation prematurely terminated at 19 microseconds, and as such did not output profiling information. We extrapolate the runtime of Case 3 to 20 microseconds when reporting it in able 2. Cases 1 and 3 — which have two and four times as many particles, respectively, as Case 2 — took 58% and 179% longer than the baseline case to complete. This nearly linear scaling in computational time indicates that WarpX is using close to all of its available resources when executing the baseline case (i.e., few parts of the GPU that WarpX can make use of are left idle). In the baseline case, the

Table 2: Performance for the baseline simulation (Case 2, highlighted in grey) as well as Cases 1 and 3, on both H100 and V100 GPUs. The performance of Case 3 is extrapolated from 19 microseconds due to a premature termination.

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 (PPPL)[9]	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%	N/A
Field solve	40.86 %	24.21 %	N/A
Charge deposition	8.43 %	10.72 %	N/A
Particle injection and field correction (Python)	1.9 %	1.28 %	N/A

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, the fraction of time WarpX spent in the particle routines increased by up to 50%.

B. Impact of higher-order shape functions

In Table 3, we reported that using higher-order shape functions for particles had a detrimental impact on performance. However, if using these shape functions reduces noise, it is feasible that the simulation could be run with fewer computational particles. To assess the degree of noise reduction, we show the electron temperature at the final simulation timestep ($t = 20\,\mu s$) when using for linear, quadratic, and cubic particle shape functions.

In each of the images, a small amount of noise or grain can be seen. This noise is inherent to the particle-in-cell method; as each computational particle stands for millions or billions of real particles, the observed distribution functions and moments represent coarse Monte Carlo approximations of those that would be obtained by using Eulerian kinetic approximations, or by increasing the number of particles.[24] However, qualitatively, it does not appear that using quadratic or cubic shape functions significantly reduced the noise in the simulation compared to using linear shape functions. This likely has to do with the fact that WarpX applies a low-pass filter to the charge density at each timestep. This helps remove some of the spurious high-frequency noise, but may limit the usefulness of higher-order shape functions.

C. Effect of resampling

Of the parameters tested, only the use of particle resampling had a significant effect on the benchmark results. In the remainder of cases, the results were largely indistinguishable (within the uncertainly of the range of benchmark simulations) from those of the baseline case presented in Figure 3.

Given the results in Table 2, the average particles per cell of the baseline simulation at steady state was only 290. It is thus unsurprising that resampling only when the average number of particles per cell exceeded 300 did not alter the results at steady state, as effectively no resampling was performed. The small increase in performance (a decrease in run-time from 1.81 to 1.76 days, about 1 hour and 12 minutes) seen in Table 3 when employing resampling at a threshold of 300 can be attributed to resampling particles in the startup phase, when the particle count transiently exceeded 300 particles per cell.

In contrast, resampling below 300 particles per cell per cell had a much more significant performance impact, decreasing run-time by up to 25%. However, the chosen resampling algorithm seemed to adversely affect the physical fidelity of the simulation. When resampling at these lower thresholds, the plasma density and electric field were very similar to those of the baseline case; however, the electron temperature at the

Table 3: Recorded wall times for each parameter set. The baseline case parameters are highlighted in grey.

Parameter value		Wall time
Initial particles per cell		
75		1.81 days
150		2.86 days
300		5.06 days
Particle shape function		
Linear		1.81 days
Quadratic		2.17 days
Cubic		2.59 days
Resampling thresholds		
Min particles	Max particles	
No resampling		1.81 days
75	300	1.76 days
200	275	1.66 days
150	275	1.66 days
100	275	1.66 days
75	275	1.66 days
200	250	1.58 days
150	250	1.55 days
100	250	1.54 days
75	250	1.54 days
150	200	1.40 days
100	200	1.40 days
75	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
Collision interval		
10		2.71 days
100		1.87 days
1000		1.79 days
10,000		1.80 days
No collisions		1.81 days
Optimized		**1.52 days**

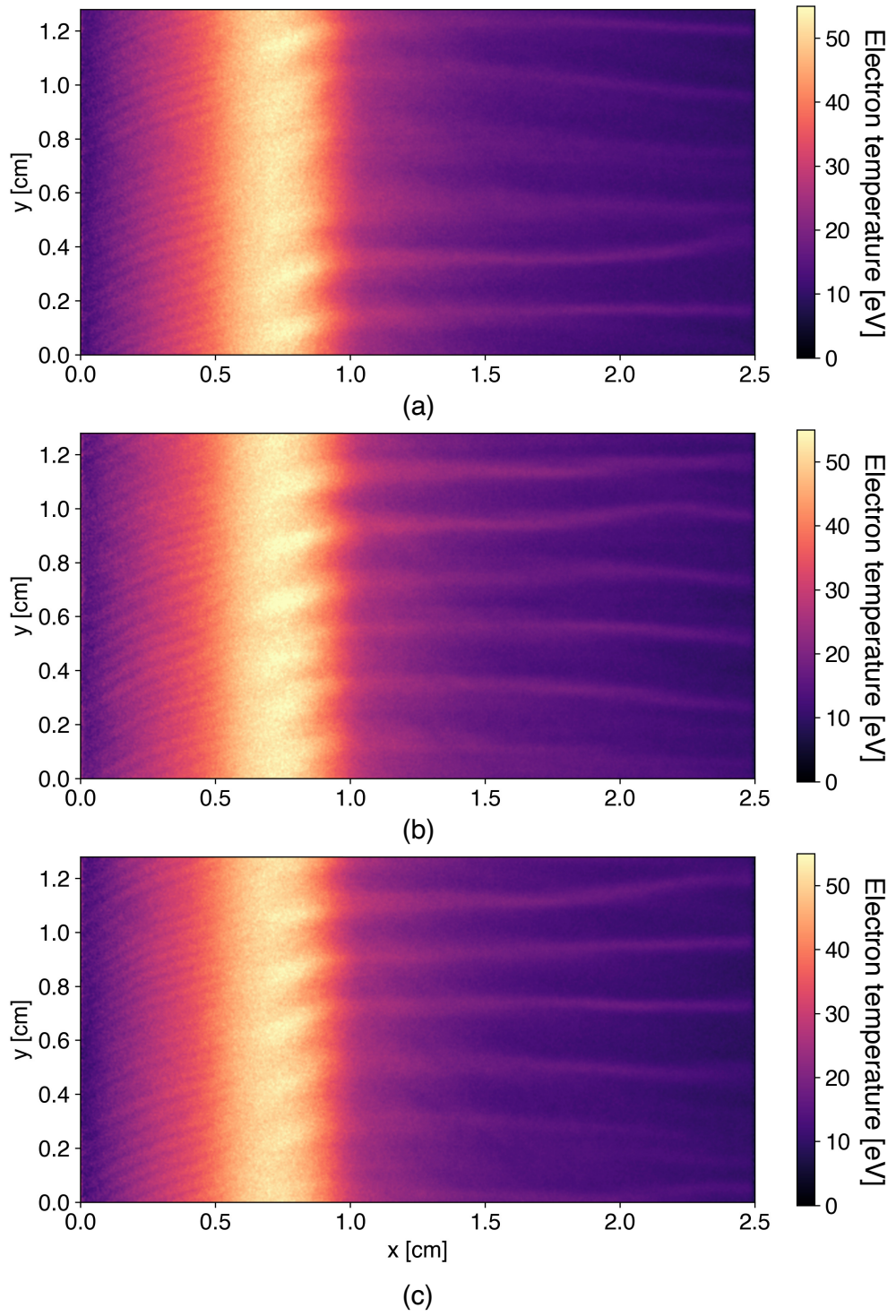

Figure 4: Electron temperature at $t = 20\,\mu s$ for (a) linear, (b) quadratic, and (c) cubic shape functions.

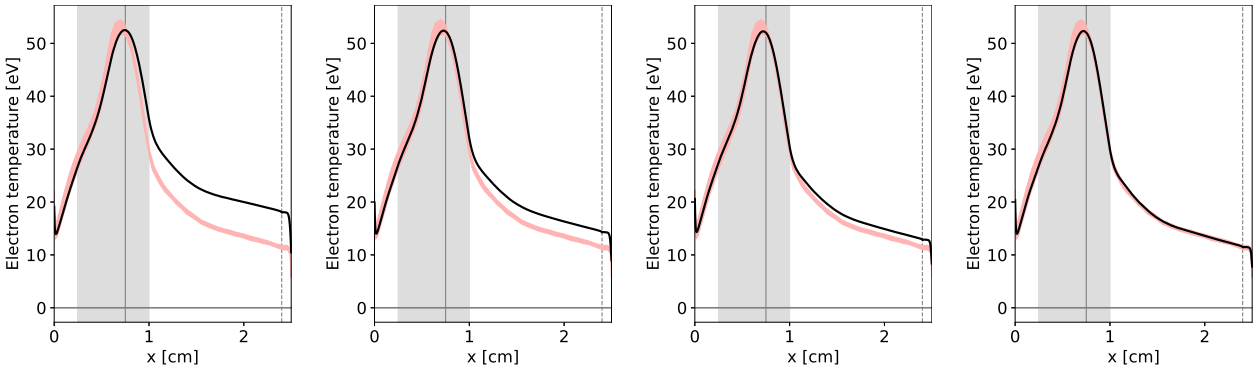

Figure 5: Effect of resampling on electron temperature profiles, with resampling thresholds of (a) 200 particles per cell, (b) 250 particles per cell, (c) 275 particles per cell, and (d) 300 particles per cell. In all cases, the resampling minimum parameter was set to 150 particles per cell. The grey-shaded area on the figures indicates the particle injection region. The pale red area represents the range of results obtained in the benchmark.[9]

right boundary varied significantly from the baseline case depending on the chosen resampling parameters. In Figure 5, we show how the simulated electron temperature changed as the maximum average particles per cell increased from 200 to 300 particles per cell. While the solution remained in good agreement with the benchmark result upstream of the location of maximum magnetic field, it begins to deviate in the downstream half of the domain. At the right boundary, the electron temperature at a maximum resampling threshold of 200 particles per cell is 7 eV higher than that of the baseline case.

To investigate why this might occur, we plot in Figure 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 regions of this mode, the particle density may exceed the threshold for resampling. This resampling leaves the density (and therefore electric field, via Poisson's equation) unchanged, but could result in a small spread of the velocity distribution functions, and therefore heating, if the particle resolution in these high-density regions is not sufficiently high. This could then produce the anomalously-high electron temperatures observed in the right half of the domain.

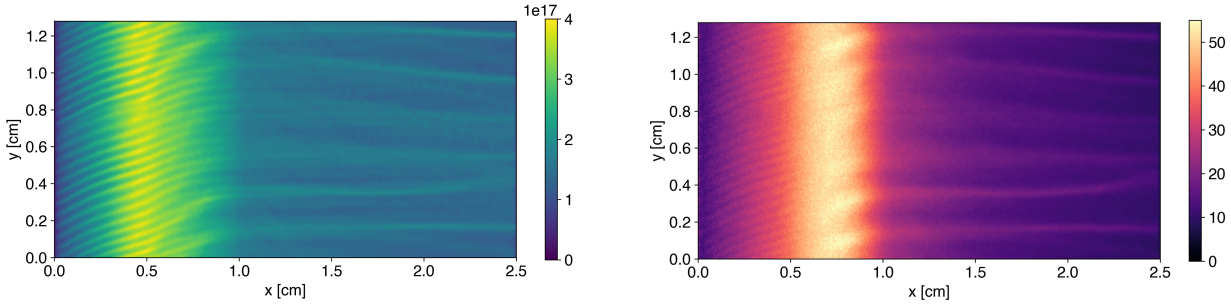

Figure 6: (a) Ion density (m^{-3}) and (b) electron temperature (eV) of the baseline simulation at the last timestep ($t = 20\,\mu s$).

Given these results, it seems that particle resampling should be used with caution in WarpX, 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.[25] 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. For the benchmark simulation, it seems that 300 particles per cell is an adequate threshold.

D. Other parameters

The results in Table 3 demonstrate that, on average, more frequent particle sorting results in better simulation performance. However, there does not appear to be a significant difference between sorting every 5 iterations and sorting every 100. Additionally, as expected, collision checks slow down the simulation when performed too frequency. Interestingly, however, checking for collisions every 1000 and 10,000 iterations seem to moderately improve performance (by about 48 minutes and 24 minutes, respectively) over the baseline case. To perform collisions in each grid cell, the DSMC method first sorts particles to determine which grid cell they occupy. This likely has similar performance effects to increasing the particle sorting interval.

Lastly, we found 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 was already sufficient 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, within the noise threshold of the PIC method and the run-to-run variance.

E. Optimized simulation

Taking all of the performance results from Table 3 together, it is clear that the resampling threshold, particle sorting interval, and the field solve tolerance had the largest performance implications of the investigated parameters. We performed one final "optimized" simulation, resampling at particle counts above 300 particles/cell, sorting every 50 iterations, and using a field solve tolerance of 10^{-3}. This simulation completed in 1.52 days, or 36.5 hours, with results indistinguishable from those of the baseline simulation.

F. 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 this work are due to WarpX's code architecture. 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[21]. In particular, while the H100 PCI-e GPU has a capacity of 26 and 51 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.

We note that all of our simulations were performed on a single GPU. When scaling to 3D simulations, the ability to use multiple GPUs in parallel is critical. We believe that WarpX's demonstrated GPU scaling capabilities[12] make it an excellent target for these much larger thruster simulations. Lastly, while we did not employ adaptive mesh refinement in this study, the AMR capabilities built into WarpX have the potential to further accelerate Hall thruster simulations by resolving regions of the discharge that need it without wasting resources on those that don't.

G. 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 ionization introduces breathing mode oscillations which have frequencies on the order of 10 kHz.[1] As such, simulations that capture these oscillations must be run for timescales of ~ 1 ms, about 50 times longer than the simulation durations employed in this work. Additionally, recent 3D particle-in-cell simulations of Hall thrusters have demonstrated that many important aspects of the instabilities governing anomalous transport are not well-resolved by 2D axial azimuthal simulations.[5] Even accounting for significant improvements in hardware, these constraints likely means kinetic, whole-device Hall thruster simulations will require wall times measuring in the months, if not years. 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 useable in engineering applications.

IV. 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 performed the well-known 2-D axial azimuthal benchmark simulation 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 un-physical 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 remaining physical parameters, including the particle shape function, the precision of the multigrid Poisson solver, and the particle sorting interval, 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 also found that the impact of the particle shape function on performance was large. Using a cubic shape function for the particles, our simulation took 40% longer than when using a linear shape function. However, using higher-order shape functions did not reduce the noise in the computed particle moments.

Lastly, we demonstrated the vast improvement in recent GPUs 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.

Using these findings, we performed a final simulation using the best parameters from each of our numerical investigations. This "optimized" simulation completed in just 36 hours on a single Nvidia H100 GPU, significantly faster than any published result to date and using far fewer 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.

V. 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.

VI. Code and data availability

WarpX is an open-source code, available on Github at https://github.com/ECP-WarpX/WarpX. Our simulations were performed on version 24.05. The scripts used to perform simulations in this work are available on Github at https://github.com/archermarx/warpx-hall. The data generated for the baseline benchmark case, as well as the code used to analyze them, are available online at thomasmarks.space/content/iepc-2024.

References

[1] Jean Pierre Boeuf. Tutorial: Physics and modeling of hall thrusters. *Journal of Applied Physics*, 121, 1 2017. ISSN 10897550. doi: 10.1063/1.4972269.

[2] Ioannis G. Mikellides and Alejandro Lopez Ortega. 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 Science and Technology*, 28:48, 1 2019. ISSN 13616595. doi: 10.1088/1361-6595/aae63b.

[3] Thomas A. Marks and Benjamin A. Jorns. Challenges with the self-consistent implementation of closure models for anomalous electron transport in fluid simulations of Hall thrusters. *Plasma Sources Sci. Technol.*, 32(4):045016, April 2023. ISSN 0963-0252. doi: 10.1088/1361-6595/accd18.

[4] Thomas A. Marks and Benjamin A. Jorns. Evaluation of algebraic models of anomalous transport in a multi-fluid Hall thruster code. *Journal of Applied Physics*, 134(15):153301, 10 2023. ISSN 0021-8979. doi: 10.1063/5.0171824.

[5] W. Villafana, B. Cuenot, and O. Vermorel. 3d particle-in-cell study of the electron drift instability in a hall thruster using unstructured grids. *Physics of Plasmas*, 30, 3 2023. ISSN 10897674. doi: 10.1063/5.0133963.

[6] Rob Farber. Wdmapp – the first simulation software in fusion history to couple tokamak core to edge physics. *Exascale Computing Project*, Jul 2022. URL `https://www.exascaleproject.org/highlight/wdmapp-the-first-simulation-software-in-fusion-history-to-couple-tokamak-core-to-edge-physics/`.

[7] Luca Fedeli, Axel Huebl, France Boillod-Cerneux, Thomas Clark, Kevin Gott, Conrad Hillairet, Stephan Jaure, Adrien Leblanc, Rémi Lehe, Andrew Myers, Christelle Piechurski, Mitsuhisa Sato, Neïl Zaim, Weiqun Zhang, Jean-Luc Vay, and Henri Vincenti. 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*, pages 1–12, 2022. doi: 10.1109/SC41404.2022.00008.

[8] J.-L. Vay, A. Almgren, J. Bell, L. Ge, D.P. Grote, M. Hogan, O. Kononenko, R. Lehe, A. Myers, C. Ng, J. Park, R. Ryne, O. Shapoval, M. Thévenet, and W. Zhang. Warp-x: A new exascale computing platform for beam–plasma simulations. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 909:476–479, 2018. ISSN 0168-9002. doi: 10.1016/j.nima.2018.01.035. 3rd European Advanced Accelerator Concepts workshop (EAAC2017).

[9] T Charoy, J P Boeuf, A Bourdon, J A Carlsson, P Chabert, B Cuenot, D Eremin, L Garrigues, K Hara, I D Kaganovich, A T Powis, A Smolyakov, D Sydorenko, A Tavant, O Vermorel, and W Villafana. 2d axial-azimuthal particle-in-cell benchmark for low-temperature partially magnetized plasmas. *Plasma Sources Science and Technology*, 28(10):105010, oct 2019. doi: 10.1088/1361-6595/ab46c5.

[10] JP Boeuf et al. Landmark plasma test cases, 2019. URL `https://jpb911.wixsite.com/landmark/test-cases`. Accessed: 2024-06-07.

[11] Luca Fedeli, Axel Huebl, France Boillod-Cerneux, Thomas Clark, Kevin Gott, Conrad Hillairet, Stephan Jaure, Adrien Leblanc, Rémi Lehe, Andrew Myers, Christelle Piechurski, Mitsuhisa Sato, Neïl Zaim, Weiqun Zhang, Jean-Luc Vay, and Henri Vincenti. 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*, pages 1–12, 2022. doi: 10.1109/SC41404.2022.00008.

[12] A. Myers, A. Almgren, L.D. Amorim, J. Bell, L. Fedeli, L. Ge, K. Gott, D.P. Grote, M. Hogan, A. Huebl, R. Jambunathan, R. Lehe, C. Ng, M. Rowan, O. Shapoval, M. Thévenet, J.-L. Vay, H. Vincenti, E. Yang, N. Zaïm, W. Zhang, Y. Zhao, and E. Zoni. Porting warpx to gpu-accelerated platforms. *Parallel Computing*, 108:102833, 2021. ISSN 0167-8191. doi: 10.1016/j.parco.2021.102833.

[13] Weiqun Zhang, Ann Almgren, Vince Beckner, John Bell, Johannes Blaschke, Cy Chan, Marcus Day, Brian Friesen, Kevin Gott, Daniel Graves, Max Katz, Andrew Myers, Tan Nguyen, Andrew Nonaka, Michele Rosso, Samuel Williams, and Michael Zingale. AMReX: a framework for block-structured adaptive mesh refinement. *Journal of Open Source Software*, 4(37):1370, May 2019. doi: 10.21105/joss.01370.

[14] A. Y. Aydemir. A unified Monte Carlo interpretation of particle simulations and applications to non-neutral plasmas. *Physics of Plasmas*, 1(4):822–831, 04 1994. ISSN 1070-664X. doi: 10.1063/1.870740.

[15] C.K. Birdsall. Particle-in-cell charged-particle simulations, plus monte carlo collisions with neutral atoms, pic-mcc. *IEEE Transactions on Plasma Science*, 19(2):65–85, 1991. doi: 10.1109/27.106800.

[16] Dominic A.S. Brown, Matthew T. Bettencourt, Steven A. Wright, Satheesh Maheswaran, John P. Jones, and Stephen A. Jarvis. Higher-order particle representation for particle-in-cell simulations. *Journal of Computational Physics*, 435:110255, 2021. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2021.110255.

[17] A. Muraviev, A. Bashinov, E. Efimenko, V. Volokitin, I. Meyerov, and A. Gonoskov. Strategies for particle resampling in pic simulations. *Computer Physics Communications*, 262:107826, 2021. ISSN 0010-4655. doi: 10.1016/j.cpc.2021.107826.

[18] G. A. Bird. Approach to Translational Equilibrium in a Rigid Sphere Gas. *The Physics of Fluids*, 6(10): 1518–1519, 10 1963. ISSN 0031-9171. doi: 10.1063/1.1710976.

[19] F. Pérez, L. Gremillet, A. Decoster, M. Drouin, and E. Lefebvre. Improved modeling of relativistic collisions and collisional ionization in particle-in-cell codes. *Physics of Plasmas*, 19(8):083104, 08 2012. ISSN 1070-664X. doi: 10.1063/1.4742167.

[20] J. D. Huba. *NRL Plasma Formulary*. Revised 2023. Washington, DC : Naval Research Laboratory, 1950-. URL `https://www.nrl.navy.mil/News-Media/Publications/NRL-Plasma-Formulary/`.

[21] *NVIDIA H100 Tensor Core GPU Datasheet*. Nvidia Corporation, 2024. URL `https://resources.nvidia.com/en-us-tensor-core/nvidia-tensor-core-gpu-datasheet`. Accessed: 2024-06-18.

[22] *NVIDIA V100 Tensor Core GPU Datasheet*. Nvidia Corporation, 2020. URL `https://images.nvidia.com/content/technologies/volta/pdf/volta-v100-datasheet-update-us-1165301-r5.pdf`. Accessed: 2024-06-18.

[23] *NVIDIA A100 Tensor Core GPU Datasheet*. Nvidia Corporation, 2024. URL `https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-nvidia-us-2188504-web.pdf`. Accessed: 2024-06-18.

[24] Alexey V. Arefiev and Boris N. Breizman. Magnetohydrodynamic scenario of plasma detachment in a magnetic nozzle. *Physics of Plasmas*, 2005. ISSN 1070664X. doi: 10.1063/1.1875632.

[25] D. Faghihi, V. Carey, C. Michoski, R. Hager, S. Janhunen, C.S. Chang, and R.D. Moser. Moment preserving constrained resampling with applications to particle-in-cell methods. *Journal of Computational Physics*, 409:109317, 2020. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2020.109317.