

© 2024 Logan Tate Meredith

ADVANCING HYBRID FLUID-KINETIC METHODS FOR PLASMA SIMULATION

BY

LOGAN TATE MEREDITH

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Physics
in the Graduate College of the
University of Illinois Urbana-Champaign, 2024

Urbana, Illinois

Doctoral Committee:

Professor Charles Forbes Gammie, Chair
Associate Professor Davide Curreli, Director of Research
Professor Rizwan Uddin
Dr. Richard Michael Jack Kramer, Sandia National Laboratories

Abstract

The complex, multiscale nature of plasmas necessitated the early adoption of computers by plasma physicists to model their behavior more than half a century ago. Their limited computational resources demanded fast, efficient algorithms for simulating plasmas. In general, these algorithms fall into two categories: kinetic and fluid. Kinetic algorithms retain much detailed information about how velocities are distributed in particles of the plasma, and as a consequence, they are very widely applicable but computationally expensive. In contrast, fluid algorithms discard much of this information to reduce computational cost, but must make assumptions about the plasma to compensate. Such algorithms have kept pace with developments in computing technology and are still in wide use today, owing to their inherent parallelizability.

However, as high-performance computing moves into the exascale era, plasma physicists have become inspired to attempt increasingly large simulations, as evidenced by the trend toward whole-device modeling in the study of magnetic confinement fusion. The widening of the computational bottleneck has exposed flaws in the traditional paradigm of using either purely kinetic or purely fluid algorithms. It is now clear that a hybrid approach, combining the strengths of both kinetic and fluid methods, is necessary to continue efficiently modeling plasmas at the largest desired scales. Although the study of hybrid techniques is not new, the problems associated with kinetic or fluid simulations of large systems all but necessitates their accelerated development.

To that end, this work is broadly concerned with the construction and implementation of hybrid methods for plasma simulation on modern high-performance computing systems. A framework for the hybridization of kinetic and fluid methods is presented. This framework was then developed into the hPIC2 plasma simulation code, which overcomes the difficulty of targeting many computing architectures through the use of the Kokkos performance portability library. hPIC2 incorporates a number of innovations deemed indispensable for large-scale plasma simulation, including a novel fault tolerance scheme and advanced finite element methods furnished by the MFEM library. This work also explores conversions between fluid and kinetic descriptions. A method was developed that identifies when a kinetically modeled portion of a plasma satisfies the fluid assumptions and dynamically converts it into a fluid, which is of particular utility in highly collisional hybrid plasma simulations.

The algorithms and code presented here permit the simultaneous use of fluid and kinetic descriptions for the plasma in a single simulation and adaptive conversion between them. They have been designed for and tested on architectures composing the largest supercomputers currently available. The result of this work is a significant step toward the realization of plasma simulations spanning many scales. This is particularly important for pulsed-power physics and magnetic confinement fusion, whose devices are large and complex and stress the limits of the traditional, non-hybrid approach.

Acknowledgments

This work would not have been possible without the love and support of my family. My parents, Jenn and Doug, always encouraged me to pursue a career in science, and I will be forever thankful. You fostered curiosity and a love of learning that will never die in me. My breaks back at home in Colorado, long hikes in the Rockies, longer soaks in the hot tub, and the occasional brewery crawl all worked to restore me after long stints in Illinois. My sisters, Shay, Rachel, and Sarah, helped me to become the person I am today. Trips to see a car show and Rip and Bo in Laporte, or to work out and terrify Beef across the street, were welcome respites from the lab. Just kick it!

I was lucky to have a close-knit group of friends during my time in Illinois. The qual-fueled bonfires, sea shanties on Lake Shelbyville, poker nights, and basement shows set a memorable and energizing backdrop to my years of study. My partner, Meirah, was a supporting and calming presence in the most stressful times of my graduate studies. I am not sure why you waited six months for me while I was in New Mexico, but I am supremely thankful that you did.

I am extremely grateful to my advisor, Davide Curreli, for his guidance and mentorship during my graduate studies. Our impromptu brainstorming sessions on the blackboard inspired me to take risks in science, most of which paid off in the end. Your leadership was a vital component in the success of this work, and I hope to emulate it during my career. The Laboratory of Computational Plasma Physics has also been an extremely helpful resource for collaboration, discussion, and friendship. Special thanks go to Huq, for enthusiastically adopting hPIC2 as your primary code, even when it was beset by bugs; Mohammad, for helping with early verification and bringing hPIC2 into the new millennium with machine learning; Jon, for continuing maintenance on even the obscure components of RustBCA and always responding so quickly to my inane questions; and Mikhail, for implementing many key features in hPIC2, writing the RF demonstration in the methods paper, and aiding me in setting out the low-level design of the code.

My collaborators were an indispensable fount of knowledge. Richard Kramer, Duncan McGregor, Chris Moore, Billy McDoniel, Chris Siefert, and the EMPIRE development team at Sandia National Laboratories were instrumental in honing my skills as a physicist, computer scientist, mathematician, and researcher in general. Being so close to the development of such a large-scale codebase as EMPIRE gave me the courage to tackle the development of hPIC2. I also thank J. Andrej, M. L. Stowell, W. Pazner, T. Kolev, and the MFEM development team for helpful discussions on the inclusion of MFEM in hPIC2.

This material is based upon work supported by the U.S. Department of Energy Office of Science, Office of Fusion Energy Sciences and Office of Advanced Scientific Computing Research through the Scientific Discovery through Advanced Computing (SciDAC) project on Plasma-Surface Interactions under Award No. DE-SC0018141; and by the U.S. Department of Energy NNSA LRGF under grant number DE-NA0003960. This research used the Delta advanced computing and data resource which is supported by the National Science Foundation (award OAC 2005572) and the State of Illinois. Delta is a joint effort of the University of

Illinois Urbana-Champaign and its National Center for Supercomputing Applications.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Table of contents

Chapter 1	Introduction	1
1.1	The Boltzmann Equation and Its Moments	1
1.2	Kinetic vs. Fluid Models	5
1.3	Thesis Goal	6
Chapter 2	Constructing a Hybrid Fluid-Kinetic Numerical Scheme	7
2.1	The Hybrid-by-Species Paradigm	7
2.2	The Kinetic Component: Particle-in-Cell Method	8
2.3	The Fluid Component: Discontinuous Galerkin Methods	15
2.4	Hybrid Time Integration	21
Chapter 3	hPIC2: A Hardware-Accelerated, Hybrid Particle-in-Cell Code for Dynamic Plasma-Material Interactions	24
3.1	Implementation Details	25
3.2	Verification	31
3.3	Performance	41
3.4	Demonstrations	51
Chapter 4	Merging Particles into a Fluid in Hybrid Fluid-PIC Plasma Simulations	57
4.1	Description of Methods	58
4.2	Implementation Details	64
4.3	Results	65
Chapter 5	Conclusions	76
5.1	Future Work	76
Appendix A	Numerical Methods for Poisson's Equation	78
A.1	Finite Difference Methods	78
A.2	Finite Element Methods	80
Appendix B	Linearization of Combined Moments for Hybrid Merge	82
References		84

Chapter 1

Introduction

First named by Langmuir [1], plasma is a state of matter in which charged particles govern dynamics. Whereas short-range interactions dominate between particles in neutral solids, liquids, and gases, the presence of significant charge in plasmas imbues them with unique collective phenomena driven by electromagnetic forces. By far, the majority of matter in the universe is plasma [2], [3], and plasmas see wide use in industrial, scientific, and technological applications, such as health [4], polymer processing [5], television production [6], and semiconductor fabrication [7]. The myriad uses of plasmas have spurred the development of a wide variety of models to describe their behavior; indeed, plasmas were the targets of some of the earliest computer simulations [8].

Some of the most extreme anthropogenic plasmas are found in pulsed-power devices [9] and magnetic confinement fusion (MCF) experiments [10]. Pulsed-power devices store immense amounts of energy and discharge it into small loads over a short period of time, which typically spawns a plasma. In particular, a Z-pinch system generates magnetic fields that can compress the load to sufficiently high pressures to induce nuclear fusion. Such devices are often used to probe material properties in extreme conditions [11], [12]. MCF experiments, on the other hand, use magnetic fields to confine a plasma at high temperatures and pressures in order to sustain continuous nuclear fusion reactions.

In these applications, the target plasma is under such extreme conditions that minute changes to experimental parameters lead to significant changes in plasma behavior and performance. For example, the magnetic fields in MCF must be precisely controlled, lest the plasma escape confinement and damage the machine. Hence, the ability to predict plasma behavior to some degree of accuracy is of paramount importance. However, the very characteristic of plasmas that makes them so useful—their collective behavior—also makes them difficult to model. This issue is exacerbated by the complex plasma conditions intrinsic to many applications.

1.1 The Boltzmann Equation and Its Moments

A nonrelativistic plasma species is generally well described by the single-particle distribution function, $f = f(\mathbf{x}, \mathbf{v}, t)$, sometimes referred to simply as the distribution. The distribution is a quasiprobability distribution; the quantity $f(\mathbf{x}, \mathbf{v}, t) d\mathbf{x} d\mathbf{v}$ is the number of particles located within a neighborhood of the position $\mathbf{x}$ with velocity in a neighborhood of $\mathbf{v}$ at time t , assuming that particles of the species are pointlike. In other words, the distribution can be interpreted as a time-dependent density in phase space. More generally,

the distribution's arguments can range over internal degrees of freedom, such as vibrational modes, but we are primarily concerned with translational degrees of freedom.

The time evolution of the distribution is governed by the Boltzmann equation [13],

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{x}} + \frac{\mathbf{F}}{m} \cdot \frac{\partial f}{\partial \mathbf{v}} = \mathcal{C}[f], \quad (1.1)$$

where $\mathbf{F} = \mathbf{F}(\mathbf{x}, \mathbf{v}, t)$ is a macroscopic force field, m is the mass of particles of the species, and $\mathcal{C}$ is a functional that encodes collisions and interactions. We ignore the effects of gravity, so the only macroscopic force is the Lorentz force

$$\mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}), \quad (1.2)$$

where q is the charge of particles of the species, $\mathbf{E} = \mathbf{E}(\mathbf{x}, t)$ is the electric field, and $\mathbf{B} = \mathbf{B}(\mathbf{x}, t)$ is the magnetic field. Hence, the Boltzmann equation becomes

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{x}} + \frac{q}{m}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = \mathcal{C}[f]. \quad (1.3)$$

The time evolution of the electromagnetic fields is governed by Maxwell's equations,

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0}, \quad (1.4a)$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \quad (1.4b)$$

$$\nabla \cdot \mathbf{B} = 0, \quad (1.4c)$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t}, \quad (1.4d)$$

where $\rho = \rho(\mathbf{x}, t)$ is the charge density, ε_0 is the vacuum permittivity, μ_0 is the vacuum permeability, and $\mathbf{J} = \mathbf{J}(\mathbf{x}, t)$ is the current density. In the case where the magnetic field varies slowly compared to plasma evolution timescales, the electric field is approximately irrotational and can therefore be represented as the gradient of a scalar potential. Traditionally, the electric field is written as

$$\mathbf{E} = -\nabla \phi, \quad (1.5)$$

where $\phi = \phi(\mathbf{x}, t)$ is the electric potential. This assumption, the electrostatic approximation, yields Poisson's equation for the potential

$$\nabla^2 \phi = -\frac{\rho}{\varepsilon_0}. \quad (1.6)$$

The electrostatic approximation is valid when currents are small, which generally occurs when particles are nonrelativistic.

Note that the charge density can be computed from every species' distribution as

$$\rho = \sum_s \int_{\mathbb{R}^3} q_s f_s \, d\mathbf{v}, \quad (1.7)$$

where the subscript s ranges over species in the system. Hereafter, we consider a single species at a time, and so the subscript s is omitted.

Together, (1.3) and (1.6) form a complete description of the time evolution of a plasma under the listed

assumptions. In many cases, however, only a few macroscopic quantities are of interest, such as the number density or energy density. In these cases, the detailed velocity-space information afforded by the Boltzmann equation is unnecessary, and the additional computational expense of solving it is unwarranted. Instead, we can attempt to model the time evolution only of the desired macroscopic quantities.

To begin, consider that many important physical quantities can be computed as moments of the distribution in velocity space. The number density $n = n(\mathbf{x}, t)$ can be computed as

$$n = \int_{\mathbb{R}^3} f \, d\mathbf{v}; \quad (1.8)$$

the momentum density $nm\mathbf{u} = nm\mathbf{u}(\mathbf{x}, t)$ is

$$nm\mathbf{u} = \int_{\mathbb{R}^3} m\mathbf{v} f \, d\mathbf{v}; \quad (1.9)$$

the stress tensor $\mathbf{P} = \mathbf{P}(\mathbf{x}, t)$ is

$$\mathbf{P} = \int_{\mathbb{R}^3} m\mathbf{v} \otimes \mathbf{v} f \, d\mathbf{v}; \quad (1.10)$$

and the energy flux density $\mathbf{Q} = \mathbf{Q}(\mathbf{x}, t)$ is

$$\mathbf{Q} = \int_{\mathbb{R}^3} \frac{1}{2} m v^2 \mathbf{v} f \, d\mathbf{v}. \quad (1.11)$$

It is also useful to identify some moments in the reference frame of the moving species. With $\mathbf{w} = \mathbf{v} - \mathbf{u}$, let

$$\mathbf{p} = \int_{\mathbb{R}^3} m\mathbf{w} \otimes \mathbf{w} f \, d\mathbf{v} \quad (1.12)$$

be the pressure tensor, and let

$$\mathbf{q} = \int_{\mathbb{R}^3} \frac{1}{2} m w^2 \mathbf{w} f \, d\mathbf{v} \quad (1.13)$$

be the heat flux density. For convenience, let $p = \text{Tr}(\mathbf{p})/3$ be the scalar pressure and decompose the pressure tensor as

$$\mathbf{p} = p\mathbf{I} + \boldsymbol{\pi}, \quad (1.14)$$

where $\boldsymbol{\pi}$ is the generalized viscosity tensor. Finally, the total energy density is given by

$$nmE = \int_{\mathbb{R}^3} \frac{1}{2} m v^2 f \, d\mathbf{v}. \quad (1.15)$$

In order to model the time evolution of these quantities, we may compute the corresponding moment of (1.3) as

$$\int_{\mathbb{R}^3} \psi \left[\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{x}} + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} \right] d\mathbf{v} = \int_{\mathbb{R}^3} \psi \mathcal{C}[f] \, d\mathbf{v}, \quad (1.16)$$

where ψ is a polynomial of $\mathbf{v}$. A set of several such moment equations contributes to a fluid model for the plasma. However, in general, the moment equation for a given moment depends on higher-order moments of the distribution, leading to an infinite hierarchy of moment equations. In order to complete the fluid model, this hierarchy must be closed. Both asymptotic closure schemes, which perturb an equilibrium distribution by some small parameter, and truncation closure schemes, which use an equation of state (EOS) to related higher-order moments to the moments of interest, are common [14].

As an example, consider the Euler equations. These can be derived by taking $\psi = m$, $m\mathbf{v}$, and $\frac{1}{2}mv^2$ in (1.16), which yields

$$\frac{\partial(nm)}{\partial t} + \nabla \cdot (nm\mathbf{u}) = \int_{\mathbb{R}^3} m\mathcal{C}[f] d\mathbf{v}, \quad (1.17a)$$

$$\frac{\partial(nm\mathbf{u})}{\partial t} + \nabla \cdot \mathbf{P} = nq(\mathbf{E} + \mathbf{u} \times \mathbf{B}) + \int_{\mathbb{R}^3} m\mathbf{v}\mathcal{C}[f] d\mathbf{v}, \quad (1.17b)$$

$$\frac{\partial(nmE)}{\partial t} + \nabla \cdot (nmE\mathbf{u} + \mathbf{q} + \mathbf{p}\mathbf{u}) = nq\mathbf{E} \cdot \mathbf{u} + \int_{\mathbb{R}^3} \frac{1}{2}mv^2\mathcal{C}[f] d\mathbf{v}. \quad (1.17c)$$

Note that (1.17c) depends on the heat flux density, which is a higher-order moment. In order to close these equations, we assume that the heat flux density and generalized viscosity tensor are zero, yielding

$$\frac{\partial(nm)}{\partial t} + \nabla \cdot (nm\mathbf{u}) = \int_{\mathbb{R}^3} m\mathcal{C}[f] d\mathbf{v}, \quad (1.18a)$$

$$\frac{\partial(nm\mathbf{u})}{\partial t} + \nabla \cdot (nm\mathbf{u} \otimes \mathbf{u} + p\mathbf{I}) = nq(\mathbf{E} + \mathbf{u} \times \mathbf{B}) + \int_{\mathbb{R}^3} m\mathbf{v}\mathcal{C}[f] d\mathbf{v}, \quad (1.18b)$$

$$\frac{\partial(nmE)}{\partial t} + \nabla \cdot (nmE\mathbf{u} + p\mathbf{u}) = nq\mathbf{E} \cdot \mathbf{u} + \int_{\mathbb{R}^3} \frac{1}{2}mv^2\mathcal{C}[f] d\mathbf{v}. \quad (1.18c)$$

In addition, an EOS must be used to relate the scalar pressure to the number density, momentum density, and total energy density. A common analytic EOS for the Euler equations is the ideal gas law,

$$p = nkT, \quad (1.19)$$

where k is the Boltzmann constant and T is the temperature. When combined with the equipartition theorem for calorically perfect gases,

$$nmE = \frac{1}{2}nm\mathbf{u}^2 + \frac{1}{\gamma - 1}nkT, \quad (1.20)$$

where γ is the adiabatic index, we have

$$p = (\gamma - 1) \left(nmE - \frac{1}{2}nm\mathbf{u}^2 \right). \quad (1.21)$$

Although many sets of fluid equations and equations of state are used in the literature, it is nearly universally true that the closer a plasma distribution is to a drifting Maxwellian, the better it is described by fluid equations. Collisions and other interactions tend to drive plasmas toward a drifting Maxwellian distribution, and this process is called thermalization. Fluid descriptions are therefore best reserved for thermalized, highly collisional plasmas. Collisionality can be quantified by the Knudsen number

$$\text{Kn} = \frac{\lambda}{L}, \quad (1.22)$$

where λ is the mean free path (the average distance a particle travels between collisions) and L is a characteristic length scale. A fluid description is generally valid when $\text{Kn} \ll 1$ [15].

In general, fluid equations so derived can be written in conservation-law form

$$\frac{\partial y}{\partial t} + \nabla \cdot \mathbf{F} = G, \quad (1.23)$$

where y is some fluid moment, $\mathbf{F} = \mathbf{F}(y)$ is known as a flux functional, and the source G may depend on space, time, the fluid state, and the electromagnetic fields.

1.2 Kinetic vs. Fluid Models

Simply put, kinetic models directly solve (1.3), whereas fluid models solve a set of moment equations of the form (1.16) paired with some closure scheme. Though fluid models incur additional complexity because they require the solution of multiple coupled equations rather than the single Boltzmann equation, this complexity will turn out to be more than offset by the reduction in dimensionality associated with taking moments. This is an example of the curse of dimensionality, where the computational cost of solving a problem grows drastically with the number of dimensions in the problem. A result of this curse is that numerical methods for fluid models tend to perform better than their kinetic counterparts. The cost of this performance is the set of assumptions required to close the moment equation hierarchy for a particular fluid model, which restricts the range of phenomena that can be modeled using fluids. Furthermore, a fluid can unintentionally be driven into a regime where its closure is invalid, silently rendering its results meaningless.

A number of numerical schemes have emerged for kinetic models. Continuum Boltzmann solvers form a family of schemes that partition both configuration space and velocity space into a grid or mesh and solve for the distribution on this mesh. BOLSIG+ is a popular code using this method [16], as is Gkeyll, which notably uses a unique discontinuous Galerkin formulation for the distribution [17], [18]. However, particle methods, which represent the distribution using a finite set of particles that move throughout the domain, are far more common. Particle-in-cell schemes represent the vast majority of particle methods for kinetic models [8], [19]–[21]. By assuming that charged particles gyrate about magnetic field lines, gyrokinetic schemes have also been developed and implemented with great success in codes such as GTC-P [22] and XGC [23]. Algorithms based on quantum descriptions of kinetic models have also been developed, but have thus far not seen wide use [24]–[26].

In contrast to schemes targeting kinetic models, the fluid literature is dominated by continuum methods. Finite difference methods were once the methods of choice [27], but finite volume methods have since supplanted them [28], and now form the core of most commercial fluid codes. Notably, a number of time integration schemes based on operator splitting have been developed for various fluid models, including the SIMPLE algorithm and its extensions [29], and the PISO algorithm [30]. More recently, discontinuous Galerkin methods have been devised [31], [32]. These methods provide some advantages over finite volume methods in that parallelism and boundary condition imposition are somewhat easier to implement. Some particle methods have been developed for fluid models, including smoothed-particle hydrodynamics [33], though they have yet to see widespread adoption.

In the context of plasma physics, the term “hybrid” refers only to the combination of kinetic and fluid methods, and the manner of their combination is left unspecified. Common hybrid paradigms include:

1. splitting the domain into kinetic and fluid regions [34], [35];
2. using a fluid model for some species and a kinetic model for others [36]–[38];
3. and using both a fluid model and a kinetic model for the same species at the same time [39], [40].

The second option is the primary focus of this work and shall be referred to as the “hybrid-by-species” paradigm. Its utility lies in the fact that fast, light species such as electrons often thermalize on much faster

timescales than the more massive ions. This means that the electron population is amenable to fluid modeling, whereas the ions require a kinetic treatment. Meanwhile, the third hybrid option, which shall be referred to as “single-species hybrid”, is useful in cases where a species has a thermalized subpopulation and a beam-like, non-thermal subpopulation. The former can be treated as a fluid and the latter must be kinetically modeled.

1.3 Thesis Goal

Broadly, this work is dedicated to developing methods and code combining the advantages of kinetic and fluid methods and processes for converting between the two. In particular, both hybrid-by-species and single-species hybrid models are investigated. To begin, a numerical framework for simulation of hybrid-by-species plasmas must be introduced. Chapter 2 establishes an example of such a framework to be used throughout the remainder of this work. This hybrid scheme combines a particle-in-cell method for the kinetic component with a discontinuous Galerkin method for the fluid component, both chosen for their widespread use throughout the literature. An algorithm for interleaving their disparate time integration schemes is presented and justified with an operator splitting argument. Note that the hybrid-by-species scheme can also be used in a single-species hybrid context by simply splitting a species into two or more effective species.

The algorithms from Chapter 2 must then be implemented in plasma simulation software. Chapter 3 presents hPIC2, the result of this effort [41]. A goal of this work is to demonstrate the effectiveness of hybrid models on modern supercomputers, and so hPIC2 is written with the Kokkos performance portability framework [42], which provides parallelization routines that offer the exploitation of devices such as graphics processing units. hPIC2 is designed with a number of innovations that improve output diagnostics and resilience on massively parallel supercomputers. The core kernels of hPIC2 are tested in a rigorous verification suite, some of whose results are also presented. hPIC2 also demonstrates good performance on some of the world’s most powerful supercomputers. Chapter 3 closes with demonstrations of hPIC2’s ability to simulate plasma-material interactions in complex domains with nontrivial surface features.

This work then turns its attention to single-species hybrid models. In many plasma systems, an initially far-from-equilibrium species may be driven to thermalize via collisions. This is particularly illustrated in simulations of relativistic electron beams propagating through initially quiescent gases [43]. A fluid model is not suitable for the electron beam, but as these electrons undergo collisions with the background gas, they slow and eventually form a satisfactory fluid. A method for identifying when a portion of a kinetically modeled plasma species may be adequately modeled as a fluid and converting it is desired to reduce computational cost. Such a method is described in Chapter 4. The utility of the method is demonstrated on small-scale thermalization problems, which show that the method is correctly able to identify thermalized subsets of particles in PIC codes and convert them into an underlying fluid.

Chapter 2

Constructing a Hybrid Fluid-Kinetic Numerical Scheme

In the context of hybrid fluid-kinetic models in computational plasma physics, we refer to a model that describes some species as a fluid and others kinetically as a “hybrid-by-species” model. In order to explore the capabilities of hybrid-by-species algorithms, they must first be established. This chapter constructs a hybrid-by-species numerical scheme for the electrostatic plasma model. The hybrid-by-species paradigm is described further in Section 2.1. The chosen kinetic component of the scheme, the particle-in-cell method, is introduced and described in Section 2.2. A discontinuous Galerkin fluid solver is described in Section 2.3. Finally, an interleaved time integrator combining these two components is presented in Section 2.4.

2.1 The Hybrid-by-Species Paradigm

Broadly, the hybrid-by-species approach designates certain species as fluid and the remainder as kinetic species. Each species independently follows (1.3), so we may take moments to derive fluid equations for each species separately, as desired. Consider a plasma with two species, which shall be labeled with the subscripts k and f for the kinetic and fluid species, respectively. They each satisfy (1.3), so that

$$\frac{\partial f_k}{\partial t} + \mathbf{v} \cdot \frac{\partial f_k}{\partial \mathbf{x}} + \frac{q_k}{m_k} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f_k}{\partial \mathbf{v}} = \mathcal{C}_k[f_k], \quad (2.1)$$

$$\frac{\partial f_f}{\partial t} + \mathbf{v} \cdot \frac{\partial f_f}{\partial \mathbf{x}} + \frac{q_f}{m_f} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f_f}{\partial \mathbf{v}} = \mathcal{C}_f[f_f]. \quad (2.2)$$

As in Section 1.1, we may take moments of (2.2) and pair it with an EOS to yield

$$\frac{\partial f_k}{\partial t} + \mathbf{v} \cdot \frac{\partial f_k}{\partial \mathbf{x}} + \frac{q_k}{m_k} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f_k}{\partial \mathbf{v}} = \mathcal{C}_k[f_k], \quad (2.3)$$

$$\frac{\partial y_f}{\partial t} + \nabla \cdot \mathbf{F}_f = G_f. \quad (2.4)$$

Since the fluid equations are decoupled from the kinetic species’ equation, and we have only assumed an EOS for the fluid species, the kinetic species retains all of its velocity-space information. The extension to many kinetic or fluid species is straightforward.

The advantages of the hybrid-by-species paradigm are now clear. Conceptually, splitting the plasma into fluid and kinetic components is simple and, as we shall soon see, amenable to discretization and implementation in code. The physical problems where hybrid-by-species algorithms are useful are numerous. It is often the case that one species is significantly more collisional than others, which makes it a good candidate for fluid modeling. In such cases, a fully kinetic treatment is computationally inefficient and a fully fluid treatment is inaccurate, since closure assumptions must be imposed on all species. The hybrid-by-species approach allows one to efficiently model the species that are most amenable to fluid treatment and accurately model the species that are far from equilibrium.

The governing equations for the kinetic and fluid species can be solved independently. Section 2.2 describes a method for solving (2.3), and Section 2.3 describes a method for solving (2.4).

2.2 The Kinetic Component: Particle-in-Cell Method

Particle-in-cell (PIC) algorithms are commonly used to compute the time evolution of dense statistical ensembles subject to force fields in non-equilibrium statistical mechanics [8], [19]–[21]. The ensemble’s distribution is approximated as a set of particles whose Lagrangian coordinates in a continuous phase space are calculated by integrating Newton’s second law, and which may represent more than one physical particle. The simulated particles are often called macroparticles to distinguish them from physical particles, but the terms “particle” and “macroparticle” are used interchangeably in this work. In plasma physics, these particles are subject to the Lorentz force. The electric and magnetic fields are computed at Eulerian spatial coordinates composing a mesh, typically utilizing the finite difference method as in Section A.1 or the finite element method as in Section A.2. The fields are self-consistent in that they result directly from the phase space coordinates of the particles in the domain and, in some cases, the histories thereof.

In this work, we consider the electrostatic PIC algorithm, which couples (1.3) and (1.6). Since the magnetic field is not self-consistently computed in this approximation, it is assumed that a magnetic field is externally applied.

An interval of time between an initial condition and the desired simulation termination is discretized. The time integration scheme typically used demands that this interval be partitioned evenly, so that the time-step Δt is constant. The time step and spatial resolution h are subject to stability conditions that demand [20]

$$\omega_p \Delta t \leq 0.2, \quad (2.5)$$

$$h \leq \frac{\lambda_D}{2}, \quad (2.6)$$

where the plasma frequency ω_p and Debye length λ_D are given by

$$\omega_p = \sqrt{\frac{nq^2}{m\varepsilon_0}}, \quad (2.7)$$

$$\lambda_D = \sqrt{\frac{\varepsilon_0 kT}{nq^2}}. \quad (2.8)$$

The PIC algorithm then proceeds from the initial condition via a four-part sequence performed every time-step:

1. **Field solve:** the electromagnetic fields are computed on the mesh.

2. **Gather:** electromagnetic field values are interpolated at the positions of each particle so that the acceleration over Δt can be computed.
3. **Push:** each particle's phase-space coordinates are updated according to the Lorentz force they experience.
4. **Scatter:** charged particles distribute their charge and, in some cases, current to the mesh, so that the fields can be updated in the next time-step.

We shall briefly expand on each of these steps.

2.2.1 Field solve

Given the particles' charge density, (1.6) is solved numerically, typically utilizing the finite difference method as in Section A.1 or the finite element method as in Section A.2. The details of computing the charge density will be discussed in Section 2.2.4.

Since electrons are much less massive and often hotter than the ions in a plasma, simulating them kinetically requires a much more finely resolved time grid. By assuming that the electrons follow a Maxwell-Boltzmann distribution, we can step over typical electron transport timescales. This assumption is called the adiabatic or Boltzmann electron approximation. Such an assumption modifies (1.6) by adding a nonlinear term:

$$\nabla^2 \phi = -\frac{\rho_f}{\varepsilon_0} + \frac{en_0}{\varepsilon_0} \exp\left(\frac{e\phi}{kT_e}\right), \quad (2.9)$$

where ρ_f is the charge density due to all sources except the Boltzmann electrons, n_0 is the reference density of the Boltzmann electrons, e is the elementary charge, k is the Boltzmann constant, and T_e is the temperature of the Boltzmann electrons. T_e is defined by the user, whereas n_0 is self-consistently computed in a charge-conserving manner using methods from either Hagelaar [44] or Elias [45]. For a given n_0 obtained from charge conservation, the nonlinear problem of (2.9) is practically solved using a Newton-Raphson method.

2.2.2 Gather

The PIC algorithm assumes that the single-particle distribution function takes the form

$$f(\mathbf{x}, \mathbf{v}, t) = \sum_{\alpha=1}^N w_{\alpha}(t) \delta(\mathbf{v} - \mathbf{v}_{\alpha}(t)) S(\mathbf{x}, \mathbf{x}_{\alpha}(t)), \quad (2.10)$$

where N is the number of macroparticles of the species; w_{α} is the weight of macroparticle α , which tracks how many physical particles the macroparticle represents; $\mathbf{v}_{\alpha}$ is the Lagrangian velocity of the macroparticle; $\mathbf{x}_{\alpha}$ is the Lagrangian position of the macroparticle; and S is a shape function, which determines how macroparticles distribute charge to the mesh and, with the finite difference method, how the fields are interpolated to macroparticle positions.

The shape function effectively imposes a finite size on the macroparticles. Some care must be taken to ensure that this finite size does not introduce nonphysical effects or additional error. Indeed, it can be shown that as long as the size of the macroparticles is at most on the order of a Debye length, both linear and nonlinear electrostatic wave effects are correctly reproduced [46]. It is algorithmically convenient to choose a shape function with support in only one mesh element, so that the finite size is less than or equal to h . Together with (2.6), this means that the shape function exhibits the desired behavior.

The details of the gather step differ between FDM and FEM. These are respectively described in the following sections.

Finite difference method

The most common gather scheme uses a linear or “cloud-in-cell” weighting, so that the shape function is given by

$$S(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 - \frac{|\mathbf{x}-\mathbf{y}|}{V} & \text{if } \mathbf{x} \text{ in same cell as } \mathbf{y}, \\ 0 & \text{otherwise,} \end{cases} \quad (2.11)$$

where V is the volume of the cell containing $\mathbf{x}$.

With this interpretation, the value of the electric and magnetic fields at particle positions can be computed as

$$\mathbf{E}_\alpha = \sum_{i \in \mathcal{N}_\alpha} \mathbf{E}_i S(\mathbf{x}_i, \mathbf{x}_\alpha), \quad (2.12)$$

and similarly for $\mathbf{B}$, where $\mathcal{N}_\alpha$ is the set of all grid points in the cell containing particle α .

Finite element method

Recall that, from (A.15), the electric potential is approximated as

$$\phi \approx u_h = \sum_{i=1}^M z_i \psi_i, \quad (2.13)$$

for basis functions ψ_i of S_h . The electric field is therefore approximated as

$$\mathbf{E} = -\nabla \phi \approx -\sum_{i=1}^M z_i \nabla \psi_i, \quad (2.14)$$

which can be evaluated at particle positions.

Similarly, the magnetic field is approximated in the finite element space, so that

$$\mathbf{B} \approx \sum_{i=1}^M \mathbf{B}_i \psi_i. \quad (2.15)$$

2.2.3 Push

Boris’ method [19], [47], [48], a type of leapfrog method, is used to update particle coordinates in phase space. Positions are stored at time steps and velocities at half steps. Denote the value of a time-dependent function at time step k by $f^k = f(k\Delta t)$, and analogously for values at half steps. The method accelerates for half of a time step due to the electric field, rotates the intermediate velocity due to the magnetic field, then completes

the acceleration due to the electric field:

$$\begin{aligned}
\mathbf{v}^- &= \mathbf{v}^{k-1/2} + \frac{\Delta t}{2} \frac{q}{m} \mathbf{E}^k, \\
\tau &= \frac{\mathbf{B}^k}{B^k} \tan\left(\frac{\Delta t}{2} \frac{q}{m} B^k\right), \\
\mathbf{v}' &= \mathbf{v}^- + \mathbf{v}^- \times \tau, \\
\mathbf{v}^+ &= \mathbf{v}^- + 2\mathbf{v}' \times \frac{\tau}{1 + \tau^2}, \\
\mathbf{v}^{k+1/2} &= \mathbf{v}^+ + \frac{\Delta t}{2} \frac{q}{m} \mathbf{E}^k, \\
\mathbf{x}^{k+1} &= \mathbf{x}^k + \mathbf{v}^{k+1/2} \Delta t,
\end{aligned} \tag{2.16}$$

$$\tag{2.17}$$

where we have suppressed the particle α subscript for brevity.

2.2.4 Scatter

The details of the scatter step differ between FDM and FEM. These are respectively described in the following sections.

Finite difference method

Given (2.10), the charge density can be computed as

$$\rho(\mathbf{x}) = q \int_{\mathbb{R}^3} d\mathbf{v} f(\mathbf{x}, \mathbf{v}) = q \sum_{\alpha=1}^N w_{\alpha} S(\mathbf{x}, \mathbf{x}_{\alpha}), \tag{2.18}$$

where we have suppressed the time dependence for brevity. Note that the charge density at grid points depends only on particles contained in adjacent grid cells. Hence, the nodal charge density ρ_i can be written as

$$\rho_i \equiv \rho(\mathbf{x}_i) = q \sum_{\alpha} w_{\alpha} S(\mathbf{x}_i, \mathbf{x}_{\alpha}), \tag{2.19}$$

where the sum is over particles in cells containing grid point i .

Finite element method

Again, the charge density is given by (2.18). However, in the finite element method, it is typical to use Dirac delta distributions for the particle shape functions:

$$S(\mathbf{x}, \mathbf{y}) = \delta(\mathbf{x} - \mathbf{y}). \tag{2.20}$$

Hence, the vector $\mathbf{b}$ as in (A.16) is defined such that

$$b_i = q \sum_{\alpha=1}^N w_{\alpha} \psi_i(\mathbf{x}_{\alpha}). \tag{2.21}$$

Generally, the basis functions ψ_i are supported only in the elements to which they are associated, so the particle sum can be restricted to include only particles contained in those elements.

2.2.5 Particle boundary conditions

PIC permits a considerable range of boundary conditions. Boundary conditions that may be difficult to impose in fluids are often far simpler in a PIC context. Heuristically motivated particle boundary conditions often exactly reproduce desired plasma behavior.

A simple example is the absorbing boundary condition, which removes a macroparticle from the simulation when it reaches the boundary. This is useful in the case of ions impinging on a material wall, since most ions are captured by the material and those that are reflected are often neutralized over the course of their interactions with the wall. Similarly, electrons incident on metal walls are often completely absorbed.

Although the absorbing boundary condition can be useful in many cases, detailed PIC simulations of plasma-material interactions require a somewhat more sophisticated model. In particular, the sputtering and erosion rates of materials are often of interest. In these cases, binary-collision approximation codes can be used to generate tables of sputtering yields as functions of incident ion energy and angle [49]. During subsequent PIC simulations, the table can be used to determine yields of sputtered material, given each incident ion’s energy and angle of incidence [50]. The sputtered material can then be tracked as it is transported throughout the plasma.

Infinite or semi-infinite domains can be approximated with periodic or perfectly reflecting boundaries. When a macroparticle reaches the edge of a periodic domain, it is moved to the opposite side of the domain with its velocity unchanged. This can be a good approximation to infinite domains in the direction of periodicity, but can introduce undesired plasma waves with wavelength equal to the domain size. Perfectly reflecting boundaries, meanwhile, reverse the velocity of a macroparticle normal to the boundary; that is, the velocity is updated to become

$$\mathbf{v}' = \mathbf{v} - 2\hat{n}(\mathbf{v} \cdot \hat{n}), \quad (2.22)$$

where $\hat{n}$ is the outward-facing unit normal to the boundary. Since this boundary condition can be considered to spawn a new particle originating from the other side of the boundary, it can be used as an approximation to a semi-infinite domain.

2.2.6 Kinetic collision models

The PIC algorithm as presented thus far is strictly applicable only to collisionless plasmas, where $\mathcal{C}[f] = 0$ in (1.3). Monte Carlo methods, which exploit randomness to efficiently model complex systems, are often used to augment the PIC algorithm and restore the effects of collisions. PIC particles may collide with each other or with species not represented by PIC particles, and the algorithms used in each of these cases differ. In Section 2.2.6, a Monte Carlo method for modeling PIC particle collisions with continuum background species is presented. In Section 2.2.6, an algorithm for modeling collisions between PIC particles is discussed.

Monte Carlo collisions

Monte Carlo collision (MCC) methods form a family of algorithms that model collisions between a PIC “source” species and a continuum “target” species. Broadly, MCC methods work by randomly sampling particles from the target species, computing a collision probability for each particle pair, and then stochastically determining whether a collision occurs. The industry standard within the family of MCC algorithms is the null collision method [51], which allows for the efficient modeling of multiple collision processes; throughout this section, when we refer to MCC, we refer specifically to this method.

MCC allows a source species to collide with arbitrarily many target species, but an individual source particle may only undergo a maximum of one collision per time step. Suppose that the source species is subject to M collision processes. For each source particle, we could compute a collision probability from the sum of collision rates over all M possible collisions as

$$P = 1 - \exp\left(-\Delta t \sum_{i=1}^M n_i \sigma_i(\mathbf{g}_i) g_i\right), \quad (2.23)$$

where $n_i = n_i(\mathbf{x})$ is the local number density of the target species for collision i , $\sigma_i = \sigma_i(\mathbf{g}_i)$ is the collision cross-section function, and $\mathbf{g}_i$ is the relative velocity between the source particle and its target collision partner. Collision partners are drawn randomly from the target species' distribution.

It can be expensive to compute the collision probability for each source particle. Instead, we add a fictitious “null” collision whose cross-section is such that the total collision frequency is constant. This is illustrated in Figure 2.1. We choose the constant collision frequency to be

$$\nu' = \max_{\mathbf{x}, \mathbf{g}} \sum_{i=1}^M n_i(\mathbf{x}) \sigma_i(\mathbf{g}) g. \quad (2.24)$$

Now the maximum fraction of particles that will undergo collisions is

$$P_{\text{null}} = 1 - \exp(-\Delta t \nu') \approx \Delta t \nu'. \quad (2.25)$$

Rather than compute the collision probability for each source particle, we now only need to compute the collision rates of a fraction P_{null} of source particles. Given N particles, $\lceil NP_{\text{null}} \rceil$ are randomly sampled without replacement. We must then decide which type of collision will occur for each such particle. Uniformly draw a random number $U \in [0, 1)$. Then collision i will occur if

$$\frac{\sum_{j=1}^{i-1} n_j \sigma_j g_j}{\nu'} \leq U < \frac{\sum_{j=1}^i n_j \sigma_j g_j}{\nu'}, \quad (2.26)$$

and the null collision (that is, nothing) will occur if

$$\frac{\sum_{j=1}^M n_j \sigma_j g_j}{\nu'} \leq U \quad (2.27)$$

for the selected particle, again where collision partners are drawn randomly from the target species' distribution. This strategy effectively weights the possible collisions by their respective collision rates.

Many things can happen when a collision occurs. If the collision is meant to model ionizing source electrons impinging on a neutral target, the collision may spawn a new electron and a new ion; if the collision models elastic scattering events, the source particle's trajectory may simply be skewed to reflect the collision.

Direct simulation Monte Carlo

Direct simulation Monte Carlo (DSMC) methods form a family of algorithms for collisions between particle-based species. The most common method for DSMC is the no-time-counter (NTC) scheme, which first computes a maximum number of collisions, selects that many particle pairs, then decides between possible collision processes with some probability [52]. The original NTC scheme does not allow for variable particle

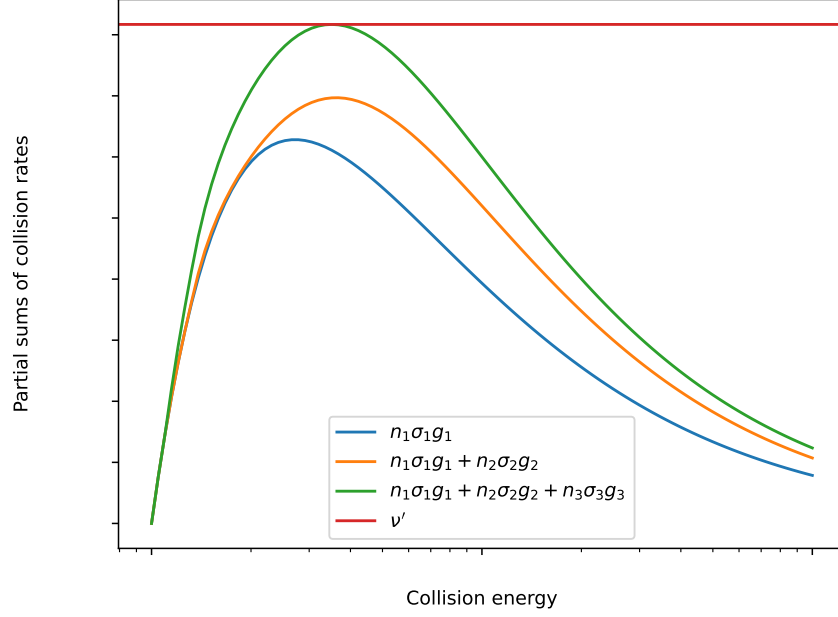


Figure 2.1: An example of the addition of a null collision to three possible collision processes. Partial sums of the collision rates are plotted as functions of collision energy, i.e. kinetic energy in the center-of-mass frame of the collision. The null collision’s cross-section is chosen such that the total collision rate is constant. This means that the collision probability for any one particle pair is independent of the details of the collision.

weights, so the stochastic weight particle method (SWPM) was developed to address this modeling gap [53], but was restricted to collisions between particles of the same species. The SWPM was subsequently extended to the case of interspecies collisions [54]. This latter approach is discussed in this section.

First, for each pair of species that are participating in at least one collision process, define

$$\chi = \max_{\mathbf{g}} \sum_{i=1}^M \sigma_i(\mathbf{g})g, \quad (2.28)$$

where M is the number of collision processes between the two species and $\sigma_i(\mathbf{g})$ is the cross-section of reaction i as a function of relative velocity $\mathbf{g}$. For each mesh element, the maximum number of collisions is computed as

$$N_{\text{coll}} = N_s N_t (\bar{w}_s + \bar{w}_t - \min w) \frac{\chi \Delta t}{V}, \quad (2.29)$$

where N_s and N_t are the numbers of macroparticles of the source and target species, respectively,

$$\bar{w}_s = \frac{1}{N_s} \sum_{\alpha=1}^{N_s} w_{\alpha} \quad (2.30)$$

and

$$\bar{w}_t = \frac{1}{N_t} \sum_{\beta=1}^{N_t} w_{\beta} \quad (2.31)$$

are the average weights of macroparticles in the two species, $\min w$ is the minimum macroparticle weight across both species, and V is the volume of the mesh element. In practice, if N_{coll} is computed to be small

(< 64), it is sampled from a Poisson distribution, and otherwise rounded up to the nearest integer.

For each of N_{coll} possible collisions, a pair of macroparticles (say, particle α from species s and particle β from species t) is randomly sampled and accepted with probability

$$P_{\alpha\beta} = \frac{w_\alpha + w_\beta - \min w}{\max_s w + \max_t w - \min w}, \quad (2.32)$$

where the maximum weights are computed over macroparticles of the appropriate species. If a pair is rejected, the process of selecting a new pair is repeated until the pair is accepted. We must then decide which type of collision will occur for each such pair, if any. Uniformly draw a random number $U \in [0, 1)$. Then collision i will occur if

$$\frac{\max(w_\alpha, w_\beta)}{w_\alpha + w_\beta - \min w} \frac{\sum_{j=1}^{i-1} \sigma_j g}{\chi} \leq U < \frac{\max(w_\alpha, w_\beta)}{w_\alpha + w_\beta - \min w} \frac{\sum_{j=1}^i \sigma_j g}{\chi}, \quad (2.33)$$

and the null collision will occur if

$$\frac{\max(w_\alpha, w_\beta)}{w_\alpha + w_\beta - \min w} \frac{\sum_{j=1}^M \sigma_j g}{\chi} \leq U. \quad (2.34)$$

If the null collision occurs, nothing further happens to this pair. Otherwise, the particle with the greater weight is copied. This new particle has the smaller of the two weights, and the original particle has its weight reduced to the difference of the greater and smaller weights. The copied particle and smaller particle finally collide according to the reaction type. This particle splitting ensures detailed mass, momentum, and energy conservation. N_s and N_t are also updated to reflect the addition of new particles, effectively allowing the new particles to collide multiple times.

If $s = t$, i.e., if the species is colliding with itself, the only change to this procedure is to modify the maximum number of collisions to be [55]

$$N_{\text{coll}} = N_s(N_s - 1)(2\bar{w}_s - \min w) \frac{\chi \Delta t}{V}. \quad (2.35)$$

2.3 The Fluid Component: Discontinuous Galerkin Methods

This section describes a numerical method for solving (1.23). The fluid equations are solved broadly using the local discontinuous Galerkin method [31], in which the system is first partially discretized in space, and the resulting system of ordinary differential equations is then discretized in time.

2.3.1 Spatial discretization

A discontinuous Galerkin (DG) method is used to discretize in space [56]. DG methods can be thought of as finite element methods that incorporate aspects of the finite volume method. In contrast with continuous Galerkin methods, a family to which the algorithm presented in Section A.2 belongs, DG methods define a separate finite element space for each element in the mesh, so that finite element solutions are not constrained to be continuous. This renders the linear algebra extremely local, and parallelism is therefore relatively easy with DG methods.

We restrict our purview to DG methods on admissible meshes, as described in Section A.2. Let $\mathcal{T}_h$ be such a mesh on the polyhedral problem domain Ω . Let $\mathcal{F}_I$ be the set of interior faces in $\mathcal{T}_h$, with the convention

that in two dimensions, edges are referred to as faces, and in one dimension, vertices are referred to as faces. Similarly, let $\mathcal{F}_B$ be the set of faces in $\mathcal{T}_h$ that lie on $\partial\Omega$.

Define

$$H^s(\mathcal{T}_h) = \{\psi \in L^2(\Omega) \mid \forall T \in \mathcal{T}_h, \psi|_T \in H^s(T)\}, \quad (2.36)$$

where $H^s(T)$ is a Sobolev space of order $s > 1/2$ on the mesh element T . The weak form of (1.23) is essentially derived by choosing an element $T \in \mathcal{T}_h$, multiplying both sides of the equation by some $\psi \in H^s(T)$, and integrating by parts over T to yield

$$\frac{d}{dt} \int_T y \psi \, d\mathbf{x} + \int_{\partial T} \hat{n} \cdot \mathbf{F} \psi \, dS - \int_T \mathbf{F} \cdot \nabla \psi \, d\mathbf{x} = \int_T G \psi \, d\mathbf{x}, \quad (2.37)$$

where $\hat{n}$ is the outward-facing unit normal on ∂T .

For fluids, it is appropriate to approximate the flux on faces as the flux arising from the solution to the Riemann problem on that face [31], [57], so that

$$\int_{\partial T} \hat{n} \cdot \mathbf{F} \psi \, dS \approx \int_{\partial T} h(y^-, y^+) \psi \, dS, \quad (2.38)$$

where $h(y^-, y^+)$ is the solution to the Riemann problem for the flux at the interface, using y^+ , the state just outside T , as the “right” state, and y^- , the state just inside T , as the “left” state [58]. Hence, the weak form becomes

$$\frac{d}{dt} \int_T y \psi \, d\mathbf{x} + \int_{\partial T} h(y^-, y^+) \psi \, dS - \int_T \mathbf{F} \cdot \nabla \psi \, d\mathbf{x} = \int_T G \psi \, d\mathbf{x}. \quad (2.39)$$

A weak solution y satisfies (2.39) for all $T \in \mathcal{T}_h$ and for all $\psi \in H^s(\mathcal{T}_h)$.

It is obviously impossible to verify a potential solution y against every ψ in an infinite-dimensional space. We therefore constrain our attention to a finite-dimensional subspace $V \in H^s(\mathcal{T}_h)$, the finite element space. With DG, it is convenient to choose a basis consisting of functions that are supported on a single element. For each $T \in \mathcal{T}_h$, we construct a finite basis $\{\psi_1^T, \dots, \psi_N^T\}$ of size $N = N(T)$. Then the finite element space is the span of the union of bases in every element, $V = \text{span} \cup_{T \in \mathcal{T}_h} \{\psi_1^T, \dots, \psi_{N(T)}^T\}$. Henceforth, we suppress the T dependence where it is clear, for brevity.

We are considering only weak solutions in the finite element space, so that

$$y_h|_T(\mathbf{x}, t) = \sum_{i=1}^N y_i(t) \psi_i(\mathbf{x}). \quad (2.40)$$

(2.39) becomes

$$\frac{d}{dt} \sum_{i=1}^N y_i \int_T \psi_i \psi_j \, d\mathbf{x} + \int_{\partial T} h(y_h^-, y_h^+) \psi_j \, dS - \int_T \mathbf{F}(y_h) \cdot \nabla \psi_j \, d\mathbf{x} = \int_T G(y_h) \psi_j \, d\mathbf{x}, \quad (2.41)$$

when tested against a basis function ψ_j .

Let $\mathbf{M}$ be the $N \times N$ matrix with entries

$$M_{ij} = \int_T \psi_i \psi_j \, d\mathbf{x}, \quad (2.42)$$

and let $\mathbf{K}_{\text{face}}$, $\mathbf{K}_{\text{flux}}$, and $\mathbf{K}_{\text{source}}$ be the nonlinear maps defined by

$$K_{\text{face},j} = - \int_{\partial T} h(y_h^-, y_h^+) \psi_j \, dS, \quad (2.43)$$

$$K_{\text{flux},j} = \int_T \mathbf{F}(y_h) \cdot \nabla \psi_j \, d\mathbf{x}, \quad (2.44)$$

$$K_{\text{source},j} = \int_T G(y_h) \psi_j \, d\mathbf{x}. \quad (2.45)$$

Then, with $\mathbf{y} = (y_1, \dots, y_N)^t$ and $\mathbf{R} = \mathbf{M}^{-1} (\mathbf{K}_{\text{face}} + \mathbf{K}_{\text{flux}} + \mathbf{K}_{\text{source}})$, (2.41) can be written as

$$\frac{d\mathbf{y}}{dt} = \mathbf{R}(\mathbf{y}, t), \quad (2.46)$$

which is a system of ordinary differential equations. Since the basis functions are usually taken to be polynomials, the integrals are typically computed using Gaussian quadrature.

2.3.2 Time discretization

Now that we have reduced our problem to a coupled system of ordinary differential equations, we may use one of a number of numerical methods for discretizing in time. Runge-Kutta (RK) methods compute approximate solutions at intermediate stages between the current simulation time step and the next time step, then evaluate the solution at the next time step as a weighted average of the estimates at the stages [59], [60]. This is in contrast to linear multistep methods, which use the solution from perhaps several previous time steps to evaluate the solution at the next time step [61]. RK methods have the distinct advantage that initial conditions are simpler to specify, since we do not need to provide the solution at time steps prior to the start of the simulation.

Throughout this section, let $\mathbf{y}^n$ and t^n denote the solution and time at time step n , respectively. Generally, RK methods approximate the solution at subsequent time steps as

$$\mathbf{y}^{n+1} = \mathbf{y}^n + \Delta t \sum_{i=1}^s b_i \mathbf{k}_i, \quad (2.47)$$

where s is the number of stages in the scheme and

$$\mathbf{k}_i = \mathbf{R} \left(\mathbf{y}^n + \Delta t \sum_{j=1}^s a_{ij} \mathbf{k}_j, t^n + c_i \Delta t \right), \quad (2.48)$$

for real c_i , a_{ij} , and b_i . An RK method is uniquely specified by the choices of these constants, which are often presented in a Butcher tableau

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \quad (2.49)$$

The subset of explicit RK methods is particularly important for fluid solvers in plasma physics. For explicit

methods, the stages are of the form

$$\mathbf{k}_i = \mathbf{R} \left(\mathbf{y}^n + \Delta t \sum_{j=1}^{i-1} a_{ij} \mathbf{k}_j, t^n + c_i \Delta t \right), \quad (2.50)$$

so that a given stage depends only on previous stages. Butcher tableaux for explicit RK methods are therefore lower triangular

$$\begin{array}{c|cccc} c_1 & & & & \\ c_2 & a_{21} & & & \\ \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & \cdots & a_{s,s-1} & \\ \hline & b_1 & \cdots & b_{s-1} & b_s \end{array} \quad (2.51)$$

RK methods that are not explicit are implicit. Implicit methods require the solution of systems of equations at one or more stages, which, in the case of the Euler equations (1.18), will be nonlinear. However, explicit methods tend to have much smaller regimes of numerical stability than implicit methods; that is, explicit methods typically demand a smaller time step for stability.

While different explicit RK methods have different regions of stability, the stable time step is typically associated with the characteristic timescales of the system. We shall simply state them here. Euler fluids are associated with a signal speed timescale

$$t_s = \frac{h}{c + u}, \quad (2.52)$$

where h is a characteristic length scale of the system (in this context, usually the size of a mesh element), c is the fluid's sound speed, and u is the bulk speed of the fluid. Generally, the sound speed can be computed as

$$c = \sqrt{\left(\frac{\partial p}{\partial (nm)} \right)_S}, \quad (2.53)$$

where the derivative is computed with constant specific entropy S . For the ideal gas law EOS presented in Section 2, this becomes

$$c = \sqrt{\gamma \frac{p}{nm}}. \quad (2.54)$$

The source terms G in (1.18) may introduce additional timescales. For example, the timescale associated with the electric field acceleration in the Lorentz force term is the plasma oscillation period

$$t_p = 2\pi \sqrt{\frac{m\varepsilon_0}{nq^2}}. \quad (2.55)$$

Similarly, the timescale associated with the magnetic field in the Lorentz force term is the cyclotron period

$$t_c = \frac{2\pi m}{qB}. \quad (2.56)$$

In explicit fluid solvers, the time step is usually chosen to be

$$\Delta t = C t_{\min}, \quad (2.57)$$

where $C > 0$ is called the Courant-Friedrichs-Lewy (CFL) number [62] and $t_{\min}$ is the minimum timescale in the simulation. Most explicit RK methods require $C \leq 1$. Implicit methods may permit much larger CFL numbers.

2.3.3 Stabilization

Godunov's theorem [63] states that a linear, monotone scheme for partial differential equations can be at most first-order accurate. The contrapositive of this statement is that linear high-order schemes necessarily introduce spurious oscillations, Gibbs phenomena, that do not appear in the exact solutions. Gibbs phenomena do indeed appear near discontinuities with the schemes presented in this section.

Many techniques have been introduced to mitigate these oscillations. Artificial viscosity introduces non-physical diffusive terms to the fluid equations that are mostly quiescent in smooth regions but gradually turn on near discontinuities in order to spread out shocks [64]. Weighted essentially non-oscillatory schemes modify the polynomial interpolation stencil in areas near discontinuities in order to reduce oscillations [65]. Limiters reduce either the fluxes (flux limiters) or the fluid state itself (slop limiters) to reasonable values where gradients are excessively great. Limiters are generally difficult to extend to high-order DG spatial discretizations, but recent research has expanded the number of options, e.g., the limiter from Moe et al. [66].

2.3.4 Riemann solvers

As described in Section 2.3.1, the numerical flux through interfaces is approximated in (2.41) as the solution to a Riemann problem. Exact solutions to Riemann problems for fluid equations (in particular, (1.18)) can be difficult to compute, so a vast body of literature has been dedicated to approximate Riemann solvers [58].

A Riemann problem is an initial value problem for a conservation equation with piecewise constant initial conditions consisting of a single discontinuity at the origin. Consider (1.23) in an infinite domain with initial data

$$y(\mathbf{x}, 0) = \begin{cases} y^- & \text{if } x < 0, \\ y^+ & \text{if } x > 0. \end{cases} \quad (2.58)$$

We are interested in the solution at the origin at some later time $t > 0$. In fact, in (2.41), we only need the flux of the solution $\mathbf{F}(y)$ at the origin. Many Riemann solvers return the approximate flux without attempting to compute the solution itself. Some examples are described below.

Harten-Lax-van Leer solver

The Harten-Lax-van Leer (HLL) solver estimates the state as [67]

$$y(\mathbf{x}, t) = \begin{cases} y^- & \text{if } \frac{x}{t} \leq s^-, \\ y^{\text{HLL}} & \text{if } s^- \leq \frac{x}{t} \leq s^+, \\ y^+ & \text{if } \frac{x}{t} \geq s^+, \end{cases} \quad (2.59)$$

where the s^* are signal speeds on either side of the discontinuity and

$$y^{\text{HLL}} = \frac{s^+ y^+ - s^- y^- + F_x(y^-) - F_x(y^+)}{s^+ - s^-}. \quad (2.60)$$

By using the Rankine-Hugoniot conditions, we can compute the corresponding fluxes for use in the numerical DG flux as

$$h_{\text{HLL}}(y^-, y^+) = \begin{cases} \mathbf{F}(y^-) \cdot \hat{n} & \text{if } 0 \leq s^-, \\ \frac{[s^+ \mathbf{F}(y^-) - s^- \mathbf{F}(y^+)] \cdot \hat{n} + s^- s^+ (y^+ - y^-)}{s^+ - s^-} & \text{if } s^- \leq 0 \leq s^+, \\ \mathbf{F}(y^+) \cdot \hat{n} & \text{if } 0 \geq s^+. \end{cases} \quad (2.61)$$

There are a number of possible estimates for the signal speeds. A simple choice is [68]

$$s^- = \mathbf{u}^- \cdot \hat{n} - c^-, \quad (2.62)$$

$$s^+ = \mathbf{u}^+ \cdot \hat{n} + c^+, \quad (2.63)$$

where $\mathbf{u}^*$ and c^* are the bulk velocity and sound speed on either side of the interface, respectively.

Rusanov solver

The Rusanov solver [69], also known as the local Lax-Friedrichs (LLF) solver, is an extremely robust yet diffusive solver that can be considered a special case of the HLL solver for certain choices of signal speeds. The numerical flux is given by

$$h_{\text{LLF}}(y^-, y^+) = \frac{1}{2} [\mathbf{F}(y^-) + \mathbf{F}(y^+)] \cdot \hat{n} - \frac{1}{2} s^* (y^+ - y^-), \quad (2.64)$$

where s^* is a single signal speed. A typical choice is [68]

$$s^* = \max\{u^- + c^-, u^+ + c^+\}. \quad (2.65)$$

2.3.5 Boundary conditions

Boundary conditions are typically applied by considering a “ghost” fluid state just on the outside of the domain and computing the resulting numerical flux in (2.41) using a Riemann solver.

As an example, an impermeable wall can be modeled by making the ghost fluid state equal to the state just inside the boundary, but with a bulk velocity that has been reversed normal to the boundary. That is, if the fluid state just inside the wall is

$$y^- = \begin{pmatrix} mn^- \\ mn^- \mathbf{u}^- \\ mn^- E^- \end{pmatrix}, \quad (2.66)$$

the state just outside the wall should be set to

$$y^+ = \begin{pmatrix} mn^- \\ mn \mathbf{u}^- - 2\hat{n}(mn \mathbf{u}^- \cdot \hat{n}) \\ mn E^- \end{pmatrix}. \quad (2.67)$$

This ensures that the boundary effectively reflects fluid flow.

2.4 Hybrid Time Integration

A hybrid fluid-kinetic scheme results from combining the PIC methods presented in Section 2.2 with the DG fluid methods presented in Section 2.3. This can be further augmented with interactions furnished by the DSMC and MCC algorithms presented in Section 2.2.6. However, the time integrators for the fluid and kinetic components presented here are quite different, so a hybrid time integrator reconciling these differences must be developed. To that end, a fractional step method may be used to combine the two disparate time integrators [27]. Fractional step methods are a class of algorithms that decompose the differential operator in ordinary differential equations into a sum of simpler operators, which can then be solved in sequence. That is, if we have a differential operator that can be split into two or more operators, and we have a numerical integrator for each of these split operators, successive application of the numerical integrators will approximate the solution to the original differential equation, with some additional but manageable error.

As a simple example, consider the differential equation

$$\frac{dy}{dt} = L_1(y) + L_2(y), \quad (2.68)$$

where L_1 and L_2 are differential operators. Suppose that $\mathcal{H}_1$ is a numerical method for solving

$$\frac{dy}{dt} = L_1(y), \quad (2.69)$$

so that $y^{n+1} = \mathcal{H}_1 y^n$. Define $\mathcal{H}_2$ analogously. Then the solution to the original differential equation (2.68) can be approximated as

$$y^{n+1} = \mathcal{H}_2 \mathcal{H}_1 y^n, \quad (2.70)$$

and it can be shown that this approximation is at most first-order accurate [70]. One could also use a Strang splitting method [71], which reduces the error at the cost of a somewhat more complicated algorithm. For brevity, this section only considers the simpler first-order method.

To see how this method can be applied to the hybrid fluid-kinetic scheme, note that, after generating the macroparticles for the PIC method and semi-discretizing the fluid equations in space, the combined ordinary differential equation takes the form

$$\begin{aligned} \frac{d}{dt} \begin{pmatrix} \mathbf{y}_k \\ \mathbf{y}_f \end{pmatrix} &= \left[\begin{pmatrix} \mathbf{R}_k + \mathbf{C}_{kk} & 0 \\ 0 & \mathbf{R}_f \end{pmatrix} + \mathbf{C}_{kf} \right] \begin{pmatrix} \mathbf{y}_k \\ \mathbf{y}_f \end{pmatrix} \\ &= \left[\begin{pmatrix} \mathbf{R}_k & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{R}_f \end{pmatrix} + \begin{pmatrix} \mathbf{C}_{kk} & 0 \\ 0 & 0 \end{pmatrix} + \mathbf{C}_{kf} \right] \begin{pmatrix} \mathbf{y}_k \\ \mathbf{y}_f \end{pmatrix}, \end{aligned} \quad (2.71)$$

where $\mathbf{y}_k$ is a vector encoding the state of the macroparticles of the kinetic species, such as macroparticle positions and velocities; $\mathbf{y}_f$ is the finite element solution to the fluid equations; $\mathbf{R}_k$ is the operator encoding the collisionless time evolution of the macroparticles; $\mathbf{R}_f$ is the right-hand side of (2.46); $\mathbf{C}_{kk}$ is an operator encoding the particle-particle collisions, which will be handled through DSMC; and $\mathbf{C}_{kf}$ is an operator encoding the particle-fluid collisions, which will be handled through MCC. Let $\mathcal{H}_k$ be the numerical method corresponding to leapfrog PIC time integration, $\mathcal{H}_f$ be the RK fluid time integrator, $\mathcal{H}_{\text{DSMC}}$ be the DSMC

operator, and $\mathcal{H}_{\text{MCC}}$ be the MCC operator. Then the numerical method

$$\mathcal{H} = \mathcal{H}_{\text{MCC}} \begin{pmatrix} \mathcal{H}_{\text{DSMC}} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & \mathcal{H}_{\text{f}} \end{pmatrix} \begin{pmatrix} \mathcal{H}_{\text{k}} & 0 \\ 0 & 0 \end{pmatrix} \quad (2.72)$$

approximately solves (2.71), following the fractional step method. That is, the combined solution at the next time step can be approximated iteratively from the current time step as

$$\begin{pmatrix} \mathbf{y}_{\text{k}}^{n+1} \\ \mathbf{y}_{\text{f}}^{n+1} \end{pmatrix} = \mathcal{H} \begin{pmatrix} \mathbf{y}_{\text{k}}^n \\ \mathbf{y}_{\text{f}}^n \end{pmatrix}. \quad (2.73)$$

Heuristically, this method can be implemented using the following process:

1. particles are pushed to advance them forward by one time step;
2. the fluids' RK integrator is invoked to advance it forward by one time step;
3. interactions are performed, possibly involving both the particles and the fluids;
4. given the new state of the particles and fluids, the electric potential is updated.

This procedure is illustrated in Fig. 2.2. The result is a scheme that is capable of modeling a wide range of plasma phenomena, including high-pressure, dense species as fluids, low-pressure, dilute species as particles, and the interactions arising between them.

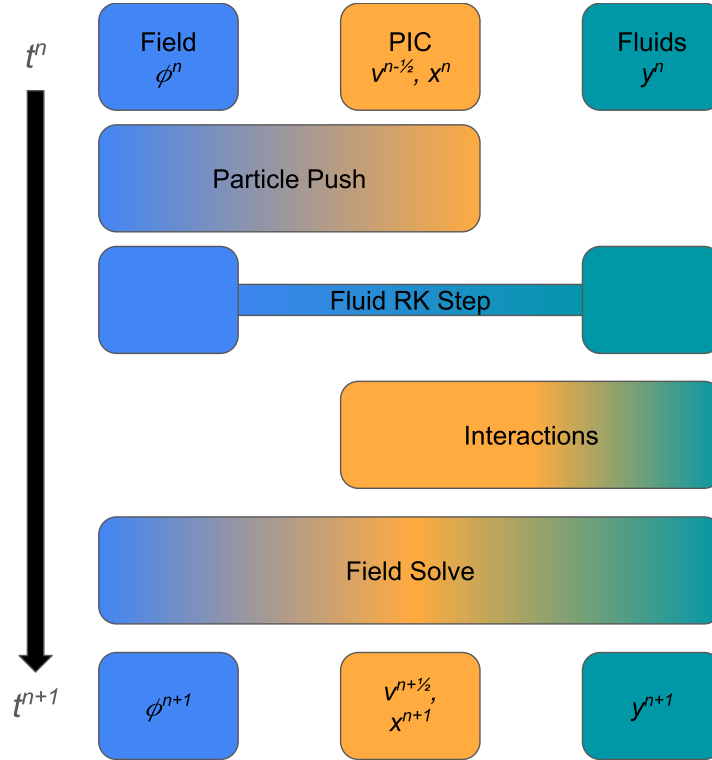


Figure 2.2: Diagram showing a hybrid fluid-PIC time stepping algorithm. The algorithm proceeds from top to bottom as it advances in time.

2.4.1 Fluid to kinetic conversion

The $\mathbf{C}_{kf}$ operator bears additional discussion. The MCC method described in Section 2.2.6 is widely used to handle collisions between PIC particles and static background species. However, the method requires significant modifications to handle collisions between PIC particles and a fluid. In particular, the standard MCC method assumes that the background species is considerably denser than the PIC species, so that collisional effects on the background species can be neglected. Since this is generally not the case with hybrid fluid-PIC simulations, the effects of collisions on the fluid must be restored.

Part of the motivation for fluid descriptions is that they discard detailed information about the fluid's underlying distribution. This is at odds with MCC, which requires a sample of the target distribution to act as a collision partner for the source particle. The finitely many moments of a fluid method are not sufficient to uniquely determine a distribution from which to draw a collision partner, so additional assumptions must be made. Since the fluid is assumed to be in or close to local thermodynamic equilibrium, the most appropriate distribution from which to draw a collision partner is the maximum-entropy distribution satisfying the fluid's moments and EOS. For the Euler equations (1.18), this is the drifting Maxwellian distribution. For other fluid models, the Pearson family of distributions [72] or expansions in Hermite polynomials [73], [74] may be more appropriate.

The hybrid MCC method accounts for the effect of collisions on the fluid by adjusting the fluid moments according to the collision partner's moments. In the process of drawing a collision partner from the fluid, we temporarily remove that particle from the fluid state. As a result, that particle's contribution to the fluid moments must also be removed. For every moment polynomial ψ composing the fluid as in (1.16), the quantity

$$\frac{w}{V}\psi(\mathbf{v}_t) \quad (2.74)$$

is removed from all DG degrees of freedom in the mesh element in which the collision occurs, where w is the weight of the source macroparticle and $\mathbf{v}_t$ is the velocity of the target fluid particle. For example, with the Euler equations, the quantities

$$\Delta nm = \frac{w}{V}m, \quad (2.75)$$

$$\Delta nm\mathbf{u} = \frac{w}{V}m\mathbf{v}_t, \quad (2.76)$$

$$\Delta nmE = \frac{w}{V}\frac{1}{2}mv_t^2 \quad (2.77)$$

are removed from the density, momentum density, and total energy density, respectively.

After the collision, one of two further actions is taken to affect the fluid. If the collision partner is destroyed or transmuted, e.g. during a chemical combination reaction, then nothing further is done to the fluid state, and the net effect on the fluid is the removal of the collision partner's contributions to the fluid moments. On the other hand, if the collision partner is preserved, e.g. during elastic collisions, then the collision partner is reinserted into the fluid state with moments corresponding to the collision partner's post-collision velocity. This coupling method ensures exact, detailed conservation of the fluid moments.

The modifications to the standard MCC method presented here permit the simulation of interactions between PIC particles and fluids. These modifications ensure that PIC particles interact with an effective fluid distribution commensurate with the fluid model used and that the fluid moments are conserved exactly. While the method imposes additional assumptions on the fluid distribution, these assumptions are reasonable in the context of fluid models.

Chapter 3

hPIC2: A Hardware-Accelerated, Hybrid Particle-in-Cell Code for Dynamic Plasma-Material Interactions

High-performance computing has come to be dominated by device acceleration. Most of the top 10 supercomputers in the TOP500 rely on accelerators as of February 2022, including Summit, Sierra, and Perlmutter. This trend will continue, as the future exascale supercomputers Aurora and Frontier will both utilize GPUs. The increased use of accelerators renders the historically dominant paradigm of multiple instruction, multiple data programming, typically facilitated by the Message Passing Interface (MPI), insufficient for fully utilizing modern supercomputing architectures when used alone. Instead, the paradigm has shifted to one in which shared-memory parallel programming models are incorporated alongside MPI, sometimes referred to as MPI+X.

The addition of a new level of parallelism furnished by accelerators introduces a potential burden to programmers in scientific computing. Completely exploiting a heterogeneous computing architecture necessitates both distributed-memory and shared-memory parallelism, but parallel programming models differ from device to device, depending on the manufacturer.

To address the nearly impossible task of rewriting architecture-specific code for every desired platform, many heterogeneous computing frameworks have been developed, including OpenACC, OpenCL, SYCL [75], and OpenMP’s target offloading [76], all of which have some industry support from device manufacturers. Of particular interest to computational physicists are RAJA [77] and Kokkos [42], both of which are parallel programming abstraction layers intended specifically to target architectures relevant to scientific computing on modern supercomputers. Kokkos describes itself as performance-portable, meaning that a single source code written with Kokkos is not only portable to a range of devices, but also maximally performant on them. Such parallel programming portability frameworks are vital to academic researchers who lack the resources to maintain separate versions of code for every target device.

The expansion of device acceleration in modern high-performance computing has had a profound impact on computational plasma physics. The paradigm shift from MPI to MPI+X has necessitated major updates to many plasma codes, such as GTC (now GTC-P) [78], EPOCH (where MPI+X was explored in the miniEPOCH miniapp) [79], XGC (with XGCm under development), and GITR (with GITRm under development) [80],

[81]. There is much to gain from device acceleration, particularly when the performance-critical kernels are well-suited to parallelization. The particle-in-cell method is one example, since particles can be simulated concurrently with essentially no communication between compute threads. This inherent parallelizability has been exploited with great success in many codes, including PConGPU [82] and EMPIRE-PIC [38].

The capabilities of modern high-performance computing architectures open up the possibility for much more complex multiphysics coupling. To that end, we present here hPIC2, the second iteration of an electrostatic, hybrid particle-in-cell code developed by the Laboratory of Computational Plasma Physics at the University of Illinois at Urbana-Champaign. hPIC2 exploits Kokkos for performance portability across a wide range of heterogeneous computing architectures. hPIC2 supports three different types of meshes: uniform meshes for one- and two-dimensional domains; block-structured non-uniform meshes furnished by the PUMImbbl mesh library; and unstructured meshes composed of simplex or tensor product elements for arbitrary two- and three-dimensional domains, which are facilitated by the MFEM finite-element method library [83]. Furthermore, hPIC2 optionally dynamically couples to RustBCA [84], which efficiently models plasma-material interactions.

Typically, sputtering and erosion are modeled with sputtering yields computed using analytical formulae [85]–[87] or precomputed and stored as tables with a code that utilizes the binary collision approximation [49], such as RustBCA, SRIM [88], SDTrimSP [89], Tridyn [90], and Fractal-Tridyn [91]. This approach is notably used by the EIRENE [92] and ERO [93], [94] code families. hPIC2 instead exploits RustBCA to directly perform BCA simulations during the main PIC simulation, using PIC particles impinging on a surface as the feed particles for BCA. While this approach is less performant than using a precomputed sputtering yield, it allows the simulation of PMI with materials for which no data exist, such as compounds, layered surfaces, surfaces with nontrivial morphology, and time-dependent surfaces. In addition, directly using BCA produces the correct energy and angle distributions for sputtered and reflected particles.

The organization of this chapter is as follows: in Section 3.1, we provide some programming details and innovations used in hPIC2; in Section 3.2, we provide results of some verification studies, showing that hPIC2 replicates analytic solutions; in Section 3.3, we investigate scaling and performance of hPIC2’s critical kernels on the Summit and Delta supercomputers; and in Section 3.4, we demonstrate the use of hPIC2 on some surrogate problems of potential interest. Note that some figures in this chapter are adapted from [41].

3.1 Implementation Details

In this section, we describe some details of the PIC algorithm as implemented in hPIC2. We also describe various software features.

hPIC2 uses the Message Passing Interface (MPI) for distributed-memory parallelism. The mesh is decomposed into roughly equal-sized partitions by assigning each MPI process a non-overlapping subsection of the mesh. hPIC2 solves the Poisson problem at each time step using hypre’s structured-grid system, or struct, interface [95] for finite-difference meshes and MFEM for finite-element meshes. By default, hPIC2 uses the Bi-CGSTAB method [96] for finite-difference meshes and a preconditioned conjugate gradient method with hypre’s BoomerAMG method as a preconditioner [97]. Details on the parallel scalability of hPIC2’s field solver are reported in Section 3.3.

Although the domain is decomposed for the field solve, each MPI process is responsible for particles on the entire domain in problems with uniform and MBBL meshes. This allows hPIC2 to be extremely tolerant to process failures during massively parallel simulations. Further description of the fault tolerance

mechanism is provided in Section 3.1.3. To aggregate charge density information needed by the field solver, `MPI_Reduce_scatter` is used to reduce charge density on nodes and distribute the reduction results to appropriate processes. Similarly, since every process requires the potential at every point in the mesh for the particle push, `MPI_Allgatherv` is used to gather the solution from each process, to every process. Examples of use for these MPI collective calls are displayed graphically in Figs. 3.1 and 3.2.

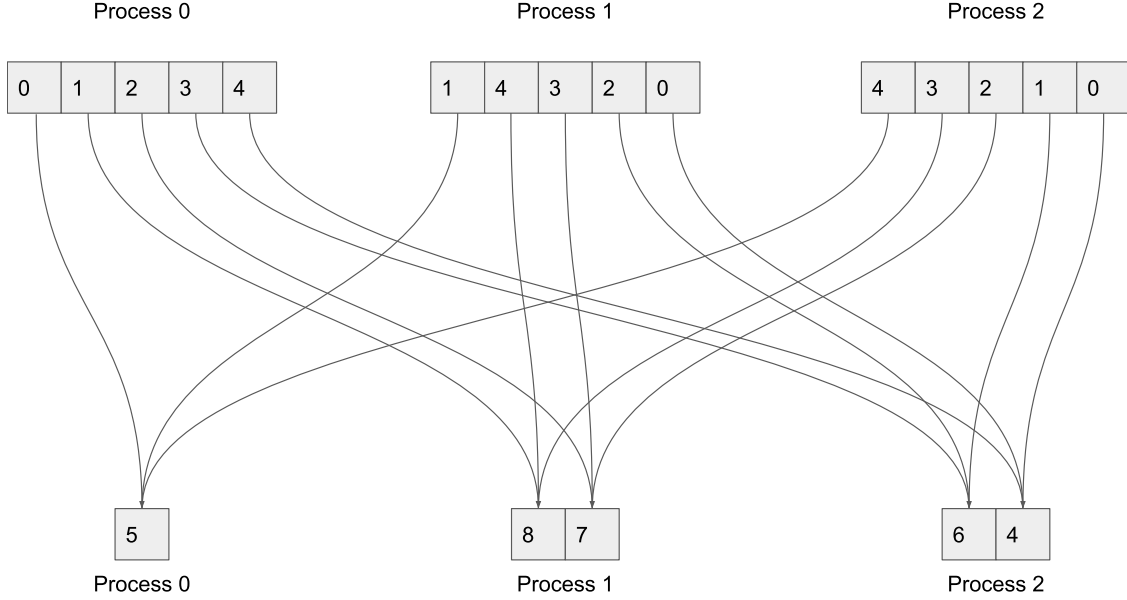


Figure 3.1: Use of `MPI_Reduce_scatter` with three processes, each of which has a send array of size 5. Process 0 receives a reduction result array of size 1, and the other two processes receive reduction result arrays of size 2. The send array corresponds to the charge density accumulated by particles owned by that process, whereas the reduction result array corresponds to the total charge density in portions of the mesh for which that process is responsible in the field solve.

hPIC2 uses the Kokkos [42] performance portability framework for shared-memory parallelism. With Kokkos, hPIC2 can achieve scalable performance on most parallel computing architectures with a single source code. To do this, Kokkos heavily leverages C++ template metaprogramming and enforces a unified interface for parallelizable for loops, reductions, and prefix scans, which it can submit to any of its available parallel programming model backends. Kokkos also provides a reference-counted data container, called a View, whose memory layout and properties are determined at compile time for optimal data access performance.

Importantly for the particle charge scatter step in a PIC code, thread-safe array reductions are abstracted with the Kokkos ScatterView object. The ScatterView is a compile-time polymorphic object that represents thread-local copies of an array for parallel kernels on host (such as with OpenMP or POSIX threads), and atomic array access for parallel kernels on device, optimizing write performance depending on the architecture while avoiding race conditions.

Similarly, the Kokkos Views containing the values of the electric and magnetic fields at grid points are imbued with the RandomAccess memory trait. When compiled with CUDA, this enables texture fetches for read-only access. Since the texture cache has a shorter cache line than standard caches, non-contiguous memory access is typically faster. This is important for the gather step in PIC, since the particle push kernel randomly accesses data from the field arrays.

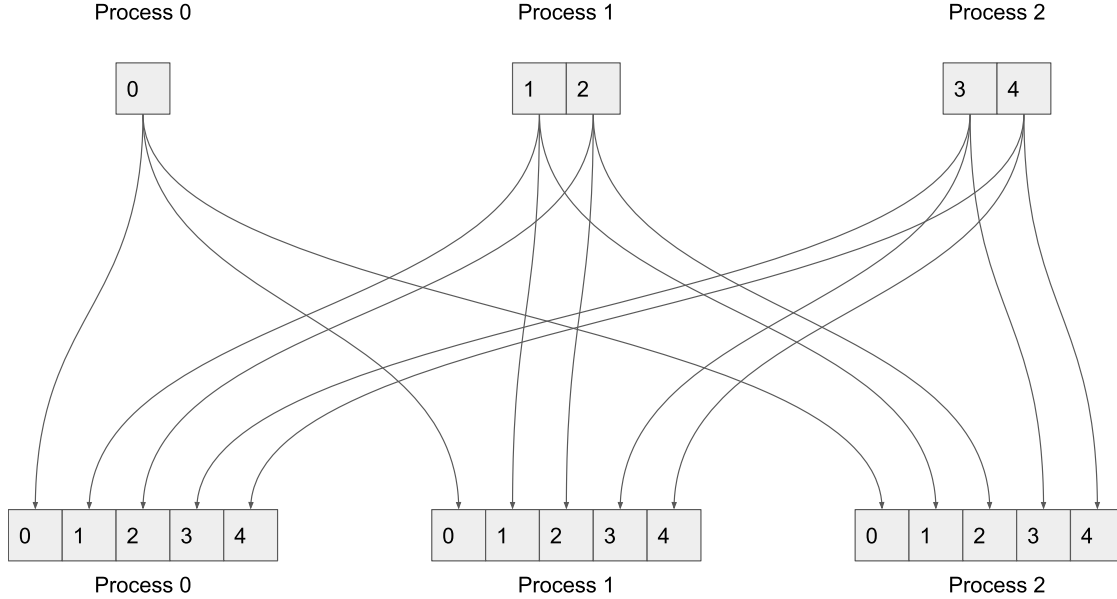


Figure 3.2: Use of MPI_Allgatherv with three processes, each of which has a receive array of size 5. Process 0 sends an array of size 1, and the other two processes send arrays of size 2. The send array corresponds to the field solution in the portion of the mesh for which that process is responsible in the field solve, whereas the receive array corresponds to the potential solution on the entire mesh, which is needed by each process.

hPIC2 can optionally exploit CUDA-aware MPI implementations. If MPI is detected as CUDA-aware at compile-time and hPIC2 is built atop a CUDA-enabled Kokkos installation, hPIC2 passes device memory pointers to MPI calls. Although the MPI standard does not specify whether memory is first transferred to host before it is collected into a communication buffer, certain implementations utilize GPUDirect technology to create communication buffers directly between devices. GPU-aware hypre installations are treated similarly. Thus, when properly configured with Nvidia hardware, hPIC2 almost never spends computing resources to copy memory from the device to the host.

hPIC2 reads input decks in the TOML format [98], which is a minimal configuration file format that can be unambiguously mapped to a hash table. The format is human-readable, and many languages have library support to enable scripts to generate TOML files. hPIC2 uses the toml11 [99] C++ library to read and parse TOML files.

3.1.1 Dynamic coupling to RustBCA

hPIC2 is intended to accurately model near-surface plasmas and plasma-material interactions at domain boundaries. To that end, hPIC2 can couple in-memory to RustBCA [84], a code that simulates the effects of energetic atom and ion impacts on materials using multiple BCA methods. RustBCA provides Python, Fortran, and C/C++ bindings for a subset of its capabilities, effectively allowing external codes to run simple simulations using a standard function call. Specifically, a locally atomically flat, homogeneous target with user-defined, arbitrary composition is used in the simulation of plasma-wall interactions.

RustBCA acts as a particle boundary condition in hPIC2 for particles of a given species at a given boundary. The user is able to specify parameters of the boundary such as binding energies, density, and composition, which realistically affect the sputtering and reflection at the boundary. During the particle push

phase of a time step, the velocities of all particles of that species that impact that boundary are recorded. During the collision phase, hPIC2 performs a RustBCA simulation using the recorded particle velocities, transformed to the local surface normal coordinates, and RustBCA returns a list of output particles, which may include both reflected and sputtered particles that can be transformed back to the global coordinates upon entering the plasma. RustBCA also simulates the implantation of particles into the material, which could be used for dynamic material simulations, but this information is not currently used. A sketch of the RustBCA coupling workflow is provided in Fig. 3.3.

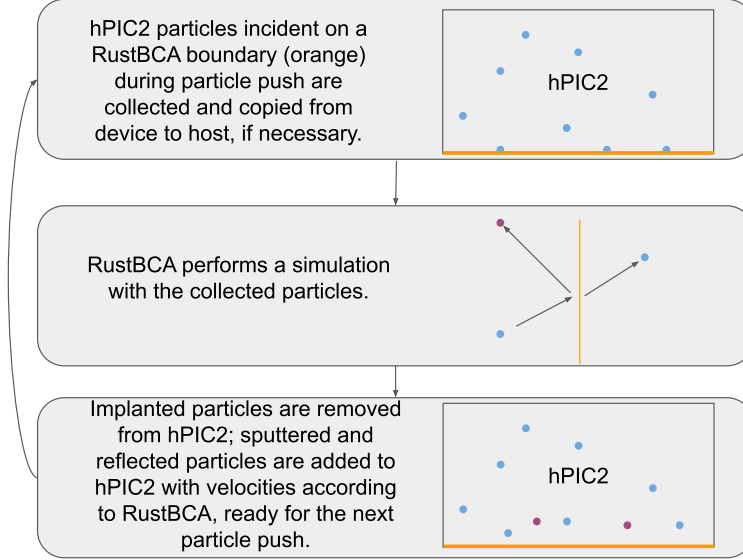


Figure 3.3: A sketch of the RustBCA coupling workflow.

Sputtered and reflected particles are added to the hPIC2 simulation uncharged. Although a small fraction (several percent) of light ions with high exit velocities (hundreds of eV) are expected to reflect as charged particles, depending on the trajectory in the material and the surface condition, these are neglected due to the relatively low incident energies in plasma-wall interaction simulations[100]. Since particles added by the RustBCA coupling are neutral, an ionization process is required to close the physical model. Monte Carlo methods are typically used to model interactions in particle-based simulations, and have found particular popularity in plasma edge modeling codes [101]–[104]. While direct simulation Monte Carlo algorithms are used for collisions between two particle-based species [104], [105], the so-called Monte Carlo collision (MCC) model [101], [106], [107] collides particles with continuum-based species such as fluids or backgrounds. Our MCC algorithm was described in Section 2.2.6.

Note that, although RustBCA runs on the host, hPIC2 itself can still exploit device parallelism. Particle data is copied to and from the device as necessary.

3.1.2 PUMImbbl meshes

PUMImbbl is an accelerator-enabled, block-structured non-uniform rectilinear mesh library developed under the parallel unstructured mesh infrastructure (PUMI) to support hPIC2 simulations on graded and anisotropic meshes. The multi-block boundary layer (MBBL) mesh works by splitting the plasma domain into a number of

submeshes or blocks. In 1D, the MBBL mesh is defined based on uni-dimensional blocks. Each uni-dimensional block can deploy a specific mesh type as necessary, such as a uniform mesh type and a geometrically graded boundary-layer mesh type, biased either at the positive or negative end. In higher dimensions, the MBBL mesh is constructed as an orthogonal tensor product of uni-dimensional blocks defined along each direction. The goal is to resolve high gradients, such as in plasma sheaths, with a boundary layer mesh block with an appropriate bias, and progressively grade the mesh resolution away from this region into the bulk plasma region, i.e., increase the coarseness of the mesh. This allows one to discretize the bulk region with a much coarser uniform mesh block, thereby effectively discretizing the plasma domain with a small number of mesh elements. On the other hand, in a fully uniform mesh the mesh resolution is dictated globally by a small portion of the plasma domain such as a local region with a high-gradient sheath. Another important feature of the MBBL mesh is its compact memory footprint due to implicit storage. Each submesh or block is parametrized and stored in the form of a set of parameters, and any property of the mesh required during the PIC simulations is computed on the fly. Fig. 3.4 compares an example of a PUMImbbl and uniform mesh of the same 2D domain, demonstrating how the size of the mesh can be tuned to resolve regions of interest.

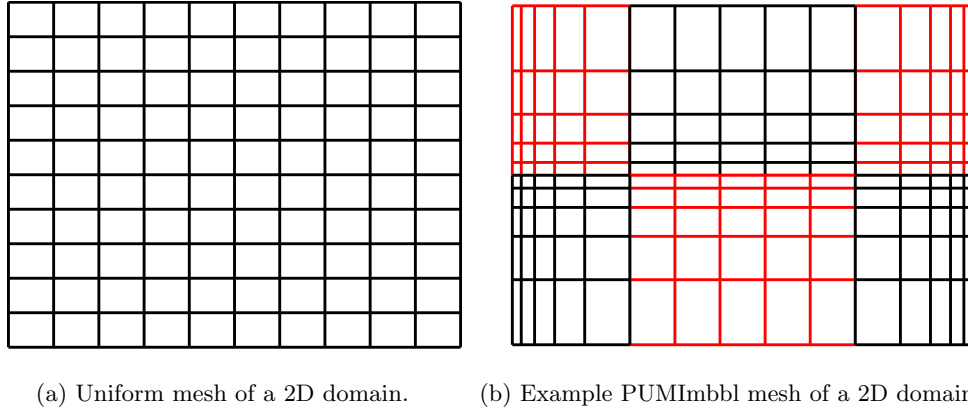


Figure 3.4: Comparison of an example of a PUMImbbl mesh and a uniform mesh of the same 2D domain.

The PUMImbbl library supports all necessary PIC operations, which are designed to closely match the performance of a uniform mesh (on both host and device, e.g., GPUs). A global particle search is only performed on a MBBL mesh when particles are globally initialized. Once particles are located, a mesh adjacency-based particle tracking is performed within each particle push step. To do this, the submesh/block and cell/element information is stored for each particle and updated as a particle moves. Efficient implementation of gather/scatter operations on the MBBL mesh is also made possible by the block and element information stored for each particle. In addition, the field solve on MBBL mesh is generalized to include the non-uniform/graded stencil, e.g., the one associated with the Poisson equation. Ongoing work includes support for non-convex rectilinear plasma domains.

3.1.3 Fault tolerance

As described above, for finite difference-based meshes, all MPI processes in a distributed hPIC2 simulation are responsible for pushing particles in the entire domain. This implies that each process has sufficient mesh information to reconstruct the mesh by itself and that the loss of a process leads only to a loss in particle fidelity. Hence, in principle, the entire simulation can be restarted from the information owned by a single process, albeit at reduced particle fidelity and obviously without the ability to distribute the field solve

workload. It is far more common for a single process to fail at a time, in which case the loss in fidelity is minimal, and the field solve can still be distributed among a significant process workforce.

The capability to detect and recover from faults is furnished by the proposed User Level Fault Mitigation (ULFM) extension to the MPI standard [108]. ULFM adds a small set of error codes and interfaces to the MPI standard. The most relevant additions are the `MPHX_ERR_PROC_FAILED` error code, returned when a process failure is detected during communication, and the `MPHX_Comm_shrink` interface, which creates a new communicator excluding processes that were detected to have failed.

hPIC2 maintains an MPI error handler during program runtime. If hPIC2 was compiled against an MPI implementation without ULFM, MPI errors cause the program to terminate, as is the default behavior. With ULFM, however, the error handler responds to detected failures during MPI communication by executing the following four steps:

1. The global communicator is immediately shrunk to exclude dead processes.
2. The weights of macroparticles owned by remaining processes are augmented to compensate for the loss of macroparticles on dead processes.
3. The mesh decomposition is reevaluated, and the mesh partition is reassigned to remaining processes.
4. A global flag is raised, indicating that a failure occurred during the time step. Just before the charge density is reduced and scattered to processes for the field solve, the charge density is reset and recomputed based on the state of the plasma at that instant in time. This is required if a process failure occurred after the particle charge scatter, since otherwise, the charge density arrays will contain charge accumulated from particles before their weight was augmented.

The advantage of augmenting particle weights rather than populating new particles on each remaining process is that the particle workload per process remains consistent, at the cost of a reduction in statistical fidelity. Thus, the only slowdowns incurred by a process failure are due to the overhead of executing the four failure recovery steps and the reduced process workforce for the distributed field solve. In massively parallel simulations such as those expected at the exascale, the loss of a few processes is therefore not expected to significantly affect wall time. This means that only catastrophic failures will cause the simulation to exceed expected wall times, which is of particular use to users on computing platforms with queued batch schedulers.

3.1.4 Output diagnostics

hPIC2 is capable of outputting a number of useful diagnostics. The electric potential, electric field, and magnetic field can all be written to disk, along with the total electrostatic energy in the problem. The fields are saved using the HDF5 library [109] in the VTKHDF format, for which VTK provides scripting and post-processing support [110]. VTKHDF files are supported by the latest versions of the ParaView visualization software [111], [112].

hPIC2 is also able to compute moments of the particle distribution of arbitrary order. We define the kinetic moment of a species characterized by a single-particle distribution $f(\mathbf{x}, \mathbf{v}, t)$ as

$$M_{ijk}(\mathbf{x}, t) = \int_{\mathbb{R}^3} m v_x^i v_y^j v_z^k f(\mathbf{x}, \mathbf{v}, t) d\mathbf{v}, \quad (3.1)$$

where i, j, k are nonnegative integers. Note that, with this definition, the density of the species is equal to M_{000} and the momentum density is equal to $(M_{100}, M_{010}, M_{001})^t$. The moments can also be computed in the

rest frame of the species, in which case (3.1) is modified to become

$$M_{ijk}^0(\mathbf{x}, t) = \int_{\mathbb{R}^3} m(v_x - u_x(\mathbf{x}, t))^i (v_y - u_y(\mathbf{x}, t))^j (v_z - u_z(\mathbf{x}, t))^k f(\mathbf{x}, \mathbf{v}, t) d\mathbf{v}, \quad (3.2)$$

where $\mathbf{u} = (M_{100}, M_{010}, M_{001})^t / M_{000}$ is the average or bulk velocity of the species. (3.1) is evaluated for a species represented by full-orbit PIC by substituting (2.10), after which we have

$$M_{ijk}(\mathbf{x}, t) = \sum_{\alpha=1}^N m w_{\alpha}(t) v_{\alpha,x}^i(t) v_{\alpha,y}^j(t) v_{\alpha,z}^k(t) S(\mathbf{x}, \mathbf{x}_{\alpha}(t)), \quad (3.3)$$

and analogously for (3.2).

In addition to the moments of the particle distribution, hPIC2 can compute the fluxes of moments. Define

$$D_{ijk,\hat{n}}(\mathbf{x}, t) = \int_{\{\mathbf{v} \in \mathbb{R}^3 | \mathbf{v} \cdot \hat{n} > 0\}} m(\mathbf{v} \cdot \hat{n}) v_x^i v_y^j v_z^k f(\mathbf{x}, \mathbf{v}, t) d\mathbf{v}, \quad (3.4)$$

which is interpreted as the flux of the moment M_{ijk} from one side of a plane located at position $\mathbf{x}$ with unit normal $\hat{n}$ to the other. Note that $D_{000,\hat{n}}$ returns the one-sided mass flux through such a plane. As before, this is evaluated for a species represented by full-orbit PIC by substituting (2.10), after which we have

$$D_{ijk,\hat{n}}(\mathbf{x}, t) = \sum_{\substack{\alpha=1 \\ \mathbf{v}_{\alpha}(t) \cdot \hat{n} > 0}}^N m w_{\alpha}(t) (\mathbf{v}_{\alpha}(t) \cdot \hat{n}) v_{\alpha,x}^i(t) v_{\alpha,y}^j(t) v_{\alpha,z}^k(t) S(\mathbf{x}, \mathbf{x}_{\alpha}(t)), \quad (3.5)$$

which can be rapidly evaluated at mesh nodes on device.

hPIC2 is also able to output the ion energy-angle distribution (IEAD) of full-orbit PIC particles incident on an absorbing boundary. hPIC2 computes a 2D histogram of particles incident on the boundary whose axes are the angle of incidence and the kinetic energy of the particle. The user can specify both the maximum kinetic energy in the histogram and the number of bins used on the energy axis.

For each of these outputs, the user can specify a time step interval for diagnostic computation and output. The user can also override this interval to force output to occur at the first or last time step. This can significantly reduce simulation times, as writing to disk is an extremely time-consuming operation.

3.2 Verification

hPIC2 is capable of simulating a wide range of plasma phenomena and is designed to perform optimally on modern high-performance supercomputers. However, such large codebases are prone to programming errors, or bugs, particularly when intended for deployment on multiple architectures. To address this potential issue, we have performed extensive verification and validation studies on hPIC2, which serve to prove that the code is functioning as intended. Broadly, verification involves comparing code output to known analytic solutions, whereas validation involves comparing code output to experimental data [113]. In this section, we present the results of several verification studies, which attempt to exercise the majority of the core physics kernels in hPIC2.

3.2.1 FDM sheath potential in a plasma slab

This problem investigates the formation of an unmagnetized sheath in a plasma slab between two grounded walls. Uniform, volumetric ionization maintains a constant average ion density. The expected electric potential at the midpoint of the domain is given by

$$\phi_{\text{midpoint}} = \frac{kT_e}{2e} \ln \left(\frac{m_i/m_e}{2\pi Z^2 \left(1 + \frac{T_i}{T_e}\right)} \right), \quad (3.6)$$

where T_e is the temperature of the electrons, m_e is the mass of electrons, T_i is the temperature of the ions, m_i is the mass of ions, and Z is the charge number of the ions [7].

This problem considers a plasma consisting of hydrogen ions with an initial number density of 10^{20} m^{-3} and temperature of 10 eV. The electrons are also held at a temperature of 10 eV. In order to model a plasma that is an order of magnitude larger than the anticipated sheath size, we choose the problem domain to be a square with side length $L = 120\lambda_D$ with Debye length $\lambda_D = \sqrt{\varepsilon_0(kT_e + kT_i)/(n_0 e^2)} \approx 3.32 \times 10^{-6} \text{ m}$, where n_0 is the initial ion number density, and T_e and T_i are the temperatures of electrons and ions, respectively. Fig. 3.5a shows a sketch of this domain. This domain is meshed by partitioning the domain into 120 square elements in each direction, for a total of 14 400 elements. With these plasma parameters, the ion acoustic transit time is computed as $\tau_{\text{tr}} = L/v_{\text{th}} \approx 9.11 \times 10^{-9} \text{ s}$, where $v_{\text{th}} = \sqrt{2kT_i/m_i}$. We choose a time step of $\Delta t = \Delta x/3v_{\text{th}} \approx 2.53 \times 10^{-11} \text{ s}$ to allow for proper thermal sampling, and simulate 495 time steps, which corresponds to $11\tau_{\text{tr}}/8$.

Ions are treated using full orbit PIC and electrons are treated using the Boltzmann electron approximation. The boundaries at the x -extrema of the domain are electrically grounded and remove incident ions from the problem, which are then repopulated uniformly throughout the mesh with temperature 10 eV for the subsequent time step. The boundaries at the y -extrema of the domain impose homogeneous Neumann boundary conditions on the potential and perfectly reflect incident ions.

A base number of macroparticles per element, or particles per cell (PPC), of 100 is chosen, then progressively doubled for several levels of refinement up to a PPC of 1600. The potential is recorded at each of the nodes that line on the centerline of the domain at $x = 60\lambda_D$. The L_2 error norm of the potential along the centerline is plotted as a function of refinement level in Fig. 3.5e. The plot shows that hPIC2 converges to the analytic solution at nearly order 1/2, which is expected for particle count refinement.

For each refinement level, we also record the potential at each of the nodes that lie along the line $y = 60\lambda_D$. These are plotted in Fig. 3.5d, which clearly shows significant noise reduction with increasing PPC, as desired. A plot of the potential for the most refined case at the last time step is also provided in Fig. 3.5b.

3.2.2 FEM sheath potential at midpoint

The verification study in Sec. 3.2.1 is repeated using the finite element method. This version of the problem is nearly identical except that each of the square elements is split into two triangles, for a total of 28 800 triangular elements, and a base level of 50 PPC is progressively doubled up to 800 PPC.

The analytic solution for the potential at the center of the sheath is again given by (3.6). The potential is recorded at each of the nodes that lie on the centerline of the domain at $x = 60\lambda_D$. The L_2 error norm of the potential along the centerline is plotted as a function of refinement level in Fig. 3.5f. The plot shows that hPIC2 achieves good order 1/2 convergence, as is expected for particle count refinement. A plot of the

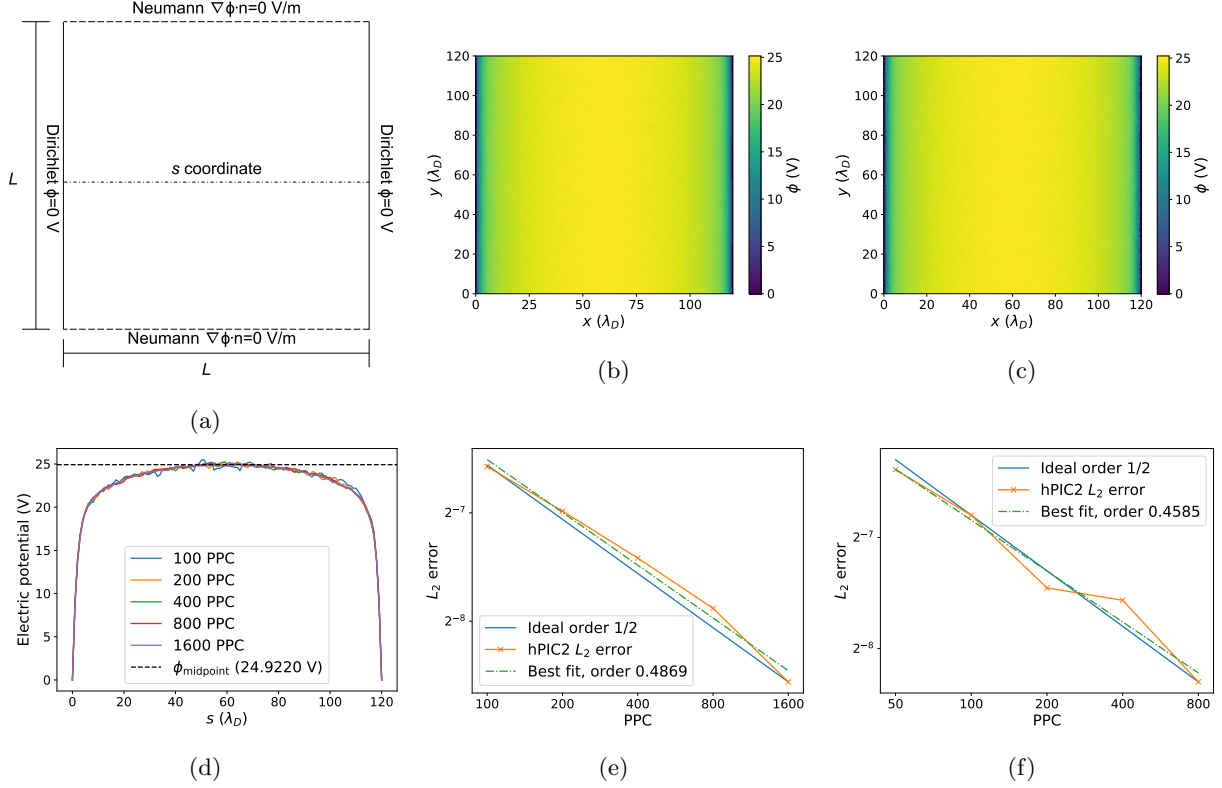


Figure 3.5: Plasma sheath in a plasma slab: (a) Sketch of the domain used in the verification problem; (b) Plasma potential at equilibrium, in the most refined version of the FDM sheath verification problem; (c) Plasma potential at equilibrium, in the most refined version of the FEM sheath verification problem; (d) Plasma potential along s coordinate for each refinement level, exhibiting noise reduction with increasing number of PPC; (e) L_2 error norm of the potential along the centerline of the domain for the FDM sheath potential verification problem. (f) L_2 error norm of the potential along the centerline of the domain for the FEM sheath potential verification problem.

potential for the most refined case at the last time step is also provided in Fig. 3.5c.

3.2.3 Two-stream instability

A plasma that is not in thermal equilibrium but whose forces are balanced is in an unstable equilibrium. Free energy from thermal non-equilibrium can excite plasma wave instabilities that drive the plasma toward thermal equilibrium. Streaming instabilities are a special case of instability whereby the thermal non-equilibrium is caused by relative drift velocities between two plasma species. In this problem, we consider the two-stream instability, which occurs when two populations of cold electrons, each with number density $n_0/2$ and equal but opposite velocities $\mathbf{v}_0 = \pm v_0 \hat{x}$, interact electrostatically with each other and a stationary ion background with number density n_0 . The dispersion relation for the two-stream instability can be derived using linearized Vlasov-Poisson theory as

$$1 = \frac{1}{2} \omega_p^2 \left[\frac{1}{(\omega - kv_0)^2} + \frac{1}{(\omega + kv_0)^2} \right], \quad (3.7)$$

where ω and k are respectively the angular frequency and wavenumber of the unstable electrostatic wave, $\omega_p \approx \sqrt{n_e e^2 / (m_e \epsilon_0)}$ is the plasma frequency of electrons with number density n_e and mass m_e , and v_0 is the

initial drift speed of the electron streams [13].

We model this phenomenon in a one-dimensional periodic domain. The primary wave of interest has wavelength equal to the length L of this domain. The roots ω of (3.7) with $k = 2\pi/L$ are generally complex. A positive imaginary part corresponds to a growing wave instability. The growth rate of the instability is then given by

$$\gamma = \Im(\omega) = \Im \left(\sqrt{\frac{1}{2}(\omega_p^2 + 2k^2 v_0^2)} - \frac{1}{2}\omega_p \sqrt{(\omega_p^2 + 8k^2 v_0^2)} \right), \quad (3.8)$$

where $\Im$ extracts the imaginary part of a complex number.

For this problem, we choose $n_0 = 10^{16} \text{ m}^{-3}$ and $v_0 = 3.2 \times 10^5 \text{ m/s}$. A time step of length $\Delta t = 2\pi/\omega_p 120 \approx 9.28 \times 10^{-12} \text{ s}$ is used to resolve the plasma oscillation period. 1800 time steps, or 15 plasma oscillation periods, are simulated. A spatial grid consisting of 400 elements of size $\Delta x = v_0 \Delta t \approx 2.97 \times 10^{-6} \text{ m}$ is used. Each electron population uses 50 000 macroparticles.

The total electrostatic energy in the domain is computed as

$$U_E = \frac{1}{2} \epsilon_0 \int_{\Omega} d\mathbf{x} \nabla \phi \cdot \nabla \phi \quad (3.9)$$

and recorded at each time step. The growth rate is computed from this output by identifying the longest period of time when $\log U_E$ is approximately linear and fitting a line to the data. The electrostatic energy is plotted as a function of time in Fig. 3.6. This plot is overlaid with the best-fit line and a line representing the analytic growth rate computed in (3.8). The plot shows that the computed growth rate differs from the analytic growth rate with an error of 2%. Fig. 3.7 displays snapshots of the particles in phase space, showing the formation and propagation of the expected phase-space vortex typical of a two-stream instability.

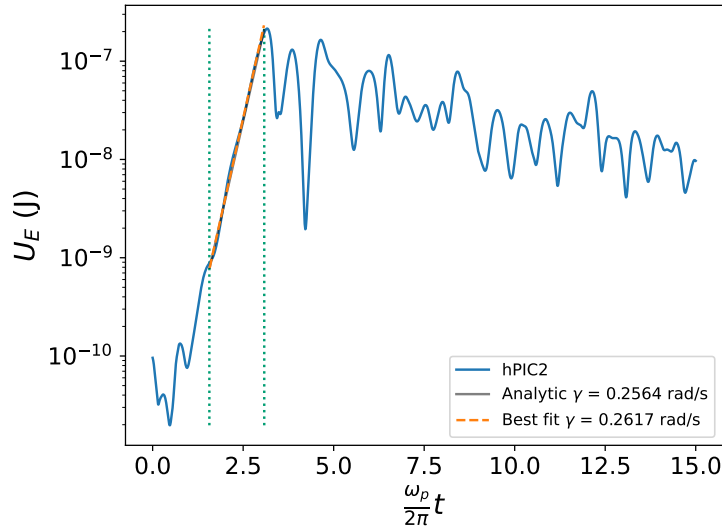


Figure 3.6: Electrostatic energy of the two-stream instability verification test alongside the lines representing the best-fit and analytic growth rates.

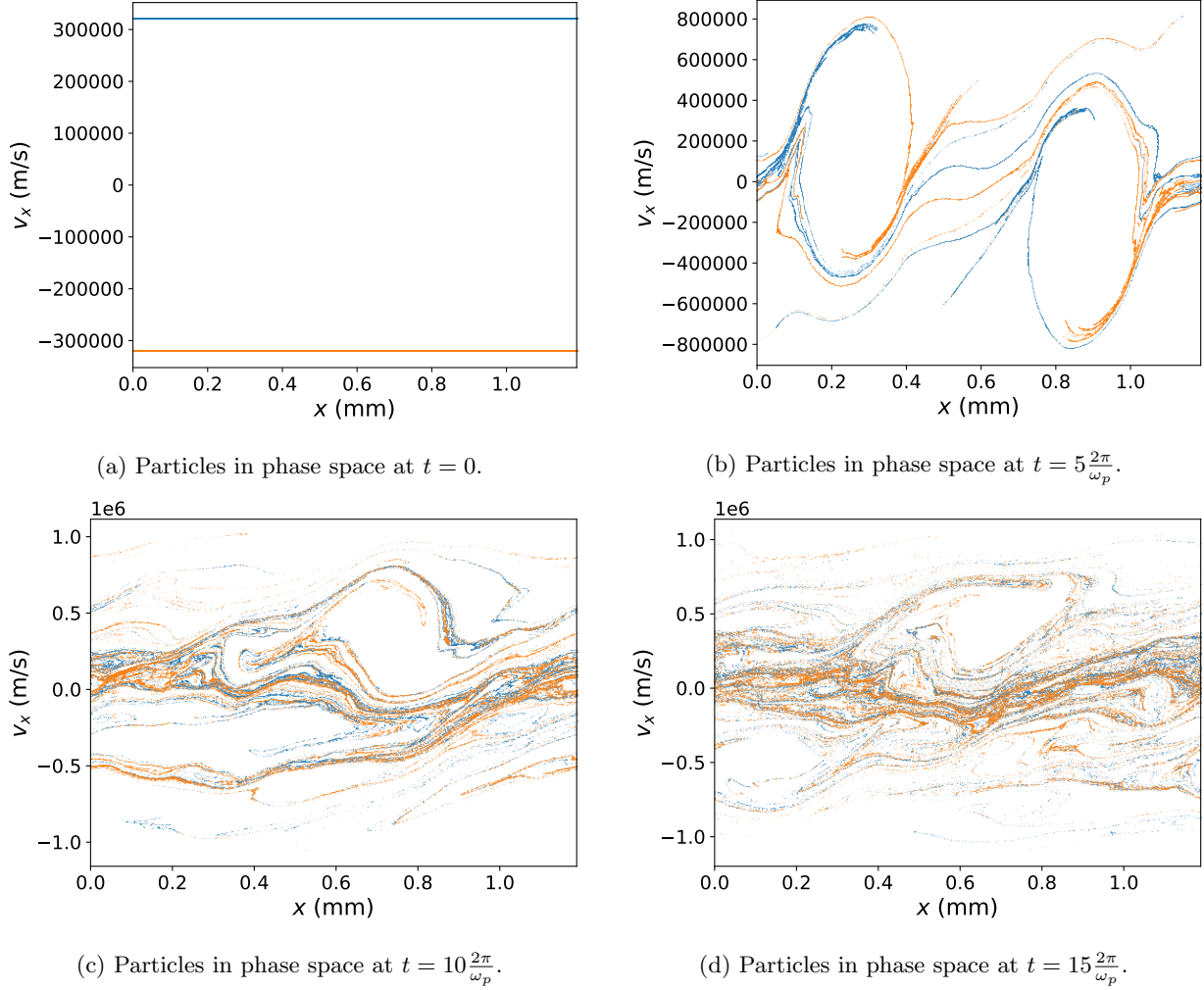


Figure 3.7: Scatter plot of particles in phase space throughout the simulation. Particles with positive initial velocity shown in blue; particles with negative initial velocity shown in orange.

3.2.4 Turner MCC benchmarks

It is notoriously difficult to derive analytic solutions for plasma problems involving collisions between a dense, high-pressure background gas and a dilute, low-pressure plasma, which is the regime in which MCC is most useful [114]. The analytic solutions that do exist are typically considered not to sufficiently exercise or test the full capabilities of MCC codes. As a result, developers of MCC software are generally content with benchmark tests, which compare code output between different codes. A suite of benchmark tests was developed by Turner et al. [115], and a comparison of hPIC2 output with selected benchmarks from this suite is presented in this section. Although they are not true verification tests, they are presented here as a demonstration of hPIC2's correctness.

The Turner benchmark problems model a capacitive, radio-frequency (RF) discharge of a helium plasma under various low-pressure conditions. The simulations are performed on one-dimensional uniform meshes with boundaries that absorb incident particles on both sides. Dirichlet boundary conditions are imposed on

the potential at the boundaries such that the voltage difference between them is given by

$$\Delta V(t) = V_0 \sin(2\pi ft), \quad (3.10)$$

where V_0 is the voltage amplitude and f is the frequency. Electrons and helium ions are modeled with PIC and the neutral helium is assumed to be a uniform Maxwellian background. MCC is used to model ion-neutral and electron-neutral collisions with cross-sections taken from [115]. Ion-electron collisions are neglected. Electrons and ions are initialized with a Maxwellian velocity distribution and a uniform spatial distribution.

We perform hPIC2 simulations of test cases 1 through 3 and compare hPIC2 output to the output of the extended calculation using code E [115]. The quantity of interest is the ion number density averaged over the final 32 RF cycles of the simulation. A table of the parametrizations for each test case is reproduced in Table 3.1.

Parameter	Variable name (units)	Case 1	Case 2	Case 3
Electrode separation	L (cm)	6.7	"	"
Neutral background density	(10^{20} m^{-3})	9.64	32.1	96.4
Neutral temperature	(K)	300	"	"
Frequency	f (MHz)	13.56	"	"
Voltage amplitude	V_0 (V)	450	200	150
Simulation time	(f^{-1})	1280	5120	5120
Electron mass	(10^{-31} kg)	9.109	"	"
Helium mass	(10^{-27} kg)	6.67	"	"
Initial plasma number density	(10^{14} m^{-3})	2.56	5.12	5.12
Initial electron temperature	(K)	30 000	"	"
Initial ion temperature	(K)	300	"	"
Initial PPC		512	256	128
Cell size	(L)	1/128	1/256	1/512
Time step size	(f^{-1})	1/400	1/800	1/1600

Table 3.1: Parametrizations for Turner benchmark test cases 1 through 3.

The averaged ion density is plotted for all three test cases as a function of space for both hPIC2 and code E in Fig. 3.8, along with a zoomed-in view of the ion density at the center of the domain for case 3. The hPIC2 results qualitatively match those of code E, which indicates that hPIC2's MCC kernel is functioning as intended.

3.2.5 Bobylev-Krook-Wu thermalization

For a particular choice of collision kernel $\mathcal{C}$, and in the absence of force fields, there is an analytic, zero-dimensional solution to (1.3). Derived independently by Bobylev [116] and Krook and Wu [117], the Bobylev-Krook-Wu (BKW) solution models the thermalization of a distribution whose constituent physical particles collide elastically with a cross-section given by

$$\sigma(\mathbf{g}) = \frac{\kappa}{g}, \quad (3.11)$$

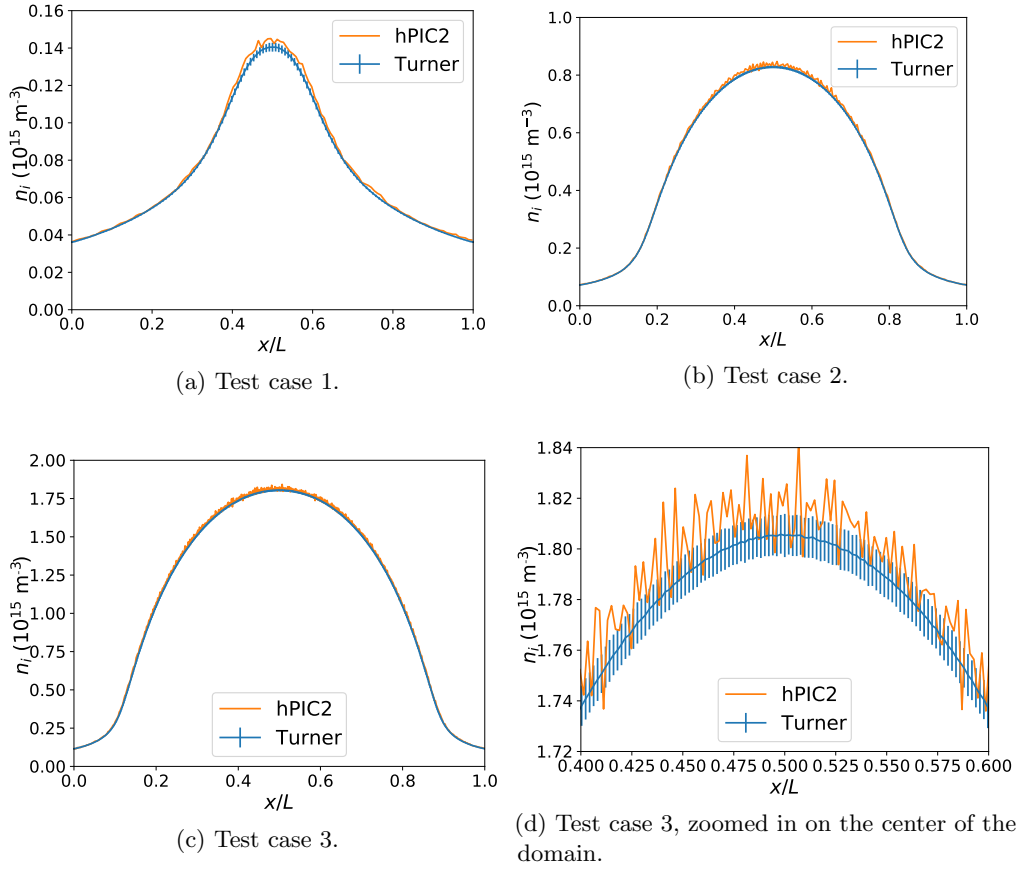


Figure 3.8: Ion density in the Turner MCC benchmark test cases from hPIC2 and code E (labeled as “Turner”), averaged over the last 32 RF cycles.

where κ is some positive real coefficient. This cross-section is sometimes called the Maxwell molecule cross-section. Let $\tau = 1/(n\kappa)$ be the mean free time and

$$K(t) = 1 - \exp\left(-\frac{t}{6\tau}\right), \quad (3.12)$$

where n is the constant number density of the species. The BKW solution is then given by

$$f_{\text{BKW}}(\mathbf{v}, t) = \frac{n}{2} \left(\frac{m}{2\pi K k T}\right)^{3/2} \left(\frac{5K-3}{K} + \frac{1-K}{K} \frac{mv^2}{kT}\right) \exp\left(-\frac{mv^2}{2K k T}\right), \quad (3.13)$$

where T is the constant kinetic temperature of the species. Note that this is nonnegative only when $K \geq 3/5$, so the analytic solution is invalid for $t < t_0 \equiv 6\tau \ln \frac{5}{2}$.

There is some ambiguity in how to represent the distribution from a PIC-DSMC simulation. Instead, we can compare moments from hPIC2 to the moments of the BKW solution. The zeroth-, first-, and second-order moments of the BKW solution are uninteresting, since they are constant in time due to conservation of mass, momentum, and energy. Similarly, the third-order moments are zero due to the symmetry of the distribution. Therefore, the lowest-order moments that are interesting to compare are the fourth-order moments. The

fourth-order moment considered in this section is

$$M_4[f] = \int_{\mathbb{R}^3} m v^4 f \, d\mathbf{v}. \quad (3.14)$$

The corresponding fourth-order moment of the BKW solution is given by

$$M_4[f_{\text{BKW}}] = 15mn \left(\frac{kT}{m} \right)^2 (2 - K)K, \quad (3.15)$$

which can be directly compared to the fourth-order moment of the PIC-DSMC simulation as computed with (3.3).

As hPIC2 does not currently support zero-dimensional domains, we simulate a one-dimensional domain with periodic boundary conditions. A goal of this verification test is to test hPIC2's ability to model collisions between particles of different species and different weights. To that end, we split the gas into two “species:” one with number density $n/3$, and the other with number density $2n/3$. In addition, we would like to test hPIC2's ability to model multiple collision processes between species. To do this, the BKW collision is split into two collisions, each with κ halved. The collective effect of these two collisions should be exactly the same as the original BKW collision, but this splitting stresses the DSMC algorithm's ability to handle more than one collision at a time.

The two species are initialized with a velocity distribution according to $f_{\text{BKW}}(\mathbf{v}, t_0)$ and equal numbers of macroparticles N . The problem is parametrized with $n = 10^{20} \text{ m}^{-3}$, $T = 300 \text{ K}$, $\kappa = 10^{-12} \text{ m}^3/\text{s}$, $\Delta t = \tau/5$, $m = 1.67 \times 10^{-27} \text{ kg}$, and a termination time of $t_0 + 25\tau$. M_4 is computed every 5 time steps, summed over the two species, and averaged over the domain. The problem is initially simulated with $N = 20000$, then successively doubled for several levels of refinement up to $N = 160000$.

The results of the verification study are shown in Fig. 3.9. The average M_4 qualitatively matches the analytic solution well. Furthermore, the relative L_2 error norm of the average M_4 converges to the analytic solution at nearly order 1/2, as expected. This indicates that hPIC2's DSMC kernel is functioning as intended.

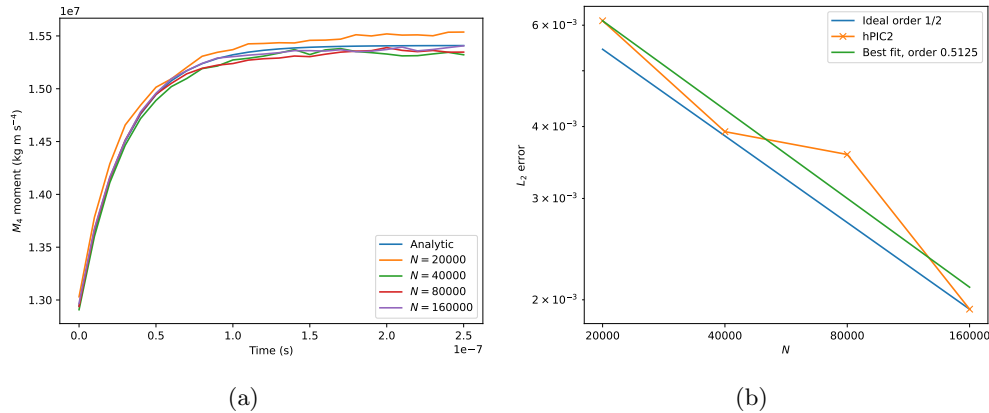


Figure 3.9: Results of the BKW verification study: (a) Average M_4 plotted as a function of time for each refinement level, compared to the analytic solution; (b) Relative L_2 error norms of the average M_4 for each refinement level, along with a best fit line indicating the approximate convergence order. Note the log scale on both axes.

3.2.6 Sod shock tube

The Sod shock tube is a classic test problem for hydrodynamic codes, named after the author who first introduced it [118]. The problem is essentially a one-dimensional Riemann problem for (1.18) using the ideal gas equation of state given by (1.21), with a certain choice of initial conditions, and in the absence of collisions and Lorentz forces. In particular, the initial condition is given by

$$\begin{pmatrix} nm \\ nm u \\ nm E \end{pmatrix} (x, 0) = \begin{cases} \begin{pmatrix} 1 \\ 0 \\ \frac{1}{\gamma-1} \end{pmatrix} & \text{if } x \leq 0, \\ \begin{pmatrix} \frac{1}{8} \\ 0 \\ \frac{1}{10(\gamma-1)} \end{pmatrix} & \text{if } x > 0, \end{cases} \quad (3.16)$$

and the specific heat ratio γ is taken to be 7/5.

A principal goal of the Sod shock tube problem is to test whether a fluid solver is stable in regimes with discontinuities in the solution and does not introduce spurious oscillations or other artifacts. The Sod shock tube problem therefore not only stresses the DG discretization itself, but also the stabilization algorithm and Riemann solver used for the numerical flux in (2.41). This is particularly important for hPIC2, since, as described in Section 2.3, the DG methods used in hPIC2 are prone to Gibbs phenomena at discontinuities, which mandates the use of some variety of stabilization. In hPIC2, we primarily use a slope limiter to achieve this purpose [66].

The solution to the Sod shock tube problem is separated into five distinct zones. The density is plotted as a function of space at some arbitrary time $t > 0$ in Fig. 3.10 to illustrate these zones, along with dashed vertical lines representing the demarcations between them. The solution can be derived by considering the characteristics propagating from the origin and the Rankine-Hugoniot jump conditions at the discontinuities arising from them [27], [28]. For brevity, the solution shall simply be stated here.

The original Sod paper considers the solution for four quantities of interest: the density nm , which, throughout the remainder of this section, we shall denote by ρ and should not be confused with charge density; the bulk velocity u ; the scalar pressure p ; and the specific energy $E = nmE/nm$. Since the specific energy can always be computed as

$$E = \frac{p}{\rho(\gamma-1)} + \frac{1}{2}u^2, \quad (3.17)$$

we report only the density, bulk velocity, and pressure. Define the speeds of sound for the left and right states respectively as

$$c_L = \sqrt{\frac{\gamma p_L}{\rho_L}}, \quad (3.18)$$

$$c_R = \sqrt{\frac{\gamma p_R}{\rho_R}}, \quad (3.19)$$

where $p_L = 1$ is the initial pressure of the left state, $\rho_L = 1$ is the initial density of the left state, $p_R = 1/8$ is the initial pressure of the right state, and $\rho_R = 1/10$ is the initial density of the right state.

First, note that the solutions in zones 1 and 5 are given simply by the initial left and right states,

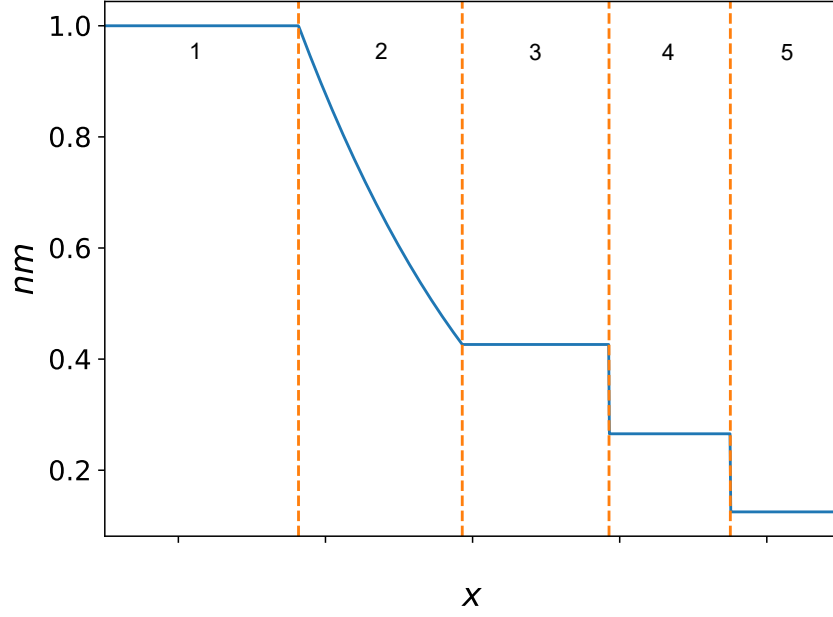


Figure 3.10: Density in the Sod shock tube problem at some arbitrary time $t > 0$. The dashed vertical lines represent the demarcations between the five zones, which are labeled 1-5 from left to right.

respectively. The pressure in zone 4 p_4 is given by the solution to

$$\frac{p_L}{p_R} = \frac{p_4}{p_R} \left\{ 1 - \frac{(\gamma - 1) \frac{c_R}{c_L} \left(\frac{p_4}{p_R} - 1 \right)}{\sqrt{2\gamma \left[2\gamma + (\gamma + 1) \left(\frac{p_4}{p_R} - 1 \right) \right]}} \right\}^{-2\gamma/(\gamma-1)}, \quad (3.20)$$

which can be found through Newton-Raphson iteration. With p_4 , the bulk velocity u_4 can be computed as

$$u_4 = \frac{c_R}{\gamma} \left(\frac{p_4}{p_R} - 1 \right) \sqrt{\frac{2 \frac{\gamma}{\gamma+1}}{\frac{p_4}{p_R} + \frac{\gamma-1}{\gamma+1}}}, \quad (3.21)$$

and the density ρ_4 is given by

$$\rho_4 = \rho_R \frac{\frac{p_2}{p_R} \frac{\gamma+1}{\gamma-1}}{\frac{p_2}{p_R} + \frac{\gamma+1}{\gamma-1}}. \quad (3.22)$$

Zones 4 and 5 are separated by a shock wave traveling with velocity

$$v_{45} = c_R \sqrt{\frac{\gamma+1}{2\gamma} \left(\frac{p_4}{p_R} - 1 \right) + 1}. \quad (3.23)$$

Zone 2 represents an expansion wave. The wavefront separating zones 1 and 2 propagates with velocity $v_{12} = -c_L$. Within zone 2, the fluid quantities are functions of x/t . The bulk velocity $u_2 = u_2(x/t)$ is given by

$$u_2 = \frac{2}{\gamma+1} \left(c_L + \frac{x}{t} \right). \quad (3.24)$$

Using the bulk velocity, the pressure $p_2 = p_2(x/t)$ can be computed as

$$p_2 = p_L \left(1 - \frac{\gamma - 1}{2} \frac{u_2}{c_L} \right)^{2\gamma/(\gamma-1)}, \quad (3.25)$$

and the density $\rho_2 = \rho_2(x/t)$ can be computed as

$$\rho_2 = \rho_L \left(\frac{p_2}{p_L} \right)^{1/\gamma}. \quad (3.26)$$

Zones 3 and 4 are separated by a contact discontinuity traveling with velocity $v_{34} = u_4$. The pressure and bulk velocity in zone 3 are identical to those in zone 4, so that $p_3 = p_4$ and $u_3 = u_4$. The separation between zones 2 and 3 propagates with a velocity v_{23} that is the solution to

$$u_3 - \sqrt{\frac{\gamma p_3}{\rho_2(v_{23})}} = u_2(v_{23}) - \sqrt{\frac{\gamma p_2(v_{23})}{\rho_2(v_{23})}}. \quad (3.27)$$

Finally, the requirement for density continuity between zones 2 and 3 demands that $\rho_3 = \rho_2(v_{23})$.

Since hPIC2 does not currently support one-dimensional domains for fluid problems, we simulate the Sod shock tube problem in two dimensions on the domain $\Omega = [-0.5, 0.5] \times [0, 0.4]$. The solution is measured along the centerline at $y = 0.2$. The termination time for the simulation is chosen to be $t = 0.2$. In order to perform a convergence study, the domain is initially discretized with a uniform square mesh whose constituent cells have side length $h = 0.02$. The simulation is repeated with successively halved mesh sizes. The finite element solution is expected to converge to the analytic solution at order 1/2 in the L_2 norm, regardless of finite element order. We use DG elements of order 1. A forward Euler time integrator is used, and the time step is dynamically chosen to satisfy the stability constraints with a CFL number of 0.3. The HLL Riemann solver is used [67], along with the slope limiter from Moe et al. [66].

The results of the verification study are shown in Figs. 3.11 and 3.12. hPIC2 output is plotted alongside the exact solutions for the four quantities of interest in Fig. 3.11. As h decreases, the hPIC2 solution matches the exact solution more closely. This trend is quantified in Fig. 3.12, which shows that the hPIC2 solution converges to the exact solution in the L_2 error norm at approximately order 1/2, as desired [119].

3.3 Performance

In this section, we demonstrate the performance and scalability of hPIC2 on the Summit and Delta supercomputers, which are based at Oak Ridge National Laboratory (ORNL) and the National Center for Supercomputing Applications (NCSA), respectively. We investigate both strong scaling, which concerns simulation wall time when running a problem of fixed size with increasing levels of parallelism, and weak scaling, which concerns wall time when running a problem where the size increases with parallelism. Both studies provide metrics for how efficiently an application performs on massively parallel architectures. We also compare simulation wall times on GPU- and CPU-based architectures to demonstrate the effectiveness of GPU acceleration.

With traditional CPU-based applications, the basic unit of parallelism is the processor, so strong scaling studies typically involve running a fixed problem on increasingly many processors. Likewise, weak scaling studies typically involve running a problem whose size is increased such that the workload per processor

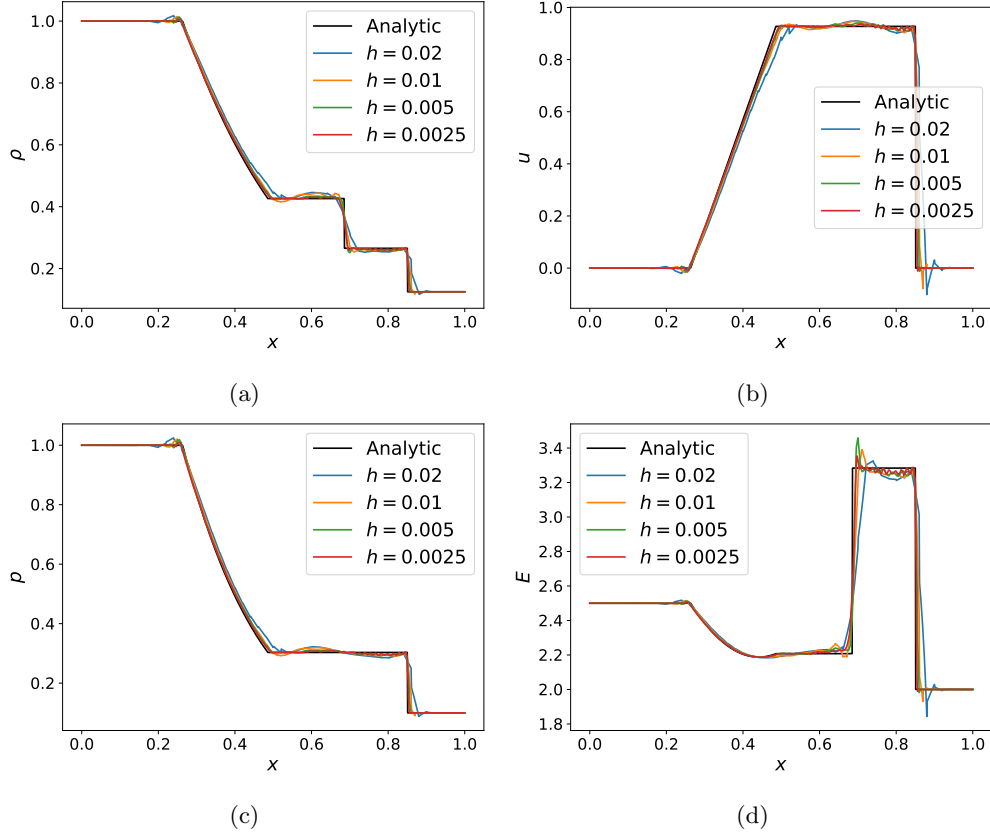


Figure 3.11: hPIC2 output for the Sod shock tube verification study compared to the exact solution: (a) density ρ ; (b) bulk velocity u ; (c) pressure p ; (d) specific energy E .

remains constant. On heterogeneous computing systems, it is more appropriate to use a compute node as the basic unit of parallelism.

Summit has 4608 IBM Power System AC922 nodes, each with 2 IBM POWER9 CPUs and 6 NVIDIA Tesla V100 GPUs. The POWER9 CPU has 22 physical cores, although one of these is reserved for overhead and is not available to users. Each core supports up to 4 hardware threads. Each node therefore has 42 available physical cores and 168 available hardware threads. Most compute nodes also have 512 GB of DDR4 RAM for the CPUs, although there are some high-memory nodes with 2 TB. The V100 accelerator has 16 GB of second-generation high-bandwidth memory on standard compute nodes, while high-memory nodes use V100s with 32 GB of memory.

Delta is primarily composed of three different types of compute nodes. All primary nodes have 256 GB of DDR4 RAM and use AMD EPYC 7763 Milan processors, which have 64 physical cores. Delta also utilizes Nvidia A100 GPUs, which have 40 GB of second-generation high-bandwidth memory, and Nvidia A40 GPUs, which have 48 GB of GDDR6 memory. Delta has 124 CPU-only nodes consisting of 2 processors, 100 nodes consisting of 1 processor and 4 A100 GPUs, and 100 nodes consisting of 1 processor and 4 A40 GPUs. There are also a small number of nodes with alternate configurations.

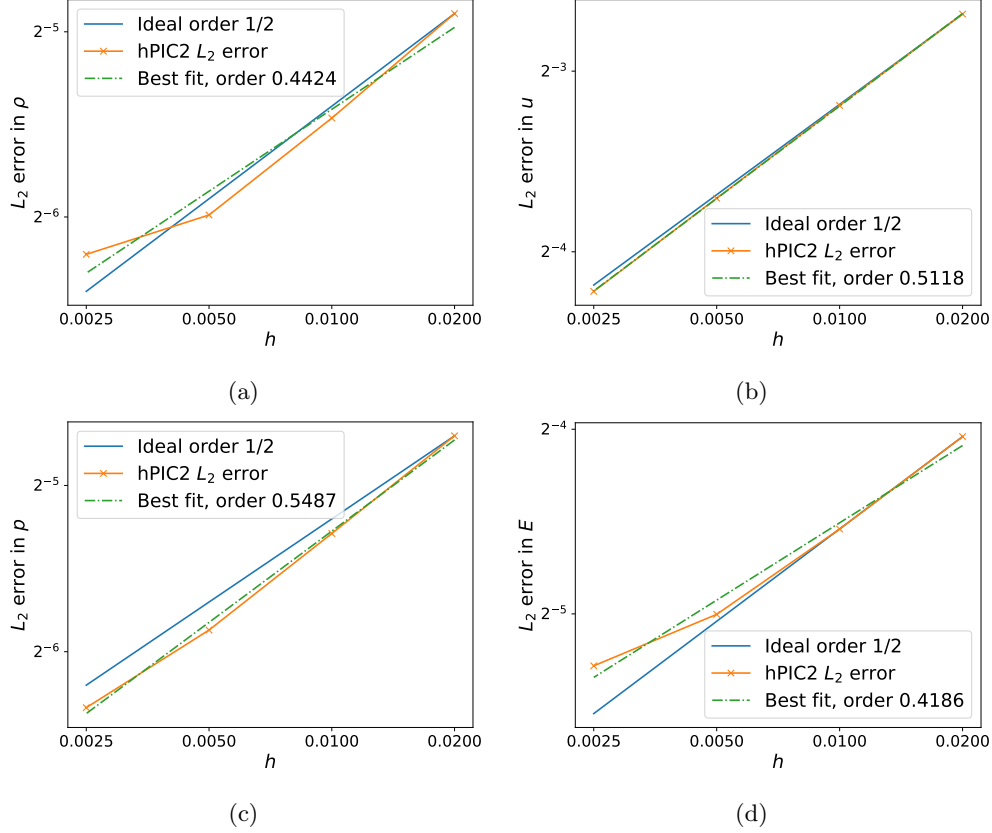


Figure 3.12: Convergence study for the Sod shock tube verification study, showing the L_2 error norm of the finite element solution alongside a best-fit line indicating the approximate convergence order and a line indicating the ideal order-1/2 convergence [119]: (a) density ρ ; (b) bulk velocity u ; (c) pressure p ; (d) specific energy E . Note that other error norms may induce different convergence rates; for example, the L_1 error norm is expected to converge at order 1 [120].

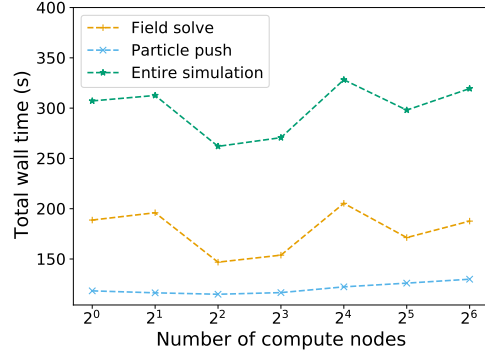
3.3.1 Weak scaling on Summit

For our scaling studies on Summit, we use a Kokkos installation with both the CUDA and OpenMP backends. We assign 6 MPI processes to each compute node, so that each GPU is bound to a unique MPI process. Each MPI process is also assigned 7 physical CPU cores such that each OpenMP thread is bound to a unique physical core. We disable hardware threading. As previously stated, we use the compute node as the basic unit of parallelism.

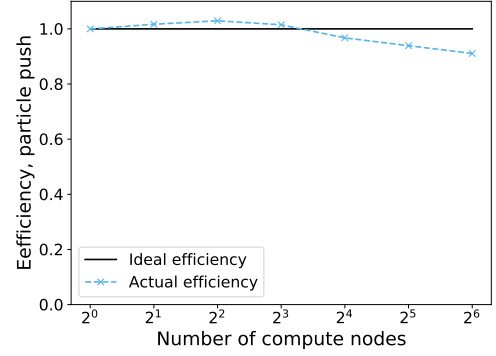
When selecting a problem with which to evaluate performance, it is important to provide enough work to saturate available computing resources and to balance the load evenly among them. To this end, we have chosen to simulate an unmagnetized PIC electron cloud with a static ion background in a square domain with electrically grounded, reflecting walls.

The domain is uniformly filled with electrons up to a number density of $5 \times 10^{15} \text{ m}^{-3}$ and with an initial temperature of 10 eV. A static uniform charge background of hydrogen ions with number density $5 \times 10^{15} \text{ m}^{-3}$ is applied in order to maintain quasineutrality.

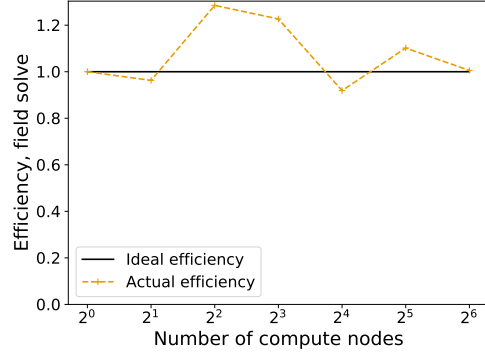
A baseline problem size is chosen to run on a single compute node. The problem size is then approximately doubled and run on twice as many nodes. Our baseline problem uses 100 million macroparticles. The side length of the baseline problem's square domain is taken to be $L = 0.8 \text{ m} \approx 2400 \lambda_D$. A triangular mesh



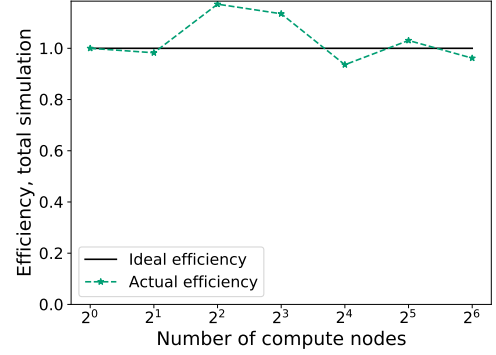
(a) Plot of wall times versus number of compute nodes.



(b) Efficiency plot of the particle push.



(c) Efficiency plot of the field solve.



(d) Efficiency plot of the entire simulation.

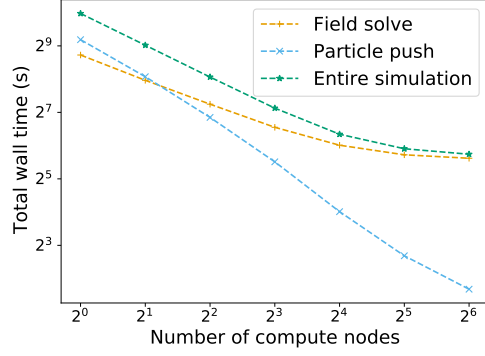
Figure 3.13: Results of the weak scaling study on an unstructured mesh on Summit.

is applied to this domain by first applying a uniform mesh, then splitting each square element into two triangular meshes. The baseline mesh consists of 1000 edges in each direction, for a total of 2 million elements. With electron thermal speed $v_{th} = \sqrt{2kT/m}$, the time step is chosen to be $\Delta t = 10^{-11} \text{ s} \approx \lambda_D/18v_{th}$ and 100 time steps are simulated. Each time the problem is refined, the size of the domain is also increased in order to maintain a constant $\Delta x/\Delta t$. Problem sizes for refined problems are shown in Table 3.2.

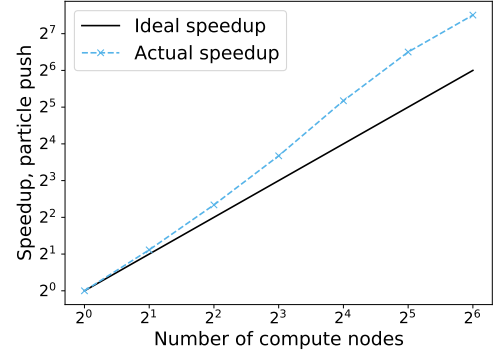
# compute nodes	# elements	# DoFs	# particles	$L (\lambda_D)$
1	2000000	1002001	100M	2400
2	3998792	2002225	200M	3394
4	8000000	4004001	400M	4800
8	15995168	8003241	800M	6788
16	32000000	16008001	1.6B	9600
32	63980672	32001649	3.2B	13 577
64	128000000	64016001	6.4B	19 200

Table 3.2: Problem sizes used during the unstructured mesh weak scaling study on Summit.

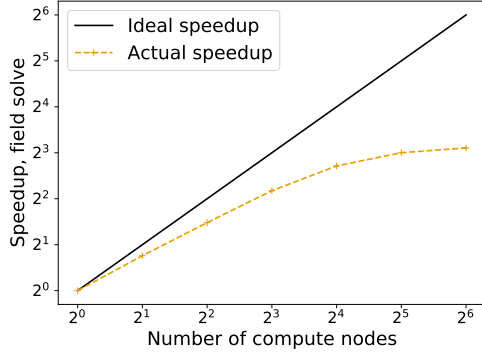
We record the total wall times spent in both the particle push and field solve, as well as the wall time of the entire simulation, excluding initialization. The field solve wall time includes time spent in MPI halo communications, and the particle push wall time includes time spent migrating particles between MPI processes. These are plotted as a function of the number of compute nodes in Fig. 3.13a. We also compute



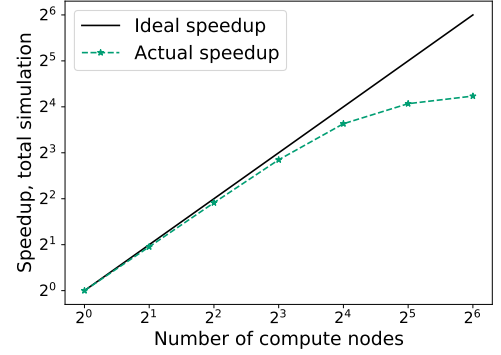
(a) Log-log plot of wall times versus number of compute nodes.



(b) Speedup plot of the particle push.



(c) Speedup plot of the field solve.



(d) Speedup plot of the entire simulation.

Figure 3.14: Results of the strong scaling study on an unstructured mesh on Summit.

the weak scaling efficiency as

$$e(N) = \frac{t(1)}{t(N)}, \quad (3.28)$$

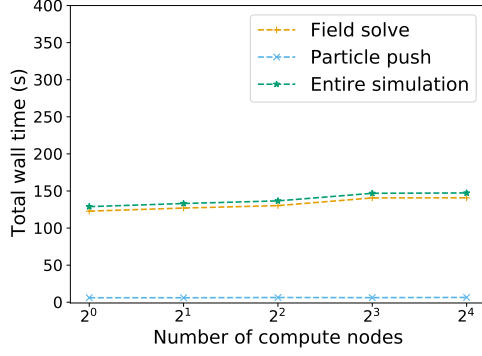
where N is the number of compute nodes and $t(N)$ is the wall time from the simulation with N compute nodes. This is plotted for each of the three timed kernels in Fig. 3.13.

The field solve wall time dictates the total simulation wall time. The varying field solve efficiencies in Fig. 3.13c are primarily due to differences in the quality of the mesh decomposition and slight load imbalances that arise therefrom. However, Fig. 3.13b shows that the particle push scales excellently, achieving near-perfect weak scalability.

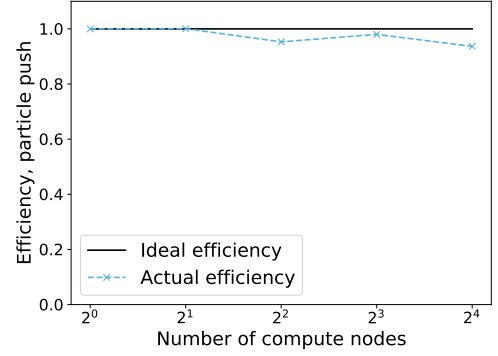
3.3.2 Strong scaling on Summit

For the strong scaling study, a single problem size is chosen, then run on increasingly many compute nodes. The runtime configuration is identical to that described in Section 3.3.1. The problem domain is chosen to have side length $L = 0.8 \text{ m} \approx 2400 \lambda_D$. This domain is meshed with a regular triangular mesh consisting of 2 million elements, which corresponds to 1 002 001 degrees of freedom. The plasma is identical to the weak scaling study, with 100 million macroparticles representing the electrons.

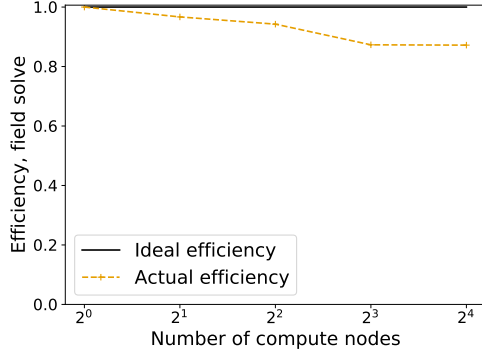
We record the total wall times spent in both the particle push and the field solve, as well as the wall time of the entire simulation, excluding initialization. The field solve wall time includes time spent in MPI



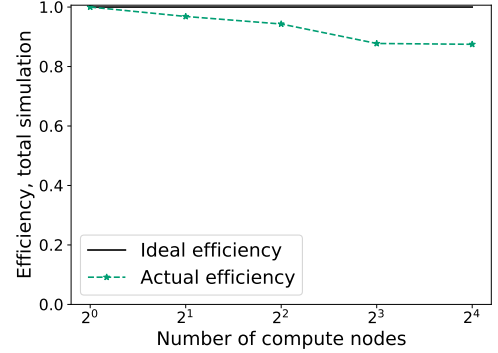
(a) Plot of wall times versus number of compute nodes.



(b) Efficiency plot of the particle push.



(c) Efficiency plot of the field solve.



(d) Efficiency plot of the entire simulation.

Figure 3.15: Results of the weak scaling study on an unstructured mesh on Delta without RustBCA.

halo communications, and the particle push wall time includes time spent migrating particles between MPI processes. These are plotted versus the number of compute nodes in Fig. 3.14a. We also compute the speedup as

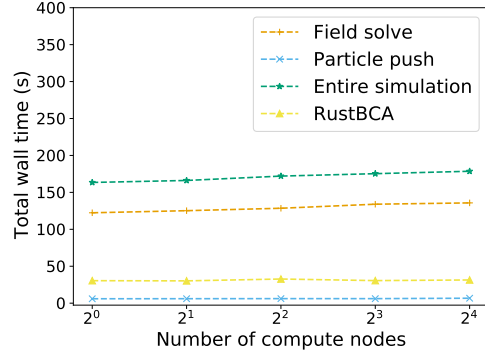
$$s(N) = \frac{t(1)}{t(N)}. \quad (3.29)$$

This is plotted for each of the three timed kernels in Fig. 3.14. Fig. 3.14b shows that hPIC2's particle push scales well even down to less than one million macroparticles per GPU. Fig. 3.14c shows that the field solve does not exhibit perfect linear scaling for this problem. This is because increasing the number of compute nodes reduces the problem size per node at the cost of increased halo communication, which can dominate the simulation when the problem size per node becomes sufficiently small.

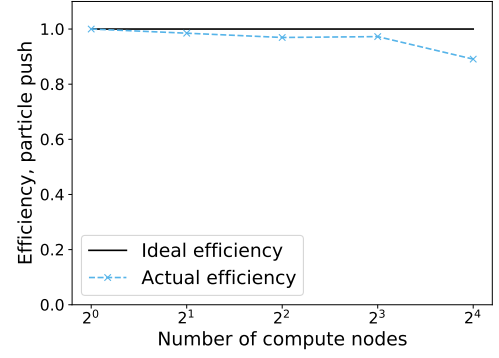
3.3.3 Weak scaling on Delta

For our scaling studies on Delta, we use a Kokkos installation with both the CUDA and OpenMP backends. We assign 4 MPI processes to each compute node, so that each GPU is bound to a unique MPI process. Each MPI process is also assigned 16 physical CPU cores such that each OpenMP thread is bound to a unique physical core.

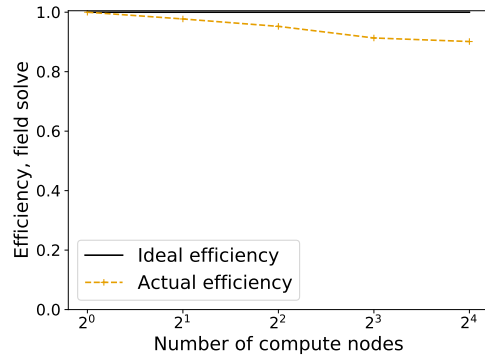
In contrast to the scaling studies on Summit, the problem for the scaling studies on Delta consists of a kinetic ion plasma against a static electron background. A square domain is uniformly filled with hydrogen



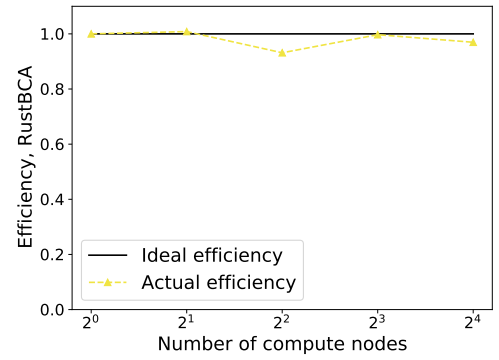
(a) Plot of wall times versus number of compute nodes.



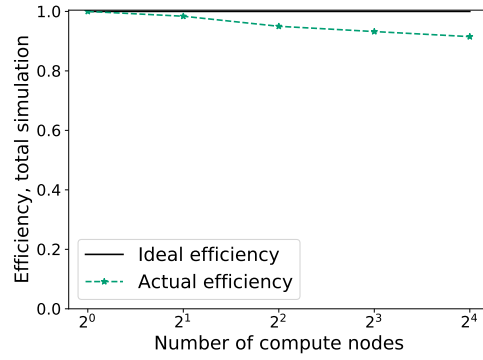
(b) Efficiency plot of the particle push.



(c) Efficiency plot of the field solve.



(d) Efficiency plot of RustBCA.



(e) Efficiency plot of the entire simulation.

Figure 3.16: Results of the weak scaling study on an unstructured mesh on Delta with RustBCA.

# compute nodes	# elements	# DoFs	# particles	$L (\lambda_D)$
1	2000000	1002001	100M	1000
2	3998792	2002225	200M	1414
4	8000000	4004001	400M	2000
8	15995168	8003241	800M	2828
16	32000000	16008001	1.6B	4000

Table 3.3: Problem sizes used during the unstructured mesh weak scaling study on Delta.

ions up to a number density of $5 \times 10^{15} \text{ m}^{-3}$ and with an initial temperature of 10 eV. A static uniform charge background of electrons with number density $5 \times 10^{15} \text{ m}^{-3}$ is applied in order to maintain quasineutrality. The walls are electrically grounded. We perform the study with two different configurations for the particle boundary conditions: first, with all reflecting walls; and second, with three reflecting walls and one wall treated as a boron wall with RustBCA.

As in Section 3.3.1, a baseline problem is chosen to run on a single compute node. The baseline problem uses 100 million macroparticles. The side length of the baseline problem’s square domain is taken to be $L = 0.332 \text{ m} \approx 1000 \lambda_D$. A regular triangular mesh is applied consisting of 1000 edges in each direction, for a total of 2 million elements. With ion thermal speed $v_{\text{th}} = \sqrt{2kT/m}$, the time step is chosen to be $\Delta t = 1.5 \text{ ns} \approx \lambda_D/5v_{\text{th}}$ and 100 time steps are simulated. Each time the problem is refined, the size of the domain is also increased in order to maintain a constant $\Delta x/\Delta t$. Problem sizes for refined problems are shown in Table 3.3. Due to constraints on per-user node limits, we run this problem on a maximum of 16 compute nodes.

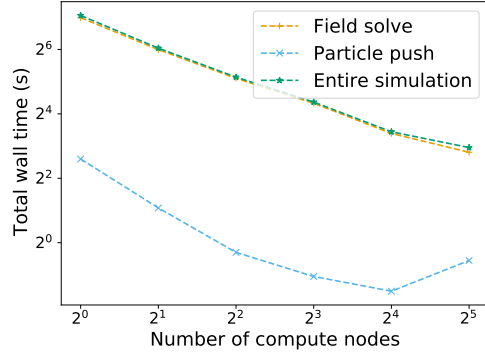
We record the total wall times spent in the particle push, field solve, and RustBCA kernel if it is turned on, as well as the wall time of the entire simulation, excluding initialization. The field solve wall time includes time spent in MPI halo communications, and the particle push wall time includes time spent migrating particles between MPI processes. These are plotted as a function of the number of compute nodes in Fig. 3.15a without RustBCA and Fig. 3.16a with RustBCA. The weak scaling efficiency is plotted for each of the timed kernels in Fig. 3.15 without RustBCA and Fig. 3.16 with RustBCA.

As in Section 3.3.1, the field solve wall time dictates the total simulation wall time, though this problem appears to be less susceptible to differences in mesh decomposition. In all cases, the particle push, field solve, and RustBCA kernels scale well.

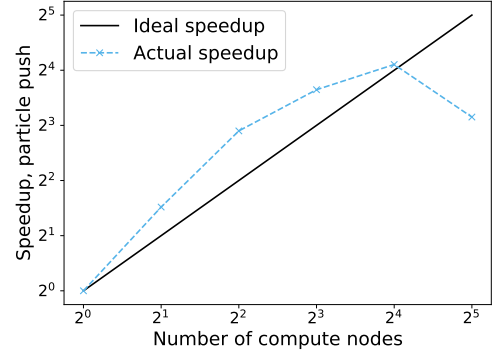
3.3.4 Strong scaling on Delta

As in Section 3.3.2, a single problem size is chosen, then run on increasingly many compute nodes. The runtime configuration is identical to that described in Section 3.3.1, and we use the baseline problem from the weak scaling study, both with and without RustBCA.

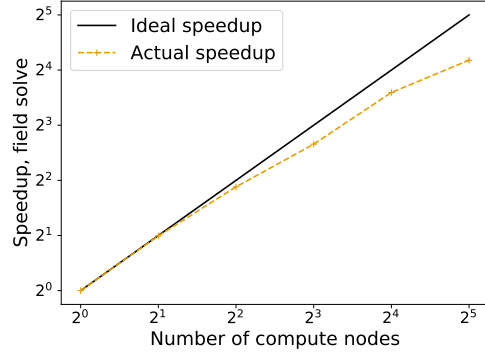
Wall times for the particle push, field solve, RustBCA kernel if enabled, and entire simulation are recorded, excluding initialization. The field solve wall time includes time spent in MPI halo communications, and the particle push wall time includes time spent migrating particles between MPI processes. These are plotted versus the number of compute nodes in Fig. 3.14a. We also compute the speedup with (3.29). This is plotted for each of the timed kernels in Fig. 3.17 without RustBCA and in Fig. 3.18 with RustBCA. Figs. 3.17b and 3.18b show that hPIC2’s particle push scales well until 16 compute nodes, at which point a sharp decline in performance is recorded. This is because the GPU’s are insufficiently saturated with particle work; with 16 compute nodes, each GPU is responsible for less than 2 million macroparticles, which is considerably less



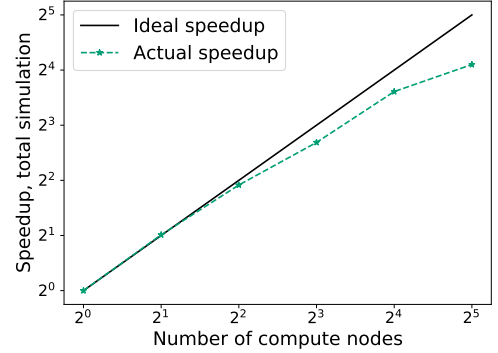
(a) Log-log plot of wall times versus number of compute nodes.



(b) Speedup plot of the particle push.



(c) Speedup plot of the field solve.



(d) Speedup plot of the entire simulation.

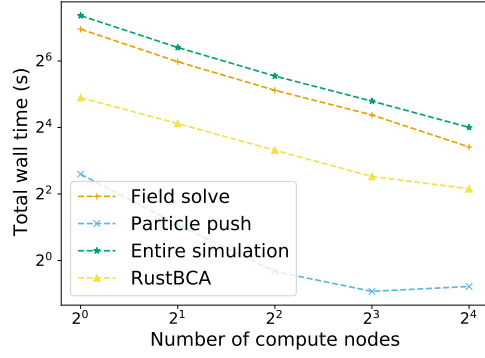
Figure 3.17: Results of the strong scaling study on an unstructured mesh on Delta without RustBCA.

than the typical particle load in real problems. The fact that this failure in strong scalability was not seen on Summit is because the A100 GPU used in Delta is individually more powerful than the V100 GPU used in Summit. All other kernels scale as desired, both with and without RustBCA.

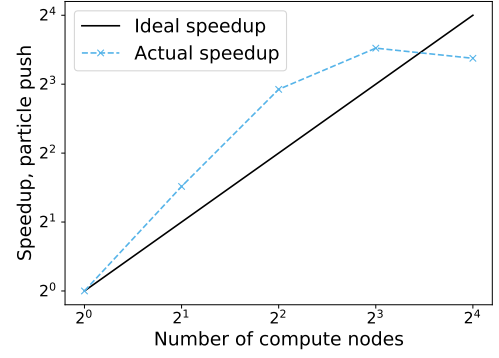
3.3.5 Wall time comparison on various architectures

The scaling studies were performed with GPUs. To demonstrate the effectiveness of GPU acceleration in PIC simulations, we also simulate the problem used in Section 3.3.2 on one compute node in a CPU-based configuration on Summit and on each of the three types of primary nodes on Delta. The Summit CPU-based configuration uses a Kokkos installation with the OpenMP backend. We use 2 MPI processes, so that each socket on the compute node is bound to a unique MPI process. Each MPI process is assigned 21 physical CPU cores such that each OpenMP thread is bound to a unique physical core. The Delta CPU node configuration uses 2 MPI processes, each exploiting all 64 available CPU cores. Both Delta GPU node configurations use 4 MPI processes, each of which is assigned 16 CPU cores and 1 GPU.

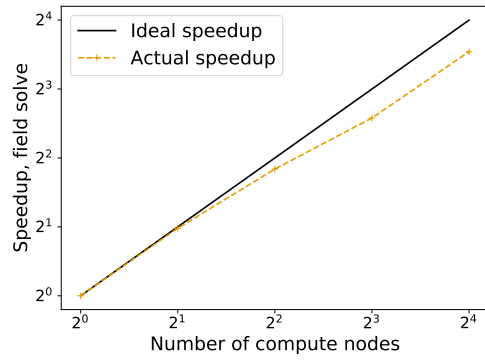
A bar chart with the wall times for the simulation with all configurations is shown in Fig. 3.19. The chart also breaks down the total wall time into particle push, field solve, and other segments. Note that, in all cases, hPIC2 spends an extremely small fraction of the wall time in kernels outside the particle push or field solve. The chart shows a clear advantage to GPU acceleration for PIC codes, especially for the particle push. On CPU-based architectures, the particle push tends to dominate total simulation wall time compared to the



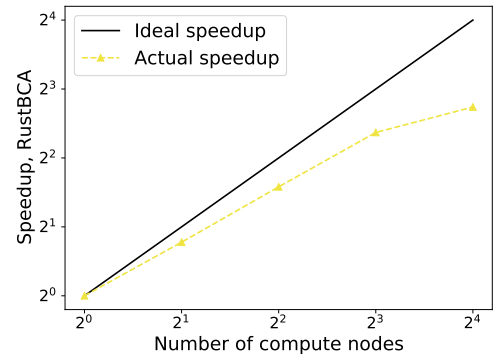
(a) Log-log plot of wall times versus number of compute nodes.



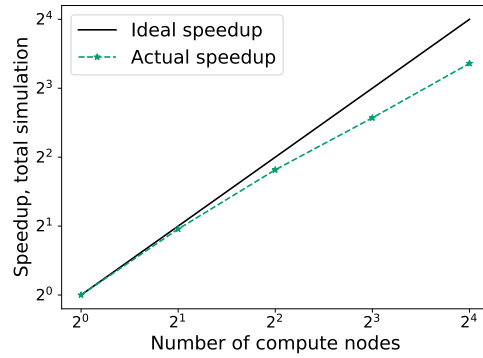
(b) Speedup plot of the particle push.



(c) Speedup plot of the field solve.



(d) Speedup plot of RustBCA.



(e) Speedup plot of the entire simulation.

Figure 3.18: Results of the strong scaling study on an unstructured mesh on Delta with RustBCA.

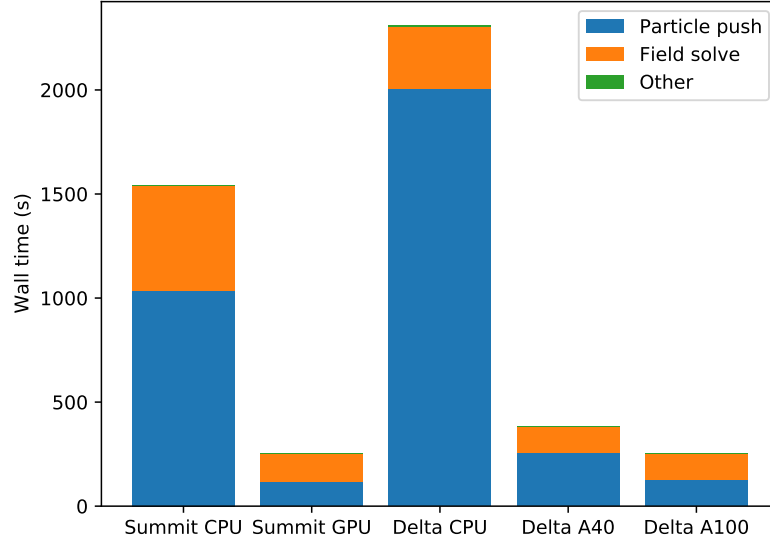


Figure 3.19: Bar chart of wall times for the performance comparison test, broken down into particle push, field solve, and other segments. Note that the field solve wall time includes MPI halo communication, and the particle push wall time includes particle migration.

field solve. However, on GPU architectures, the particle push consumes a much smaller portion of the total wall time. For some problems, this portion is a minority of the total wall time. Both kernels see an absolute performance improvement with GPU acceleration.

3.4 Demonstrations

In this section, we show two demonstrations on problems of practical interest using hPIC2, selecting one problem of interest in fusion energy sciences, and one problem relevant to low-temperature plasmas. Section 3.4.1 models a 1D radio-frequency (RF) plasma sheath with Boltzmann electrons and sputtering from material walls, a typical problem in fusion and RF-heated plasmas. Section 3.4.2 models a 2D plasma with sputtering in a domain with silicon trenches and geometrical features.

3.4.1 Radio-Frequency Sheath in Nuclear Fusion Conditions

In this section, we demonstrate hPIC2’s BCA and Monte Carlo ionization operator to simulate a 1D radio-frequency sheath bordered by a material wall. Radio-frequency sheaths are of interest in nuclear fusion plasmas because they lead to increased material sputtering and impurity accumulation in the plasma due to sheath rectification. After being released from the surface into the plasma, impurities ionize and are accelerated back toward the wall.

Figure 3.20 shows a sketch of the RF sheath model problem for hPIC2 to simulate. The boundary at $x = 0$ is an electrically grounded absorbing boundary. The boundary at $x = L$ imposes an RF Dirichlet boundary condition given by

$$\phi(x = L) = \frac{V_{pp}}{2} \cos(\omega t), \quad (3.30)$$

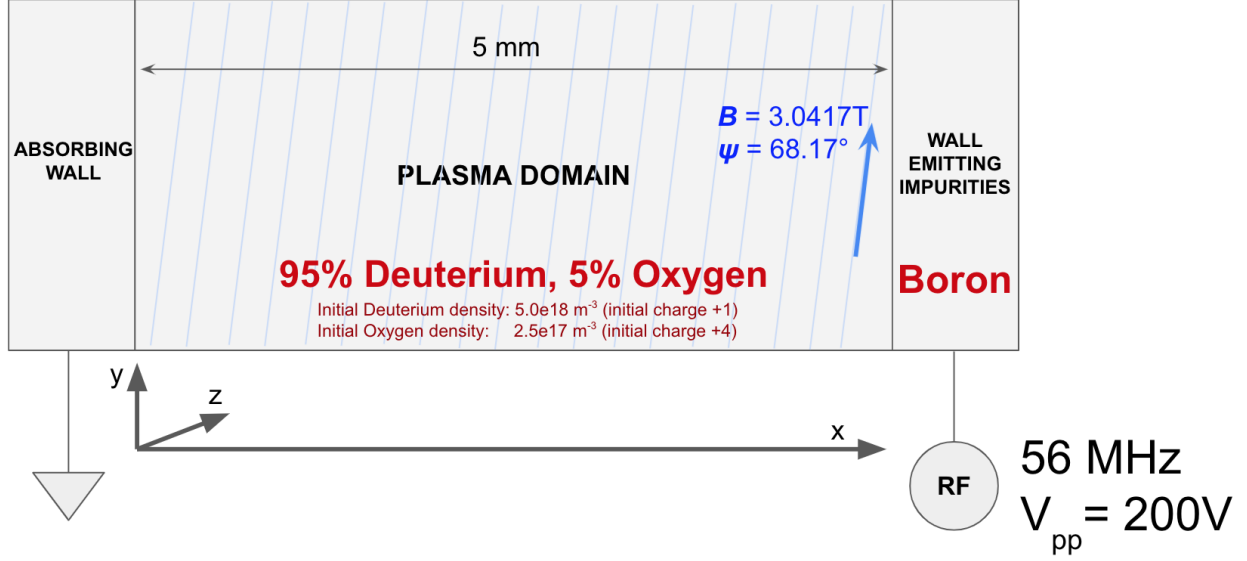


Figure 3.20: 1D radio-frequency simulation model.

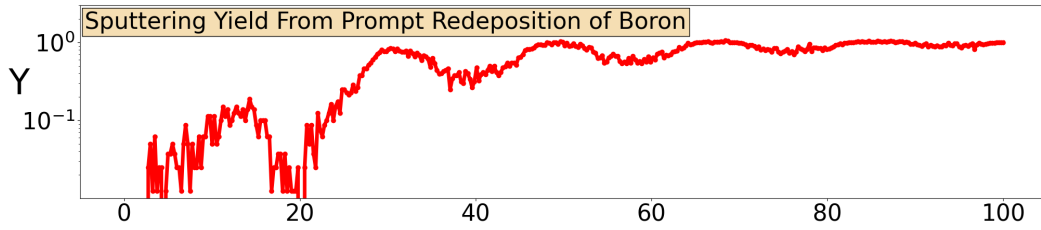


Figure 3.21: Sputtering Yield of Boron from incident Boron to the wall in RF simulation with PMI and Monte-Carlo ionizations enabled. This can only occur if neutral Boron is sputtered from the wall, ionized, and redeposited onto the wall. This highlights the power of a dynamic coupling between PIC and PMI models.

The mesh for this problem is a uniform mesh with dimensions $[0 \text{ mm}, 5 \text{ mm}]$ and cell size $5 \times 10^{-5} \text{ m}$. The total physical time of the simulation is 100 ns with a time step of 0.25 ns . The magnetic field of 3.0 T is inclined with the surface at an angle of 68.17° . V_{pp} is 200 V while $\omega = 56 \text{ MHz}$. The electron density near the wall is $5.25 \times 10^{18} \text{ m}^{-3}$, which is chosen from typical experimental values at the ICRH limiter in modern fusion devices. We use Boltzmann electrons with a fixed temperature of 10 eV . Kinetic ions are sourced into the domain with a temperature of 10 eV .

This simulation was run on the Illinois Campus Cluster. It utilized 80 MPI processes, with OPENMP enabled and no GPU acceleration. Each MPI process used 12 cores. The simulation domain was initialized with 500M Deuterium macroparticles as well as 500M Oxygen macroparticles. The total wall time for the simulation was 5 hours.

Fig. 3.22 demonstrates the imposed periodic boundary condition as well as the resulting periodic plasma and PMI properties. The plasma potential at the center of the domain shows contributions from a thermal and rectified sheath, demonstrating hPIC2's ability to capture this well-known radio-frequency phenomenon. Deuterium and oxygen ion fluxes to the wall experience a spike at the beginning of the simulation and then settle into a periodic trend following the sheath potential drop $-(V_{RF}(t) - \Phi_0(t))$. During time steps of high ion flux to the wall, boron is sputtered in greater quantities, as seen from the time-dependent sputtering yields

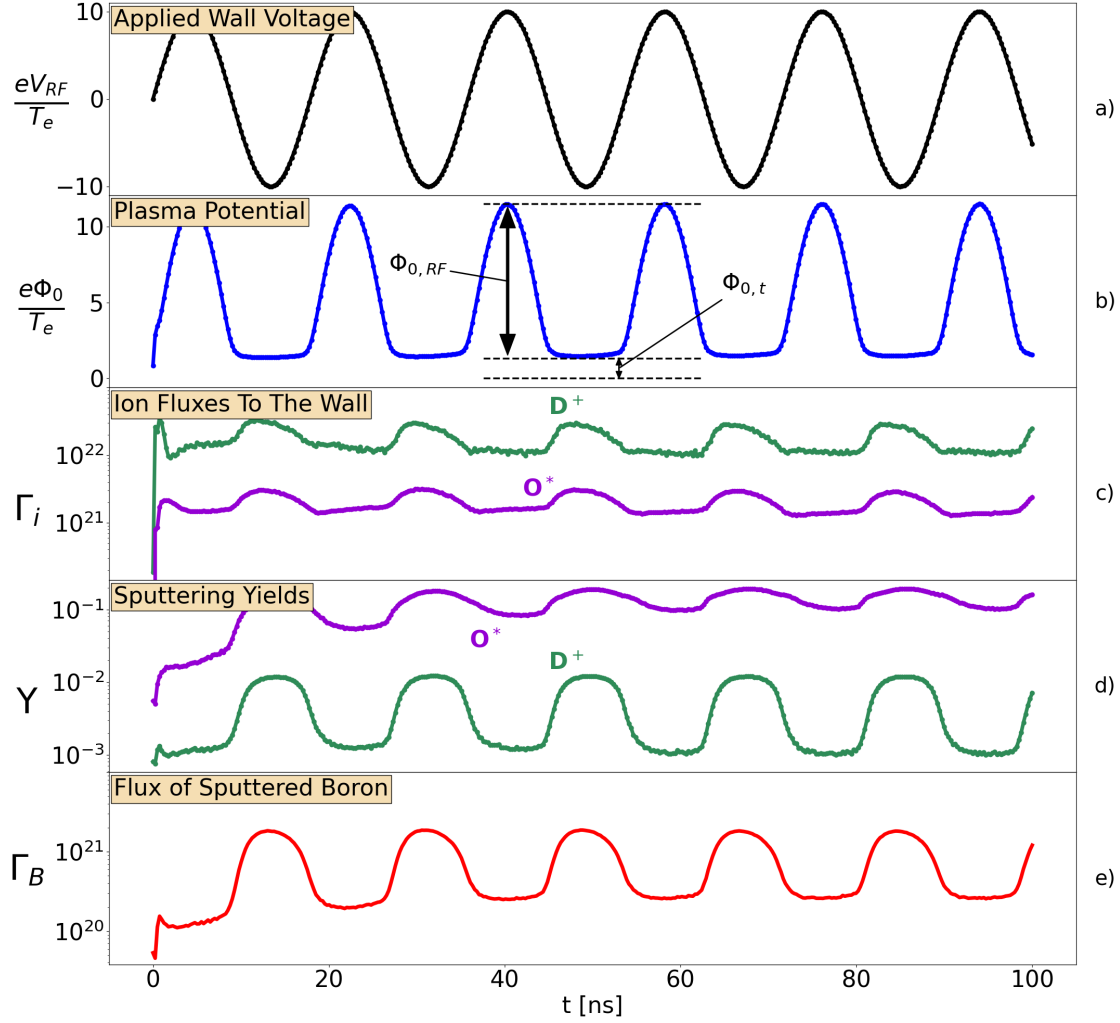


Figure 3.22: Periodic boundary conditions and resulting periodic properties of RF simulation with PMI and Monte-Carlo ionizations enabled. a) the boundary condition imposed on the material wall; b) the plasma potential at the midpoint of the domain, showing thermal and rectified sheath contributions. This demonstrates hPIC2's ability to solve for a rectified potential; c) ion fluxes to the wall (O^* represents all Oxygen charge states); d) sputtering yield of Boron from the wall, broken down by incident species; e) flux of sputtered (neutral) Boron from the wall.

and fluxes of sputtered boron. Once sputtered, boron enters the plasma domain as a neutral particle and then is ionized via the electron background using hPIC2’s Monte Carlo ionization operator. After ionization, a portion of this boron will be redirected toward the wall, where more sputtering can occur. Figure 3.21 demonstrates the time-dependent increase of this erosion-redeposition phenomenon. The self-sputtering yield approaches unity because of the mass equivalence of incident and material particles.

This simulation demonstrates the ability for an hPIC2-RustBCA coupling to model the time-dependent, phase-resolved plasma-material interactions in radio-frequency fusion conditions.

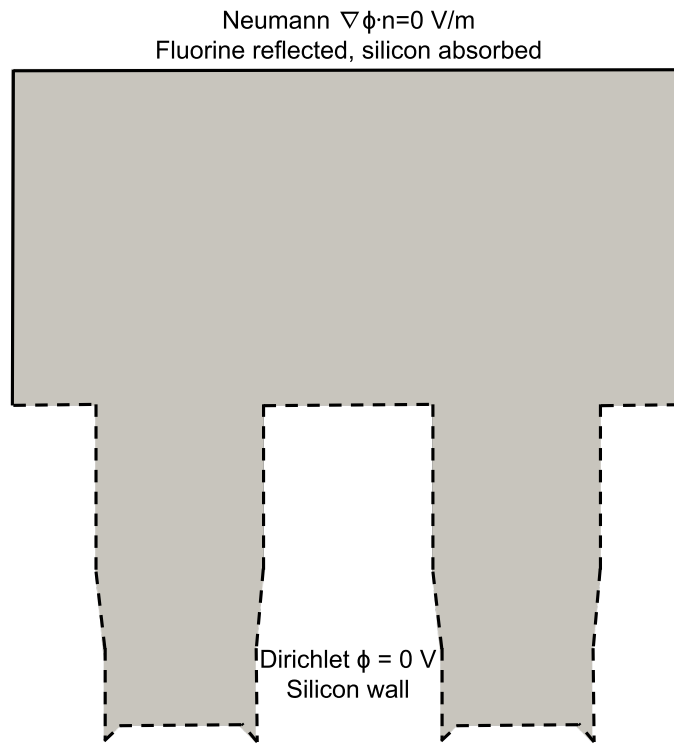
3.4.2 Sputtering from trenches in a silicon wafer

In this section, we combine hPIC2’s FEM and BCA coupling capabilities to simulate a plasma in a complex 2D domain with surface interactions. The problem domain consists of two trenches in a silicon wafer, with nontrivial surface geometrical and composition features within the trenches. It has direct applications to semiconductor plasma etching, and is intended to act as a surrogate model for other problems such as the simulation of plasmas in tokamak tile gaps.

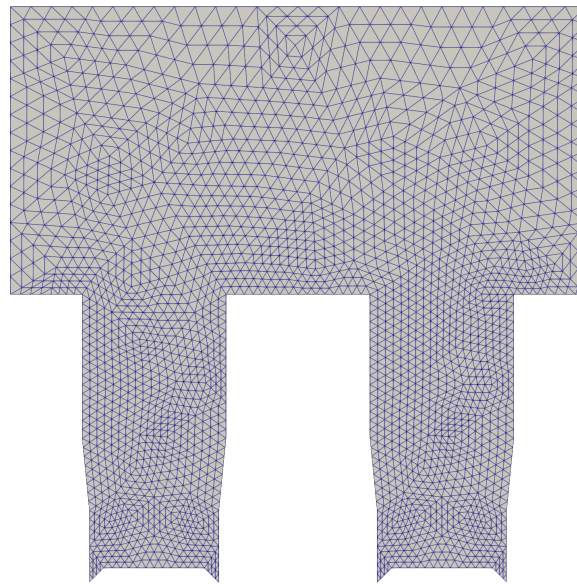
The domain and mesh for the problem is shown in Fig. 3.23. The bounding box for the domain is $[0 \text{ mm}, 1.6 \text{ mm}] \times [-0.8 \text{ mm}, 0.8 \text{ mm}]$. The upper three edges, marked by a solid line in Fig. 3.23a, represent outflow boundaries; the remaining lower edges and trenches, marked by a dashed line, represent the surface of the silicon.

The domain is uniformly filled with singly ionized fluorine ions up to a number density of $1.6 \times 10^{17} \text{ m}^{-3}$ and with temperature 300 K. This approximately corresponds to a pressure of 50 mtorr with an ionization ratio of 10^{-3} , which is typical of industrial plasmas. Boltzmann electrons are fixed at temperature 10 eV; this is greater than most industrial plasmas, but is chosen to induce a higher sputtering rate on an unbiased substrate. The silicon surface is treated using RustBCA and electrically grounded. The outflow boundaries impose a homogeneous Neumann boundary condition on the potential and perfectly reflect fluorine, but remove silicon from the simulation. As fluorine ions are neutralized due to interactions with the silicon surface, new fluorine ions are populated uniformly throughout the domain in order to maintain the initial ion count. This approximates ionization events between the electrons and a neutral fluorine background. 500 time steps of $2.5 \times 10^{-10} \text{ s}$ each are simulated.

Fig. 3.24 shows plots of the electric potential throughout the domain at a few time steps of the simulation. Note how a sheath forms along the edge of the silicon surface. Fig. 3.25 shows scatter plots of the positions of fluorine macroparticles (blue) and silicon macroparticles (red) at a few time steps of the simulation. This problem induces significant silicon sputtering, as seen in Fig. 3.25d. Note that by time step 50, fluorine particles have just begun to accelerate sufficiently toward the silicon to cause sputtering. Only after the formation of a sheath does sputtering begin to occur at a significant rate.

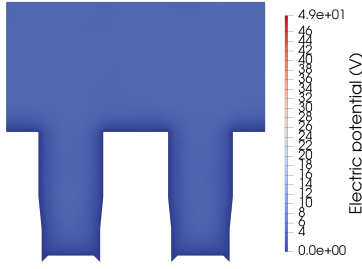


(a) Domain for the trench problem.

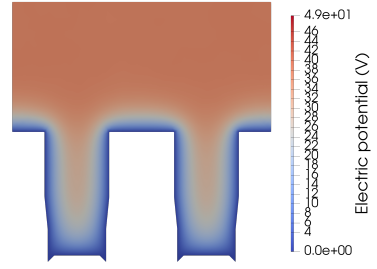


(b) Mesh for the trench problem.

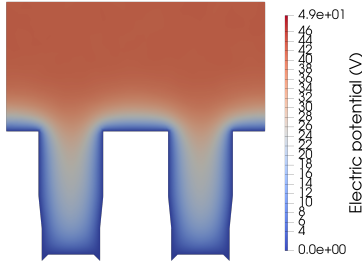
Figure 3.23: Domain, boundary conditions, and mesh for the trench problem.



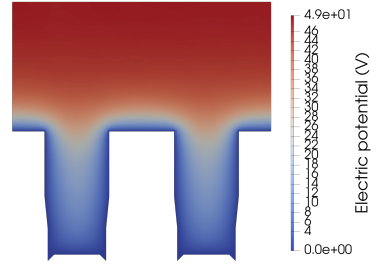
(a) Initial electric potential at time step 0, $t = 0$ s.



(b) Electric potential at time step 50, $t = 1.25 \times 10^{-8}$ s.

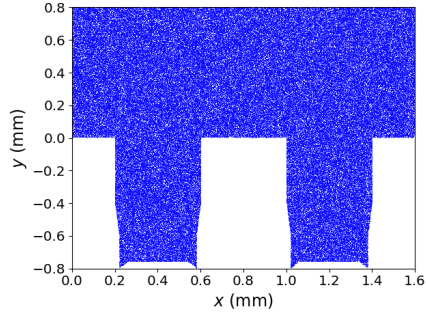


(c) Electric potential at time step 100, $t = 2.5 \times 10^{-8}$ s.

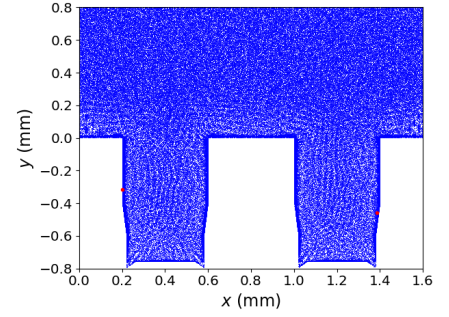


(d) Electric potential at time step 500, $t = 1.25 \times 10^{-7}$ s.

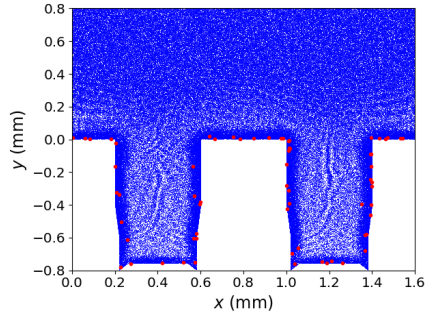
Figure 3.24: Plots of the electric potential during the trench problem at four time steps.



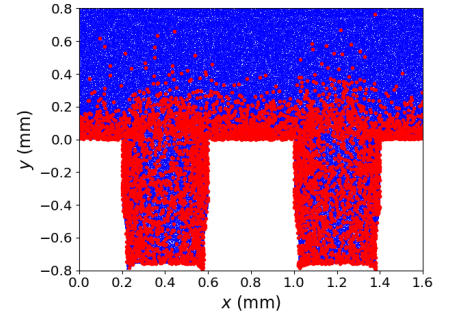
(a) Initial particle positions at time step 0, $t = 0$ s.



(b) Particle positions at time step 50, $t = 1.25 \times 10^{-8}$ s.



(c) Particle positions at time step 100, $t = 2.5 \times 10^{-8}$ s.



(d) Particle positions at time step 500, $t = 1.25 \times 10^{-7}$ s.

Figure 3.25: Scatter plots of particle positions during the trench problem at four time steps.

Chapter 4

Merging Particles into a Fluid in Hybrid Fluid-PIC Plasma Simulations

The particle-in-cell (PIC) method is widely used in computational plasma physics [19], [121]–[123], owing to its ease of implementation, ability to characterize far-from-equilibrium kinetic effects, and expandability to collisional regimes with various collision algorithms [51], [52], [106], [107]. Since it is effectively a nonlinear Monte Carlo scheme, however, PIC tends to introduce noise, and can be computationally expensive when the number of particles is large, particularly in problems where the size of elements varies greatly over the mesh. This computational cost is exacerbated in multiphysics simulations incorporating collisional models, where the number of particles can grow as the plasma density grows in time due to ionization or other processes. Several algorithms have been developed to reduce the number of particles in a problem by combining or merging several particles into fewer particles [124]–[130], and we refer to this class of algorithms as particle merges.

Fluid models avoid many of the issues of PIC. They do not introduce numerical noise into problems, more easily admit implicit solvers that allow one to underresolve the plasma frequency, and generally perform faster than pure PIC simulations in identical regimes. The caveat is that the fluid equations that they solve make some assumptions on the shape of the velocity distribution in order to be closed. Hence, the regime of validity of fluid solvers is limited compared to PIC. As an example, the Euler equations assume that the fluid is perfectly inviscid with zero heat flux, which is violated by collisionless plasmas far from equilibrium.

Hybrid plasma schemes combine the strengths of multiple methods. It has long been found that modeling different species with different methods can reduce computational workload [36], [131]–[135]. For example, assuming that electrons adiabatically follow a Maxwell-Boltzmann distribution with PIC ions allows one to step over typical electron kinetic timescales [44], [136], greatly reducing compute time. More recent work has explored using multiple methods for a single plasma species, splitting the distribution among the methods [137]–[140]. In this chapter, we restrict our attention primarily to this latter hybrid scheme, where the Maxwellian portion of the species is treated with fluid and the far-from-equilibrium portion with PIC.

Though hybrid methods are associated with plasmas wherein some physical drive prevents the plasma from approaching equilibrium, hybrid methods may also be useful in transient systems that are initially far from equilibrium but thermalize over the course of a simulation. A representative example is the simulation of power flow in pulsed-power applications. Historically, power flow has been modeled using PIC, since the electron velocity distribution initially violates the assumptions of typical fluid closures. The particle count

eventually rises as the power output reaches its peak, often significantly slowing down compute time. However, a substantial subgroup of the particle distribution approximates a drifting Maxwellian distribution, which is well-characterized by a fluid. In principle, it is ideal to model this subgroup using a fluid solver while the remainder is modeled using PIC. For this approach to be effective, there must be a way for PIC particles to be absorbed into the fluid, a process we refer to as hybrid merge. Otherwise, collisional and other physical events will generally cause the particle count to increase unimpeded.

Another salient example is the simulation of relativistic electron beams propagating through a neutral background gas [43]. The electron beam is not initially suitable for a fluid description, since it violates typical fluid model closure assumptions. However, through collisions with the background gas, the constituent electrons quickly thermalize and readily resemble a fluid, even over simulation timescales. Without some means of converting the thermalized subpopulation of the former electron beam into a fluid, the computational cost of describing them kinetically can rapidly overwhelm computing resources. In this context, a hybrid merge algorithm is desired.

The purpose of our hybrid merge is to work in tandem with a particle merge. A hybrid merge is not necessarily intended to reduce the PIC particle count by itself, but to reduce the number of physical particles represented by PIC and transfer their representation into the fluid solver. A subsequent particle merge then completes the task of reducing the PIC particle count. Hybrid merge does this by siphoning weight from variable-weight PIC particles to the fluid.

The goal of a hybrid merge, then, is to identify a weighted subgroup of particles that, when absorbed by the fluid, does not appreciably skew the fluid into a regime where its closure assumptions are violated. Furthermore, since an ultimate goal of hybrid modeling is massively parallel performance, a hybrid merge must be parallelizable. This chapter describes a hybrid merge algorithm satisfying these requirements and demonstrates its use in the next-generation hybrid plasma simulation code EMPIRE [38].

The remainder of this chapter is organized as follows: Section 4.1 describes a framework for hybrid merge algorithms for the Euler equations; Section 4.2 provides programming details on the implementation in EMPIRE; and Section 4.3 shows results of the use of hybrid merge according to various metrics.

4.1 Description of Methods

Many fluid closures are used throughout plasma physics, including asymptotic closures such as those of Braginskii [141] or Chapman and Enskog [142], and truncation closures that reduce the fluid equation hierarchy to the Navier-Stokes, Euler, or advection-diffusion equations. In this chapter, we restrict our purview to the Euler equations in d dimensions, which are given by (1.18), along with the ideal gas law (1.21). Throughout this chapter, we denote the drifting Maxwellian velocity distribution in $\mathbb{R}^d$ parametrized by number density n , bulk velocity $\mathbf{u}$, and temperature T by

$$f_M(d, n, \mathbf{u}, T; \mathbf{v}) \equiv n \left(\frac{m}{2\pi kT} \right)^{d/2} \exp \left(-\frac{m(\mathbf{v} - \mathbf{u})^2}{2kT} \right). \quad (4.1)$$

Consider a set of N macroparticles within a cell of the computational mesh along with an Eulerian fluid whose cell-averaged number density, bulk velocity, and temperature are n_f , $\mathbf{u}_f$, and T_f , respectively. After choosing which particles to merge into the fluid, the combined distribution of the fluid and chosen particles must satisfy the assumptions made by the fluid closure. The Euler equations do not explicitly assume the form of the fluid distribution; rather, the fluid equation hierarchy is closed by assuming that there is no

viscous stress or conductive heat flow. However, the maximum-entropy distribution satisfying this is the drifting Maxwellian distribution. It follows that it is always safe to require that the combined fluid and particle distribution be a drifting Maxwellian. Hence, the particles should be chosen such that the distribution f_p that they constitute satisfies

$$f_M(d, n_f, \mathbf{u}_f, T_f) + f_p = f_M(d, n_c, \mathbf{u}_c, T_c) \quad (4.2)$$

where the subscripts f, p, and c refer to the fluid, particle, and combined distributions, respectively. Since PIC particles are point particles, f_p is a sum of Dirac deltas in phase space, so exact equality is impossible. We will ultimately find that the error in this equation can be bounded such that the combined distribution thermalizes to a drifting Maxwellian distribution within one time step. We will elaborate on this point in subsequent sections.

The hybrid merge process is shown for a simple, one-dimensional example in Fig. 4.1. Ten particles are sampled from a drifting Maxwellian distribution and plotted as Dirac delta distributions alongside a background fluid distribution. Some weight is pulled from each particle and combined with the background fluid to form a drifting Maxwellian distribution with the same density, momentum, and total energy as the background fluid and merged particles.

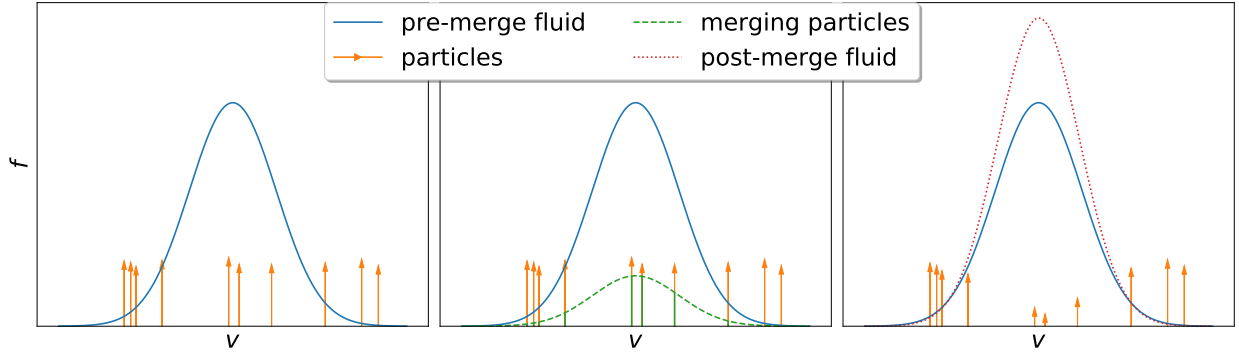


Figure 4.1: (a) A background fluid distribution and ten particles. Particles are represented by Dirac delta distributions in phase space, which are displayed here as arrows whose heights are given by the particle weights. (b) Some weight is pulled from each particle. Here, we visualize the merging particles using a drifting Maxwellian distribution. (c) The merged particles and fluid background combine to form a new fluid with greater mass and slightly perturbed bulk velocity and temperature.

Now, we must make a choice of parameters $\mathbf{u}_c$ and T_c . To simplify analysis, we may choose $\mathbf{u}_c = \mathbf{u}_f$ and $T_c = T_f$, which implies that the merged particles must have the same bulk velocity and temperature as the fluid. We refer to the resulting hybrid merge algorithm as the static fluid merge. Alternatively, we may allow the merged particles to reciprocally affect the combined distribution's bulk velocity and temperature. We refer to this as the dynamic fluid merge.

4.1.1 Static fluid

We constrain the combined bulk velocity and temperature to be equal to those of the fluid. This reduces (4.2) to

$$f_p = f_M(d, n_p, \mathbf{u}_f, T_f), \quad (4.3)$$

since $f_M(d, n_c, \mathbf{u}_f, T_f) = f_M(d, n_p, \mathbf{u}_f, T_f) + f_M(d, n_f, \mathbf{u}_f, T_f)$.

We can relax the strict equality in (4.3) somewhat. In a simulation, the combined distribution only needs to be close enough to a drifting Maxwellian to thermalize to one within one time step, given that the time step resolves characteristic simulation timescales such as the mean collision time and plasma frequency; otherwise, the fluid closure assumptions may not be satisfied for fluid solutions at subsequent time steps. This can be approximately quantified with some knowledge of the relaxation timescale τ_r by considering a BGK-like collision operator [143], given by

$$\frac{\partial f_c}{\partial t} = -\frac{1}{\tau_r} (f_c - f_M(d, n_c, \mathbf{u}_f, T_f)), \quad (4.4)$$

where f_c is the combined distribution at some time t after a merge. Note that mass, momentum, and energy conservation guarantee that the final, combined distribution will have number density n_c , bulk velocity $\mathbf{u}_f$, and temperature T_f . This roughly leads to the relation

$$\|f_c(t) - f_M(d, n_c, \mathbf{u}_f, T_f)\| = \|f_c(0) - f_M(d, n_c, \mathbf{u}_f, T_f)\| \exp\left(-\frac{t}{\tau_r}\right), \quad (4.5)$$

where $f_c(0)$ is the combined distribution immediately following the merge and $\|\cdot\|$ is a norm that we shall define later. In order to force the combined distribution to be within some tolerance ϵ of an ideal drifting Maxwellian after one time step Δt , it is therefore sufficient to choose f_p such that

$$\|f_p - f_M(d, n_p, \mathbf{u}_f, T_f)\| < \epsilon \exp \frac{\Delta t}{\tau_r}. \quad (4.6)$$

In principle, τ_r may be estimated locally from the state of the simulation or provided by the user. In Section 4.3, we elect to specify $\epsilon \exp \frac{\Delta t}{\tau_r}$ as a whole for convenience.

Obviously, it is impossible to verify that we have correctly chosen particles such that (4.3) is satisfied for all $\mathbf{v} \in \mathbb{R}^d$. Instead, we seek a reduced, finite set of constraints. Since the fluid deals exclusively with velocity-space moments of its distribution, it follows that we should verify whether the moments of the merged particles are close to those of $f_M(d, n_p, \mathbf{u}_f, T_f)$. In other words, it is sensible to take moments of (4.3). Given a length- d sequence of nonnegative integers (l_k) , define the kinetic moment of a distribution $f = f(\mathbf{v})$ as

$$M_{(l_k)}[f] = \int_{\mathbb{R}^d} \prod_{k=1}^d v_k^{l_k} f(\mathbf{v}) d\mathbf{v}. \quad (4.7)$$

Suppose that we have chosen to merge a weight x_j from each particle j . Note that we must constrain

$$0 \leq x_j \leq w_j \quad (4.8)$$

for all j , where w_j is the weight of particle j before the merge; that is, we must merge a nonnegative fraction of a particle, and possibly none of it, but we cannot merge in more weight than a particle initially has. Then

$$f_p(\mathbf{v}) = \sum_{j=1}^N \frac{x_j}{V} \delta(\mathbf{v} - \mathbf{v}_j), \quad (4.9)$$

where V is the volume of the mesh cell, so that

$$M_{(l_k)}[f_p] = \sum_{j=1}^N \frac{x_j}{V} \prod_{k=1}^d v_{jk}^{l_k}. \quad (4.10)$$

We can also compute

$$M_{(l_k)}[f_M(d, n, \mathbf{u}, T)] = n \prod_{k=1}^d \left(-\frac{1}{2} \sqrt{-\frac{2kT}{m}} \right)^{l_k} H_{l_k} \left(-u_k \sqrt{-\frac{m}{2kT}} \right), \quad (4.11)$$

where $H_q(\xi)$ is the physicists' Hermite polynomial of order q . (4.3) then implies that we should merge fractional particle weights x_j such that

$$\sum_{j=1}^N \frac{x_j}{V} \prod_{k=1}^d v_{jk}^{l_k} = \sum_{j=1}^N \frac{x_j}{V} \prod_{k=1}^d \left(-\frac{1}{2} \sqrt{-\frac{2kT_f}{m}} \right)^{l_k} H_{l_k} \left(-u_{fk} \sqrt{-\frac{m}{2kT_f}} \right), \quad (4.12)$$

for some choices of (l_k) .

To provide some options for (l_k) , we choose some positive integer n , then enforce (4.12) for all sequences (l_k) such that $\sum_{k=1}^d l_k \leq n$ and not all $l_k = 0$. This effectively enforces that all the particle moments of order up to and including n match those of the fluid. The moment of order 0 is the number density, which must be conserved. Similarly, the moments of order 1 form the momentum density, which must also be conserved. The moments of order 2 form the stress tensor, all of whose components are constrained either by conservation of energy or by the zero-viscosity assumption of the Euler equations. Finally, the conductive heat flux density, which is composed of moments of order 3, is constrained to be zero by the Euler equations. Hence, n should typically be chosen to be at least 3.

Each sequence (l_k) corresponds to a unique moment for which we would like to enforce (4.12). Since (4.12) is linear in the fractional particle weights, we can recast this system of linear equations as a matrix equation whose solution is subject to bounds given by (4.8). Suppose that there are M sequences (l_k) whose sum are less than n and let l_{ik} be the k -th component of the i -th such sequence. Recalling that N is the number of macroparticles within a cell, let A be the $M \times N$ matrix with entries

$$A_{ij} = \prod_{k=1}^d v_{jk}^{l_{ik}} - \prod_{k=1}^d \left(-\frac{1}{2} \sqrt{-\frac{2kT_f}{m}} \right)^{l_{ik}} H_{l_{ik}} \left(-u_{fk} \sqrt{-\frac{m}{2kT_f}} \right). \quad (4.13)$$

Then, with $\mathbf{x} = (x_1, x_2, \dots, x_N)^T$, (4.12) becomes

$$A\mathbf{x} = 0. \quad (4.14)$$

Adopting the norm $\|\cdot\| \equiv |\cdot|^2/2$, we can use previous analysis to relax this strict equality:

$$\frac{1}{2}|A\mathbf{x}|^2 < \epsilon \exp \frac{\Delta t}{\tau_r}. \quad (4.15)$$

The goal of the merge is to move as much weight from the particles as possible without violating (4.15).

Hence, the problem is reduced to a variation on a constrained linear least-squares problem:

$$\begin{aligned} & \text{maximize } \mathbf{x} \\ & \text{subject to the constraints } 0 \leq \mathbf{x} \leq \mathbf{w} \\ & \text{and } \frac{1}{2} |A\mathbf{x}|^2 < \epsilon \exp \frac{\Delta t}{\tau_r}. \end{aligned} \quad (4.16)$$

This problem is referred to as *blf*(2) in [144].

Practically, using A as defined by (4.13) in the problem (4.16) is problematic as its entries span many scales. Some normalization is required in order to avoid preferring the constraint imposed by any one row over another. This can be accomplished by defining the maximum particle speed $s = \max_{1 \leq j \leq N} v_j$, then normalizing A by s in (4.13), which yields

$$A_{ij} = \prod_{k=1}^d \left(\frac{v_{jk}}{s} \right)^{l_{ik}} - \prod_{k=1}^d \left(-\frac{1}{2s} \sqrt{-\frac{2kT_f}{m}} \right)^{l_{ik}} H_{l_{ik}} \left(-u_{fk} \sqrt{-\frac{m}{2kT_f}} \right). \quad (4.17)$$

This effectively nondimensionalizes the entries of A .

4.1.2 Dynamic fluid

In Section 4.1.1, we required that the merged particles approximately form a drifting Maxwellian distribution with the same bulk velocity and temperature as the fluid. However, in cases where the fluid has little mass compared to the particles and a bulk velocity or temperature that is far from that of the particles, this requirement means little particle weight will be merged. This constraint was imposed because, otherwise, the right-hand side of (4.2) is a nonlinear function of the particle weights. For performance, we would like to avoid nonlinear optimization. Rather than impose this constraint, we could instead directly take moments of (4.2) and linearize the right-hand side. Intuitively, this gives the particles some power in deciding the ultimate shape of the fluid distribution.

For some suitable sequence (l_k) , moments of (4.2) are given by

$$\sum_{j=1}^N \frac{x_j}{V} \prod_{k=1}^d v_{jk}^{l_k} + M_{(l_k)}[f_M(d, n_f, \mathbf{u}_f, T_f)] = M_{(l_k)}[f_M(d, n_c, \mathbf{u}_c, T_c)]. \quad (4.18)$$

The number density, bulk velocity, and temperature of the combined distribution can be computed as

$$n_c = n_f + n_p, \quad (4.19a)$$

$$\mathbf{u}_c = \frac{n_f \mathbf{u}_f + n_p \mathbf{u}_p}{n_f + n_p}, \quad (4.19b)$$

$$\frac{kT_c}{m} = \frac{n_f \frac{kT_f}{m} + n_p \frac{kT_p}{m}}{n_f + n_p} + \frac{\gamma - 1}{2} \frac{n_f n_p}{(n_f + n_p)^2} (\mathbf{u}_f - \mathbf{u}_p)^2. \quad (4.19c)$$

We linearize (4.18) by assuming $x_j \ll n_f V$ for each particle j ; that is, we assume that the weight of each macroparticle that will be merged into the fluid is small compared to the number of physical particles represented by the fluid in the volume of interest. Note that this assumption does not necessarily imply that $n_p \ll n_f$ because the particle density should be well resolved by $N \gg 1$ particles. Hence, the assumption can hold even when $n_p > n_f$ if N is sufficiently large, since $x_j \leq w_j \approx n_p V / N$. Future work will investigate the

limit where $x_j \gg n_f V$ and attempt to bridge the transition regime.

Let $y_j = x_j / n_f V$ and $\mathbf{y} = (y_1, y_2, \dots, y_N)^T$. Then (4.18) becomes

$$\sum_{j=1}^N \frac{x_j}{V} \prod_{k=1}^d v_{jk}^{l_k} + M_{(l_k)}[f_M(d, n_f, \mathbf{u}_f, T_f)] \approx M_{(l_k)}[f_M(d, n_c, \mathbf{u}_c, T_c)]|_{\mathbf{y}=\mathbf{0}} + \sum_{j=1}^N y_j \frac{\partial M_{(l_k)}[f_M(d, n_c, \mathbf{u}_c, T_c)]}{\partial y_j} |_{\mathbf{y}=\mathbf{0}}. \quad (4.20)$$

Since $M_{(l_k)}[f_M(d, n_c, \mathbf{u}_c, T_c)]|_{\mathbf{y}=\mathbf{0}} = M_{(l_k)}[f_M(d, n_f, \mathbf{u}_f, T_f)]$ (i.e., when all the particle weights are zero, the moments of the combined distribution reduce to those of the fluid), (4.20) becomes

$$\sum_{j=1}^N \frac{x_j}{V} \prod_{k=1}^d v_{jk}^{l_k} \approx \sum_{j=1}^N y_j \frac{\partial M_{(l_k)}[f_M(d, n_c, \mathbf{u}_c, T_c)]}{\partial y_j} |_{\mathbf{y}=\mathbf{0}}. \quad (4.21)$$

Define

$$B_j = \frac{\gamma - 1}{2} \frac{m}{2kT_f} (\mathbf{v}_j - \mathbf{u}_f)^2, \quad (4.22a)$$

$$C_j = \sum_{k=1}^d l_k \left(\frac{1}{2} - B_j \right), \quad (4.22b)$$

$$D_{jk} = -l_k \left[-v_{jk} + \left(\frac{1}{2} + B_j \right) u_{fk} \right]. \quad (4.22c)$$

The RHS of (4.21) can be explicitly evaluated as

$$\begin{aligned} \frac{\partial M_{(l_k)}[f_M(d, n_c, \mathbf{u}_c, T_c)]}{\partial y_j} |_{\mathbf{y}=\mathbf{0}} &= n_f (1 - C_j) \prod_{k=1}^d \left(-\frac{1}{2} \sqrt{-\frac{2kT_f}{m}} \right)^{l_k} H_{l_k} \left(-u_{fk} \sqrt{-\frac{m}{2kT_f}} \right) \\ &+ n_f \sum_{k=1}^d D_{jk} \left[\left(-\frac{1}{2} \sqrt{-\frac{2kT_f}{m}} \right)^{l_k-1} H_{l_k-1} \left(-u_{fk} \sqrt{-\frac{m}{2kT_f}} \right) \right. \\ &\quad \left. \times \prod_{\substack{q=1 \\ q \neq k}}^d \left(-\frac{1}{2} \sqrt{-\frac{2kT_f}{m}} \right)^{l_q} H_{l_q} \left(-u_{fq} \sqrt{-\frac{m}{2kT_f}} \right) \right]; \end{aligned} \quad (4.23)$$

see Appendix B. As in Section 4.1.1, choose a positive integer n and let l_{ik} be the k -th component of the i -th sequence such that $0 < \sum_{k=1}^d l_{ik} \leq n$. This will effectively enforce moment equality for all moments of order between 1 and n . From discussions in Section 4.1.1, n should again be chosen to be at least 3. Suppose that there are M such sequences. With N macroparticles, let A be the $M \times N$ matrix with entries

$$A_{ij} = \prod_{k=1}^d v_{jk}^{l_{ik}} - \frac{1}{n_f} \frac{\partial M_{(l_{ik})}[f_M(d, n_c, \mathbf{u}_c, T_c)]}{\partial y_j} |_{\mathbf{y}=\mathbf{0}}. \quad (4.24)$$

Then (4.21) becomes

$$A\mathbf{x} \approx 0. \quad (4.25)$$

As in Section 4.1.1, it is appropriate to nondimensionalize the entries of A by $s = \max_{1 \leq j \leq N} v_j$ as

$$A_{ij} = \prod_{k=1}^d \left(\frac{v_{jk}}{s} \right)^{l_{ik}} - \frac{1}{n_f} \frac{\partial M_{(l_{ik})}[f_M(d, n_c, \mathbf{u}_c, T_c)]}{\partial y_j} \Big|_{\mathbf{y}=\mathbf{0}} \prod_{k=1}^d \frac{1}{s^{l_{ik}}}. \quad (4.26)$$

Since we have assumed $x_j \ll n_f V$ for each particle j , we can choose a small $c > 0$ and require that $x_j \leq cn_f V$. Let $\mathbf{z}$ be such that

$$z_j = \min(w_j, cn_f V). \quad (4.27)$$

The problem then reduces to a variation on Problem (4.16) with A defined in (4.26). The new problem is

$$\begin{aligned} & \text{maximize } \mathbf{x} \\ & \text{subject to the constraints } 0 \leq \mathbf{x} \leq \mathbf{z} \\ & \text{and } \frac{1}{2} |A\mathbf{x}|^2 < \epsilon \exp \frac{\Delta t}{\tau_r}, \end{aligned} \quad (4.28)$$

which is still a problem of type $blf(2)$.

4.2 Implementation Details

The methods described in the preceding section were implemented in EMPIRE [38]. EMPIRE is built using the Kokkos performance portability ecosystem [42], [145], [146], which allows a single source code to be portable to most computing architectures in high-performance computing. In particular, this means that the hybrid merge methods presented here are suitable for parallelism on heterogeneous computing platforms with device acceleration, including general-purpose graphics processing units (GPGPU's).

EMPIRE uses a discontinuous Galerkin method for solving the Euler equations [147], [148]. For the hybrid merge method, we approximate the fluid quantities as piecewise constant, using the cell-averaged quantities as the constant value within mesh cells. The merge algorithm is run separately in each mesh cell, attempting to merge only the particles contained within that cell. Since no communication is required between merges in different cells, the hybrid merge is intrinsically parallelizable.

Several tuning parameters are optionally exposed to the user. The user may specify the tolerance $\epsilon \exp \frac{\Delta t}{\tau_r}$ in (4.16). Generally, a greater tolerance speeds up the $blf(2)$ solve at the expense of potentially greater error. The user may also specify the maximum order of the moments used in (4.17) and (4.26). All unique moments of order up to and including the specified maximum order are used in (4.17) and (4.26). Determinations of good default values for these parameters are discussed in Section 4.3.

In order to use kinetic moments, we must be able to compute the moments of the fluid distribution according to (4.11). To speed up computation of the Hermite polynomials on accelerators, the polynomial coefficients are precomputed during the simulation setup and stored. Taking $H_p(x) = \sum_{k=0}^p a_{p,k} x^k$, the polynomial coefficients can be computed using the recurrence relation

$$a_{p+1,k} = \begin{cases} -a_{p,k+1} & k = 0, \\ 2a_{p,k-1} - (k+1)a_{p,k+1} & k > 0, \end{cases} \quad (4.29)$$

with $a_{0,0} = 1$, $a_{1,0} = 0$, $a_{1,1} = 2$. The user-provided maximum moment order enforces an upper bound on p , ensuring that the coefficient computation need be done only once.

Problems (4.16) and (4.28) can be solved using a bifurcation search with a constrained linear least-squares solver. This approach is described in [144]. We also use the bounded-variable least-squares algorithm [144] as our constrained linear least-squares solver. This algorithm requires the use of any unconstrained linear least-squares solver. We use an unconstrained linear least-squares solver based on the singular value decomposition (SVD) [149]. Kokkos provides utilities for computing the SVD of matrices, among other matrix utilities, from within parallel compute regions through the Kokkos kernels toolkit.

The solutions to Problems (4.16) and (4.28) are vectors containing the weights of particles that should be merged into the fluid. Iterating through a solution, if an entry is less than some user-specified minimum weight $w_{\min}$, nothing is done with the corresponding particle; otherwise, increment the fluid quantities in the containing mesh cell by the moments of the corresponding particle. That is, let

$$\Delta(mn) = \sum_{j=1}^N \frac{x_j m}{V}, \quad (4.30a)$$

$$\Delta(mn\mathbf{u}) = \sum_{j=1}^N \frac{x_j m \mathbf{v}_j}{V}, \quad (4.30b)$$

$$\Delta(mnE) = \sum_{j=1}^N \frac{x_j m v_j^2}{2V}, \quad (4.30c)$$

where V is the volume of the mesh cell. Then increase the cell-averaged density, momentum density, and total energy density of the fluid in that cell by $\Delta(mn)$, $\Delta(mn\mathbf{u})$, and $\Delta(mnE)$, respectively. Correspondingly decrement the weight of each particle. If the weight of a particle after the merge is less than $w_{\min}$, which occurs when an entry in the solution vector is at its upper bound, destroy or otherwise disable the particle and increment the fluid quantities by the full weight of the particle. This scheme enforces conservation of mass, momentum, and energy.

In practice, we have found that the algorithm performs optimally when less than 1000 particles are available. In order to remain robust to larger problems, our implementation splits up the particle population randomly into batches of some user-specified number of particles. The algorithm is then performed separately on each such batch.

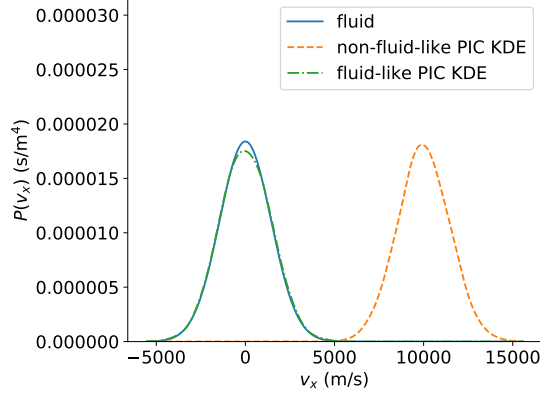
Note that the hybrid merge need not be performed every time step in a simulation. The user may specify a time step interval between merges.

4.3 Results

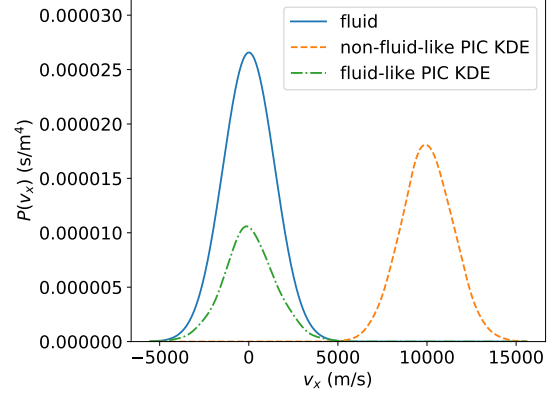
Ideally, the hybrid merge should only merge in particles that approximately satisfy the fluid closure assumptions. In addition, it should not skew the underlying fluid away from its own closure assumptions. This section explores the utility of the method described in Section 4.1 according to these metrics as used in EMPIRE. In so doing, we experimentally find reasonable default values for the user-provided parameters described in Section 4.2.

4.3.1 Particle selectivity

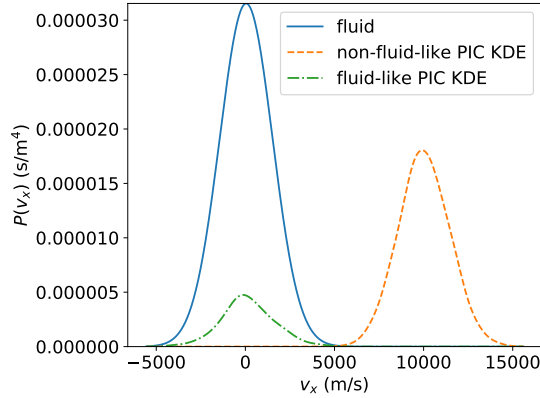
In this section, we investigate how well the hybrid merge algorithms select only those particles that are most fluid-like to merge. The canonical problem of interest has a fluid and two particle populations, initially



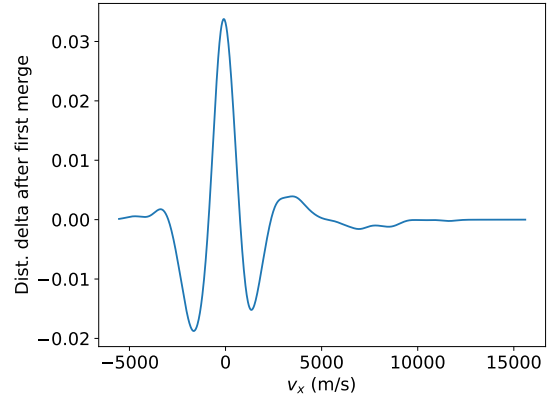
(a) The initial distributions.



(b) Distributions after one hybrid merge.



(c) Distributions after applying hybrid merge five times.



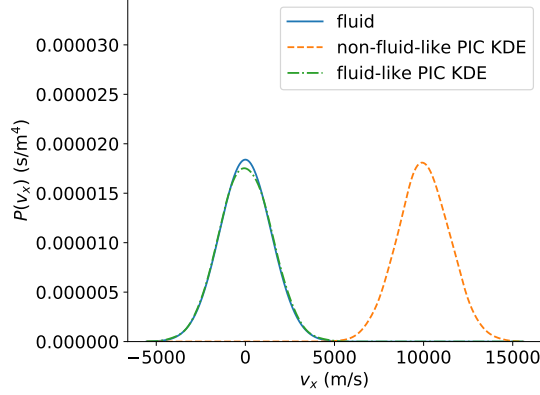
(d) Normalized difference between combined distributions before and after the first hybrid merge.

Figure 4.2: Three plots of the projections of the kernel density estimators for the particle populations and the fluid’s drifting Maxwellian distribution, and the normalized delta of the combined projections due to the static fluid hybrid merge.

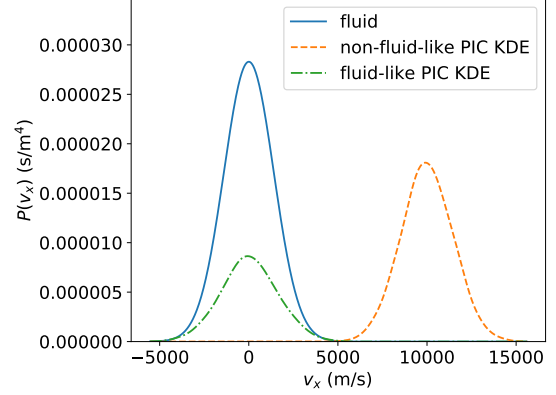
drifting Maxwellian. One of the particle populations has the same bulk velocity and temperature as the fluid, while the other particle population has varying parameters. We expect that the former population should be readily merged into the fluid, while the latter population remains relatively untouched. This section is intended only to demonstrate the selectivity of the hybrid merge algorithms; further investigation into the effects of varying the user-provided parameters will be conducted in Section 4.3.2

For simplicity, the problem is defined in a periodic mesh, replicating a spatially zero-dimensional problem. We choose neutral helium as the species. No collisions are enabled in PIC, as the problem is intended solely to judge how well the hybrid merge selects particles. Both the fluid and fluid-like particle population are parametrized with zero bulk velocity. Two different cases for the other particle population are considered: a bulk velocity of 10 km/s, such that there is very little overlap in the distributions; and a bulk velocity of 2.5 km/s, such that there is some overlap in the distributions. Each of these three distributions is initialized with number density 10^{25} m^{-3} (for a total number density of $3 \times 10^{25} \text{ m}^{-3}$) and temperature 1000 K, and each particle population is initialized with 10^5 particles.

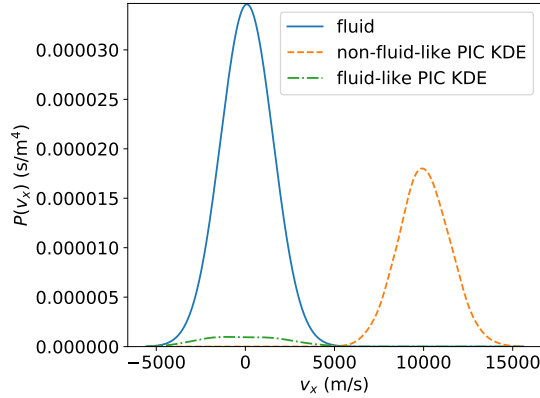
The hybrid merge is performed in this problem five times, after which subsequent merges have increasingly negligible effect for this problem. We have found a good balance between performance and accuracy can



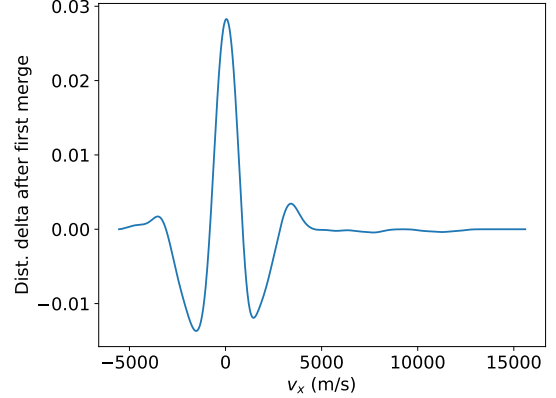
(a) The initial distributions.



(b) Distributions after one hybrid merge.



(c) Distributions after applying hybrid merge five times.



(d) Normalized difference between combined distributions before and after the first hybrid merge.

Figure 4.3: Three plots of the projections of the kernel density estimators for the particle populations and the fluid's drifting Maxwellian distribution, and the normalized delta of the combined projections due to the dynamic fluid hybrid merge.

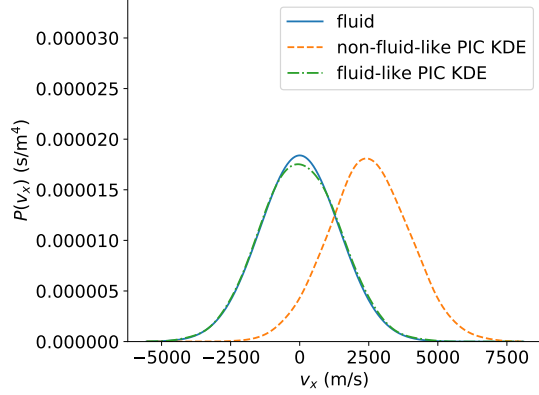
generally be obtained by setting an error tolerance such that $\epsilon \exp \frac{\Delta t}{\tau_r} = 10^{-5}$, a maximum moment order of 4, and a particle batch size of 500. For the dynamic fluid hybrid merge, we choose $c = 0.01$ in (4.27).

Kernel density estimators of each of the particle populations are separately computed using Gaussian kernels and bandwidths from Silverman's rule of thumb [150], which is suitable for approximately Gaussian data. Note that the hybrid merge does not use kernel density estimators, and these are computed primarily to visualize the particle distributions. The projections of each of these distributions onto the v_x -axis is then plotted. This projection is computed generally as

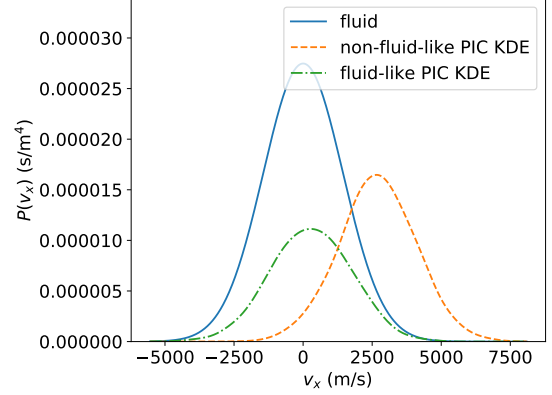
$$P(v_x) \equiv \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} dv_y dv_z f(\mathbf{v}), \quad (4.31)$$

for a distribution f .

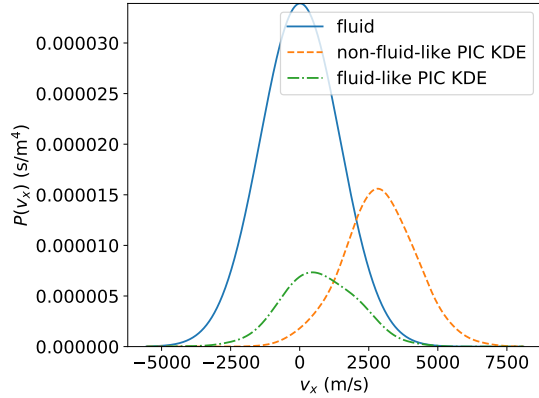
The v_x -projections of the two kernel density estimators and the drifting Maxwellian distribution corresponding to the fluid are plotted before the first hybrid merge, after the first hybrid merge, and after five hybrid merges, for each of the test cases. In addition, the difference between the sum of all three projections before and after the first hybrid merge, normalized by the maximum value of the sum of all three projections



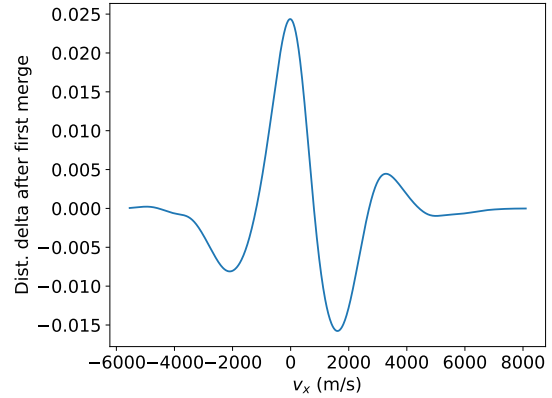
(a) The initial distributions.



(b) Distributions after one hybrid merge.



(c) Distributions after applying hybrid merge five times.



(d) Normalized difference between combined distributions before and after the first hybrid merge.

Figure 4.4: Three plots of the projections of the kernel density estimators for the particle populations and the fluid's drifting Maxwellian distribution, and the normalized delta of the combined projections due to the static fluid hybrid merge.

before the merge, is plotted. This latter plot is intended to impart some intuition for how much the hybrid merge changes the total distribution representing the species. Figs. 4.2 and 4.3 shows these four plots for the first case, where the second particle population does not appreciably overlap with the fluid, for the static and dynamic fluid algorithms, respectively. Figs. 4.4 and 4.5 shows them for the second case, where the second particle population overlaps somewhat more with the fluid, for the static and dynamic fluid algorithms, respectively. Note that the hybrid merge perturbs the combined distribution by less than 6% in all cases. The dynamic fluid algorithm merges more particles, as can be seen by comparing Figs. 4.5b and 4.4b. However, this comes at the cost of greater max error, as can be seen by comparing Figs. 4.5d and 4.4d. We shall further explore the differences between the algorithms in Section 4.3.2.

The time evolution of four kinetic moments in v_x are also plotted for each of the test cases. The particle moments are computed following (4.10) as

$$M_{n00}[f_p] = \frac{1}{V} \sum_{j=1}^N w_j v_{j,x}^n, \quad (4.32)$$

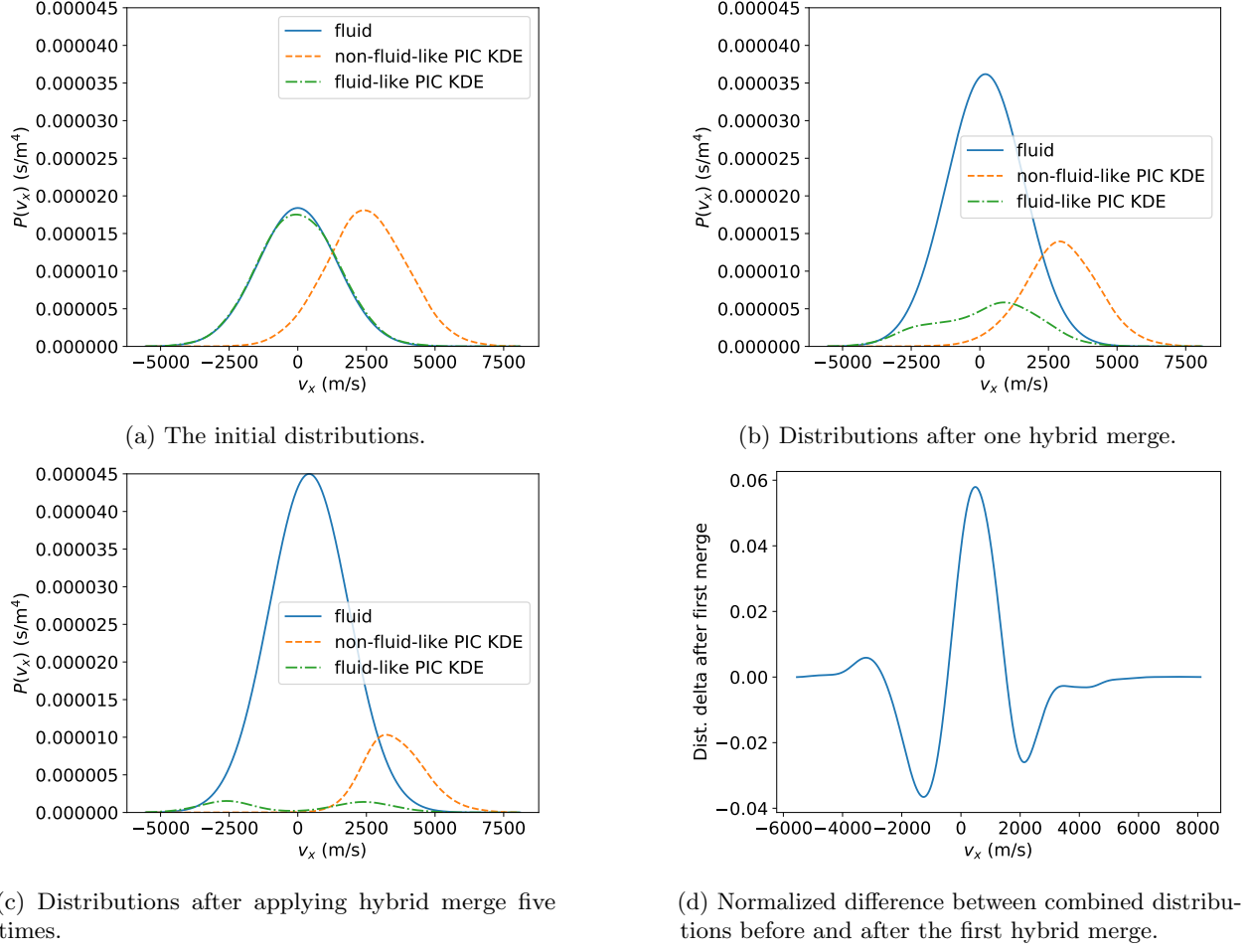


Figure 4.5: Three plots of the projections of the kernel density estimators for the particle populations and the fluid’s drifting Maxwellian distribution, and the normalized delta of the combined projections due to the dynamic fluid hybrid merge.

where V is the volume of the domain. The fluid moments are computed following (4.11) as

$$M_{n00}[f_f] = n_f \left(-\frac{1}{2} \sqrt{-\frac{2kT_f}{m}} \right)^n H_n \left(-u_{f,x} \sqrt{-\frac{m}{2kT_f}} \right). \quad (4.33)$$

The combined moment is the sum of the particle and fluid moments, $M_{n00}[f_c] = M_{n00}[f_p] + M_{n00}[f_f]$. These are plotted as a function of the number of hybrid merge iterations for $n = 0, 1, 2, 3$. Since the hybrid merge should ideally not affect the combined moment, a horizontal line is drawn representing the ideal case where the combined moment is constant. The plots for the first test case are displayed for the static and dynamic algorithms in Figs. 4.6 and 4.7, respectively. The plots for the second test case are displayed for the static and dynamic algorithms in Figs. 4.8 and 4.9, respectively. The evolution of M^0 , equivalent to the number density, clearly shows the representation of the plasma transitioning from mostly particle-based to fluid-based. In all cases and for all moments, the combined moment is conserved well.

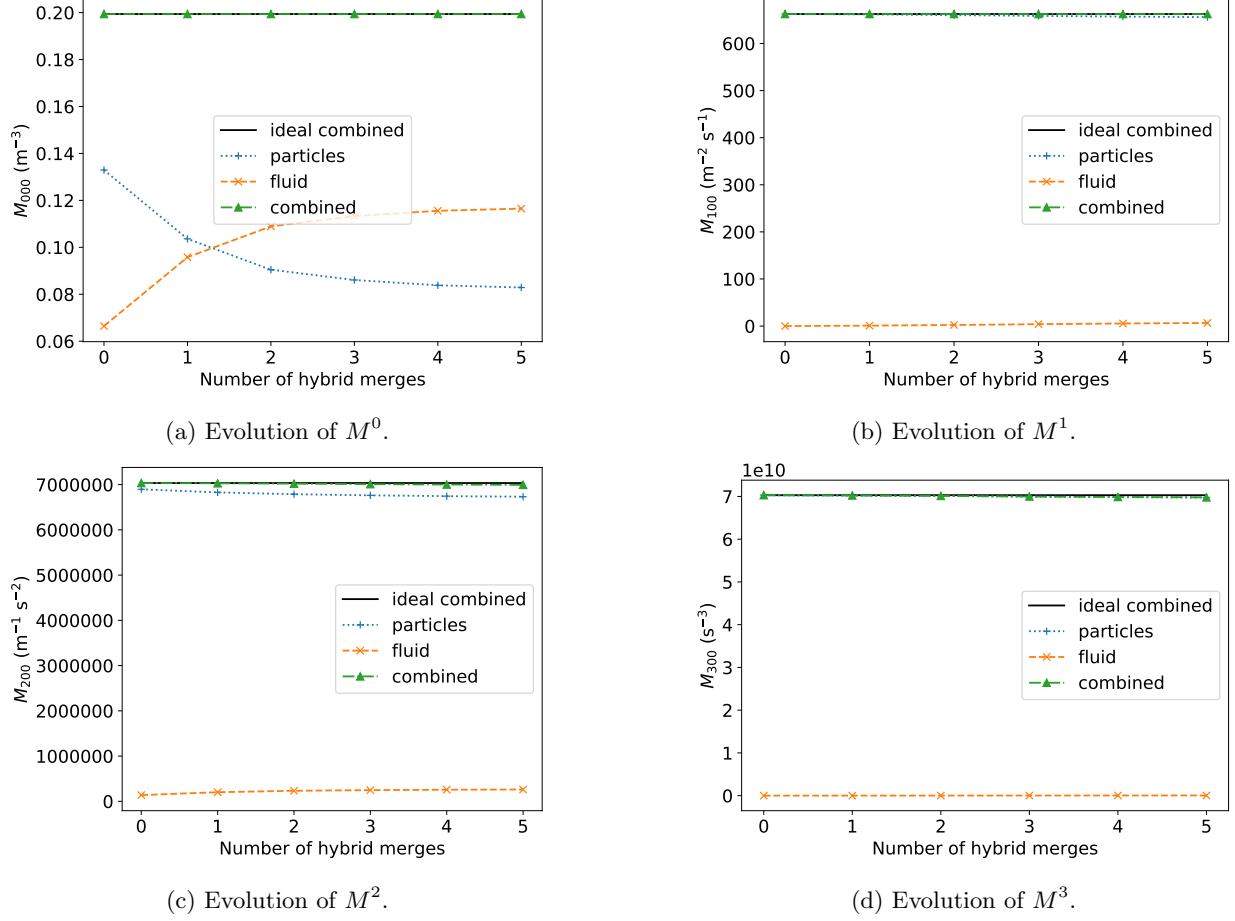


Figure 4.6: Evolution of four moments of the distributions as a function of the number of hybrid merge iterations applied for the first test case with static fluid hybrid merge.

4.3.2 Bobylev-Krook-Wu solution verification

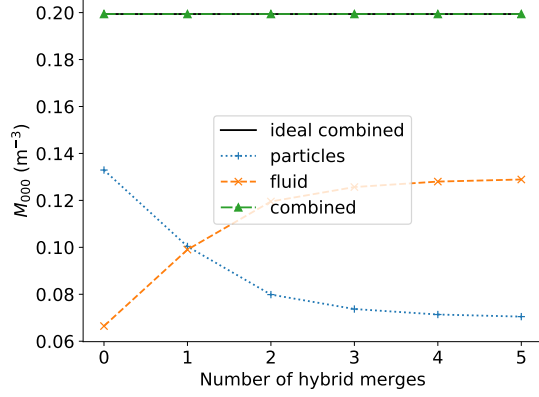
A hybrid merge should avoid merging particles in such a way that the plasma thermalizes quicker than expected. The Bobylev-Krook-Wu (BKW) solution [116], [117] to the collisional Boltzmann equation is commonly used for verification of collision models in PIC software [151]. In this section, we investigate the thermalization rate of a hybrid simulation of the BKW problem with hybrid merge activated and compare it to the analytic solution. We expect that the simulated distribution should match the BKW solution well, and the final state of the plasma should be mostly fluid, as the BKW solution asymptotes to a drifting Maxwellian distribution.

Following [117], the differential cross-section σ for elastic collisions is chosen to be given by

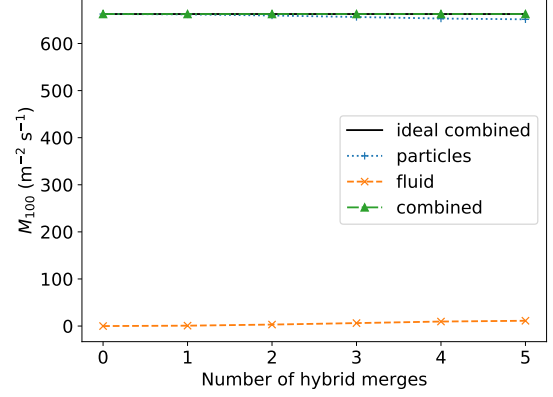
$$\sigma(g, \chi) = \frac{\kappa}{g}, \quad (4.34)$$

where κ is a constant, g is the relative speed of a colliding particle pair, and χ is the scattering angle, of which this cross-section is independent. The corresponding collision rate ν is given by

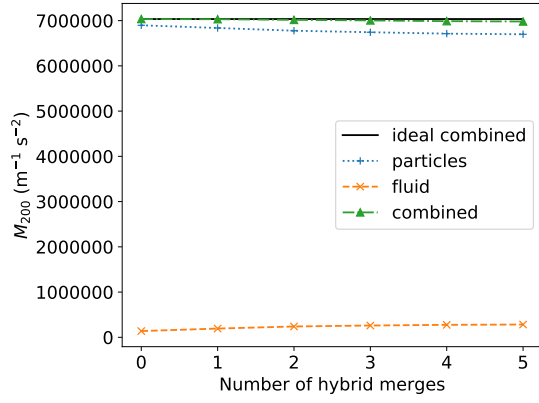
$$\nu = 4\pi n_c \kappa, \quad (4.35)$$



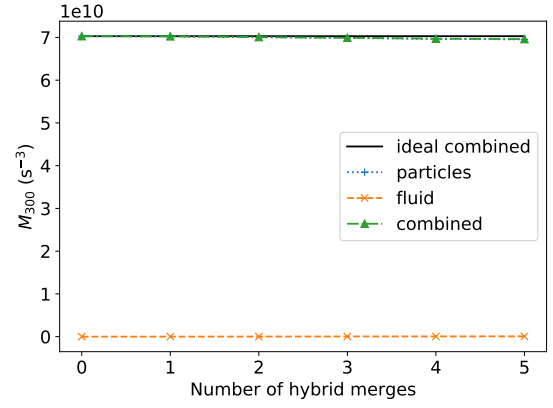
(a) Evolution of M^0 .



(b) Evolution of M^1 .



(c) Evolution of M^2 .



(d) Evolution of M^3 .

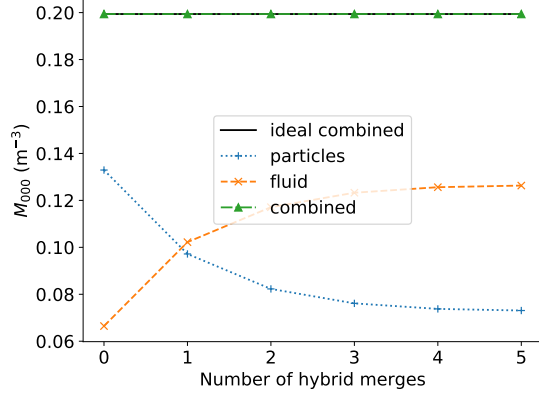
Figure 4.7: Evolution of four moments of the distributions as a function of the number of hybrid merge iterations applied for the first test case with dynamic fluid hybrid merge.

where n_c is again the combined number density of the fluid and particles. The mean collision time is the inverse of the collision rate, $\tau \equiv \nu^{-1}$. Collisions are enabled between two particles using a direct-simulation Monte Carlo (DSMC) method [52] and between a particle and the fluid using a hybrid null-collision Monte Carlo collisions (MCC) method as presented in Section 2.4.1. Fluid elastic self-collisions are inherent in the fluid closure assumptions and do not need to be explicitly considered. The hybrid merge method presented here is paired with the particle merge algorithm from [128].

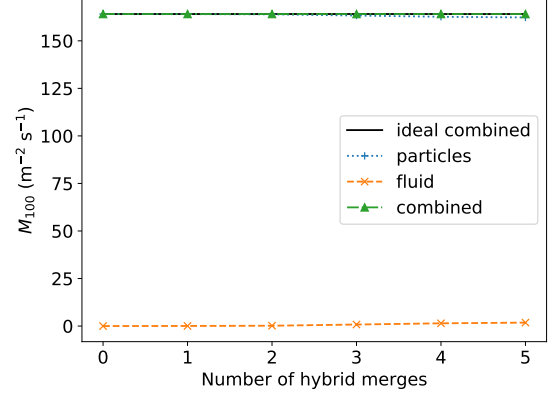
The final simulation time is taken to be 25τ with 5 time steps per mean collision time, for a total of 125 time steps. Both collision algorithms are performed every time step. The hybrid and particle merge algorithms are performed once per mean collision time.

The problem's mesh consists of a single quadrilateral cell with reflecting boundary conditions for both the particles and fluid. The dimensions of the domain are arbitrary since the collisional models used perform interactions between particles and fluid within each mesh cell, irrespective of cell size. A neutral species is chosen, so electromagnetic effects are ignored.

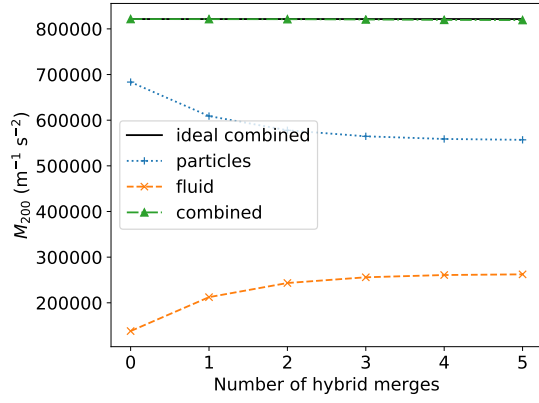
The simulation is initialized with a fluid of small number density with zero bulk velocity and temperature T to act as a “target” for the hybrid merge. The particles are sampled from the initial BKW distribution,



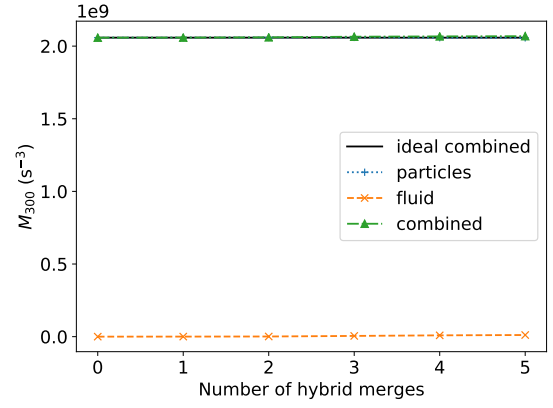
(a) Evolution of M^0 .



(b) Evolution of M^1 .



(c) Evolution of M^2 .



(d) Evolution of M^3 .

Figure 4.8: Evolution of four moments of the distributions as a function of the number of hybrid merge iterations applied for the second test case with static fluid hybrid merge.

given by

$$f_{\text{BKW}}(\mathbf{v}, t = t_0) = \frac{5}{9} \frac{mv^2}{kT} \left(-\frac{5m}{6\pi kT} \right)^{3/2} \exp \left(-\frac{5mv^2}{6kT} \right) \quad (4.36)$$

(note that [117] erroneously writes $K \geq 5/3$, which should be $K \geq 3/5$).

We will compare a centered fourth-order moment from the simulation to an analytic solution. This is because the moments of order up to and including 2 are trivially conserved and the centered moment of order 3 is zero. The moment used is computed from a general distribution f with bulk velocity $\mathbf{u}$ as

$$M^4 = \int_{\mathbb{R}^3} d\mathbf{v} (\mathbf{v} - \mathbf{u})^4 f(\mathbf{v}). \quad (4.37)$$

The fourth-order moment of the BKW distribution as a function of time is given by

$$M_{\text{BKW}}(t) = n \left(\frac{2kT}{m} \right)^2 \left[\frac{15}{4} - \frac{3}{5} \exp \left(-\frac{4}{3} \pi n \kappa t \right) \right]. \quad (4.38)$$

The corresponding fourth-order moment of the fluid in the simulation is computed at instants of time using

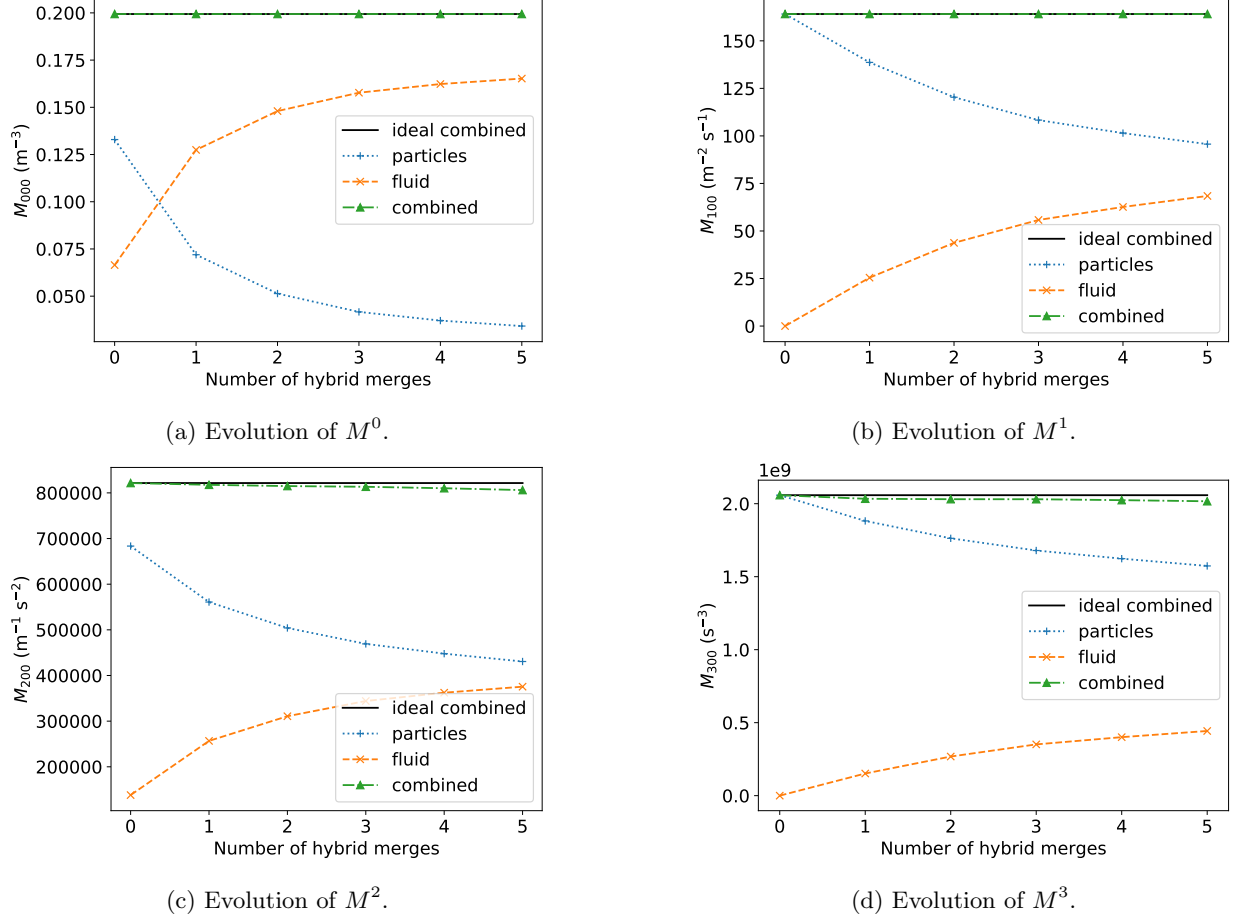


Figure 4.9: Evolution of four moments of the distributions as a function of the number of hybrid merge iterations applied for the second test case with dynamic fluid hybrid merge.

(4.11); the moment of the particle distribution is computed with

$$M_p^4 = \frac{1}{V} \sum_{i=1}^N w_i (\mathbf{v}_i - \mathbf{u}_p)^4. \quad (4.39)$$

Although the particles and fluid are ostensibly initialized with zero bulk velocity, particle noise skews the bulk velocity away from zero. We therefore found that we could not assume zero bulk velocity while post-processing.

For our problem, we model a neutral argon gas at a temperature of 273 K with initial particle and fluid number densities of $n_p = 2.687 \times 10^{25} \text{ m}^{-3}$ and $n_f = 2.687 \times 10^{21} \text{ m}^{-3}$, respectively. 10 000 macroparticles are initially loaded into the domain. We choose $\kappa = 2.83364 \times 10^{-19} \text{ m}^3 \text{ s}^{-1}$. For the particle merge, we choose to merge from 20 down to 2 particles at a time, and use 40 bins in each direction in momentum space [128]. The problem is run for several error tolerances such that $\epsilon \exp \frac{\Delta t}{\tau_r}$ varies from 10^{-6} to 10^{-3} . For each tolerance, the problem is run separately with the static fluid merge, the dynamic merge without a particle weight upper bound, and the dynamic merge with $c = 10^{-2}$ as in (4.27). In all test cases, we choose a maximum moment order of 4 and a particle batch size of 500.

Results are summarized in Figs. 4.10 and 4.11. Fig. 4.10 displays the relative L_2 error in the time evolution of M^4 for each algorithm as a function of the tolerance, which shows that the L_2 error is always less than

3% and tends to increase with increasing tolerance. Fig. 4.11 displays the ratio of the fluid to total mass at the last time step and at $t = 5\tau$ for each algorithm as a function of the tolerance. We emphasize that the tolerances in all cases guarantee that the skew from a drifting Maxwellian induced by the merged particles thermalizes within one time step.

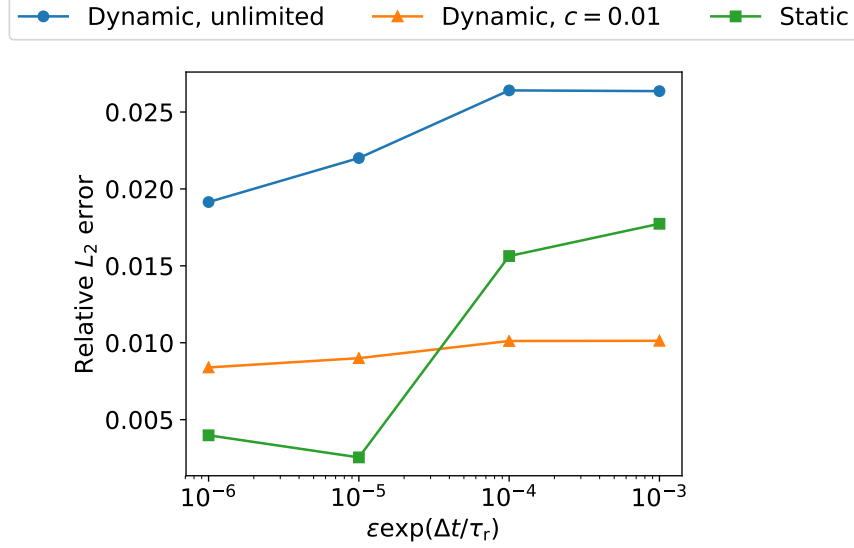


Figure 4.10: Relative L_2 error in M^4 with increasing $\epsilon \exp(\Delta t / \tau_r)$.

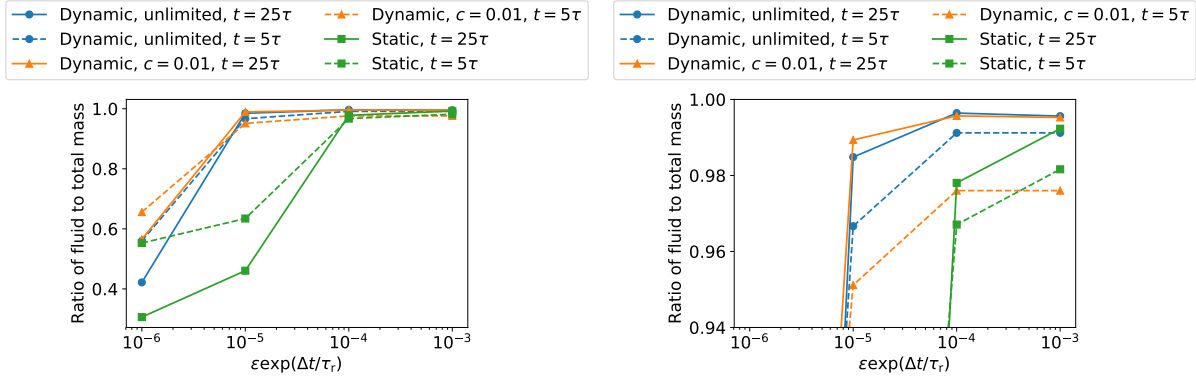


Figure 4.11: Fluid ratio with increasing $\epsilon \exp(\Delta t / \tau_r)$ at $t = 5\tau$ and $t = 25\tau$.

Let us contrast results from two particular cases. Consider the dynamic fluid merge with $c = 10^{-2}$ and the static fluid merge, both with $\epsilon \exp(\Delta t / \tau_R) \geq 10^{-5}$. Detailed test results for these cases are shown in Figs. 4.12 and 4.13, respectively. The first plot in each figure depicts the time evolution of the fluid, particle, and combined fourth-order moment alongside the analytic solution. The second plot in each figure shows the time evolution of the fluid, particle, and combined number densities, showing how hybrid merge moves mass from particles into the fluid, despite the fact that hybrid MCC continuously draws mass out of the fluid and into particles. The sawtooth pattern arises from the fact that hybrid MCC acts every time step while the hybrid and particle merges act only every five time steps. The third plot in each figure graphs the PIC particle count as a function of time step. The particle count plot exhibits the same sawtooth pattern as the number density plot for the same reason. For both algorithms, M^4 qualitatively matches the analytic solution excellently.

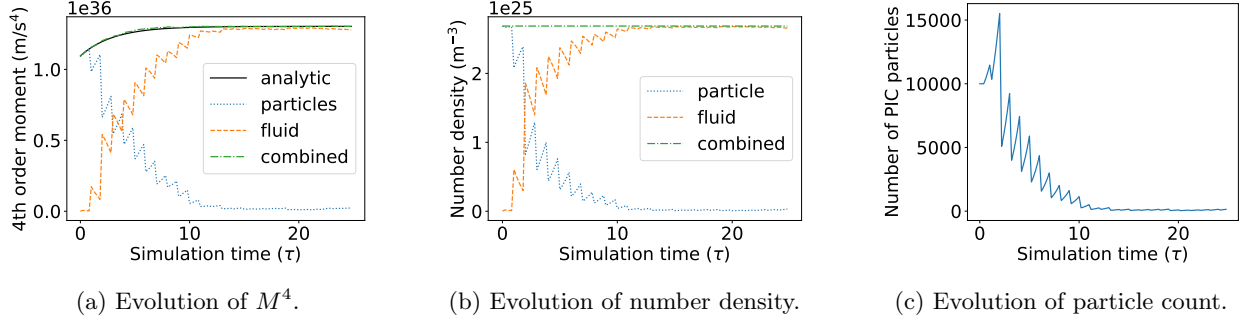


Figure 4.12: Case with $\epsilon \exp(\Delta t/\tau_r) = 10^{-5}$ using the dynamic fluid algorithm with $c = 10^{-2}$.

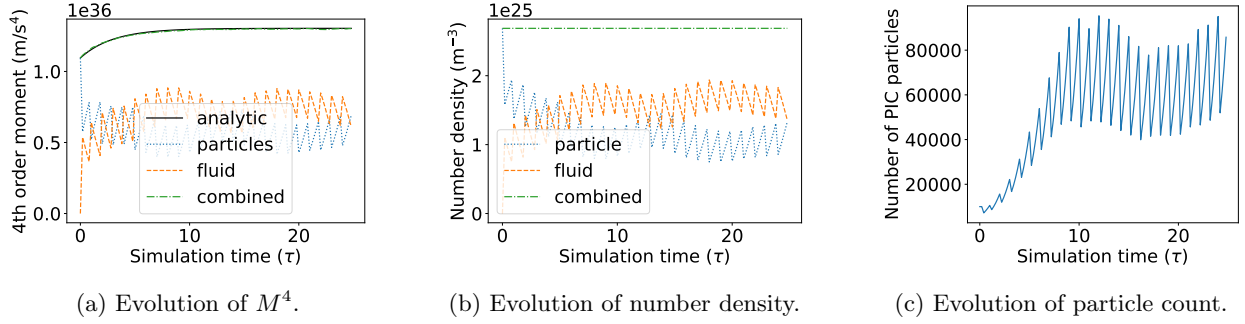


Figure 4.13: Case with $\epsilon \exp(\Delta t/\tau_r) = 10^{-5}$ using the static fluid algorithm.

However, while the dynamic fluid merge is ultimately able to drive most of the mass into the fluid, the static fluid merge struggles to merge sufficient particle mass into the fluid to counteract macroparticle generation from hybrid MCC. As a result, the dynamic fluid merge ends with very few remaining macroparticles and the static fluid merge ends with a macroparticle count in equilibrium with hybrid MCC, but much greater than at initialization. Note also that the dynamic fluid merge does not obviously immediately begin to do work. This is because the merged particle weights are bounded from above by the fluid density, which is initially very low. Eventually, sufficient particle mass is accumulated into the fluid by the hybrid merge to allow for more enthusiastic merges beginning at 2τ . In other words, the fact that merged particle weights are bounded by the fluid density effectively causes the hybrid merge to ramp up in intensity.

An ideal hybrid merge algorithm should end with most of the mass in the fluid and minimal error. The former goal is satisfied by the static fluid merge algorithm for $\epsilon \exp(\Delta t/\tau_R) \geq 10^{-4}$ and by the dynamic fluid merge algorithm for $\epsilon \exp(\Delta t/\tau_R) \geq 10^{-5}$, regardless of c . Among these choices, the dynamic fluid merge algorithm with $c = 10^{-2}$ maintains an L_2 error near 0.1. We therefore recommend this choice of parameters by default.

Chapter 5

Conclusions

Problem sizes in computational plasma physics have grown beyond the limits of the traditional kinetic or fluid approaches, despite advances in high-performance computing. To meet the need for models combining the strengths of fluid and kinetic descriptions, hybrid models must be developed. However, since these models describe the plasma in incompatible ways, it is not clear how they can be coupled.

Chapter 2 introduced a scheme where each species of a plasma can be independently modeled kinetically or as a fluid, dubbed hybrid-by-species. The scheme uses the venerable particle-in-cell method for the kinetic component and a discontinuous Galerkin method for the fluid component. Collisions are handled via Monte Carlo methods in the kinetic component and source terms incorporated into the moment equations for the fluid component. These methods are combined with a time integrator constructed using operator splitting. Together, these methods form a complete hybrid model for plasma simulations.

This hybrid-by-species scheme was implemented in the hPIC2 code, presented in Chapter 3. As a demonstration of algorithms suitable for supercomputing, hPIC2 was designed for and tested on typical modern high-performance computing architectures incorporating accelerators. This capability is furnished by the inclusion of the Kokkos performance portability library. With the MFEM finite element library, hPIC2 can simulate a vast variety of plasma conditions in complex domains. hPIC2 also optionally couples to the RustBCA code to handle plasma-material interactions. A rigorous test suite was written to ensure the correctness of the code, and selected verification studies were presented.

Single-species schemes were explored in Chapter 4, which presents methods for identifying subpopulations of PIC particles that may be well characterized as a fluid and converting them. The methods were tested on salient problems that demonstrate their ability to identify thermalized groups of particles, even when the species as a whole is far from equilibrium. The intended applications for these algorithms are simulations of electron beam propagation through background gases, wherein an initially kinetic electron beam thermalizes through collisions with the background gas.

5.1 Future Work

This work was primarily performed in the context of nonrelativistic, electrostatic plasmas. While, in principle, the hybrid-by-species scheme as presented in Chapter 2 extends to fully electromagnetic regimes, additional timescales are introduced that must be carefully examined. Such hybrid models would be useful in the study of laser wakefield acceleration, where plasma particles are driven to relativistic energies by lasers [152], and

astrophysics, where multiscale, relativistic plasmas abound [153].

In Chapter 4, a method for merging kinetic particles into a fluid was presented. The inverse of this problem is to create kinetic particles from a fluid. Though a fluid may initially be a good model for a portion of a plasma, various phenomena (collisions, in particular) may ultimately drive the fluid into a regime where its closure assumptions are violated. Since the fluid was constructed under these assumptions, it cannot independently verify whether it has violated them. In these cases, a method for identifying whether a fluid's closure assumptions may be violated and creating sufficiently many particles to return the fluid to its required regime is desirable. Such an algorithm would allow a user of a hybrid fluid-kinetic code to aggressively model the majority of a plasma as a fluid and trust that the code will adaptively convert the fluid to particles as needed.

Appendix A

Numerical Methods for Poisson's Equation

Poisson's equation is solved at some instant of time. We therefore suppress time dependence throughout this appendix for brevity.

A.1 Finite Difference Methods

The finite difference method (FDM) is derived by approximating the derivatives in (1.6) by finite differences. The domain is partitioned into a mesh, and the finite differences at each grid point depend only on the value of the solution at other grid points. Hence, the solution can be implicitly solved. For simplicity, we consider only uniform meshes with equal grid spacing h in all dimensions.

For the case of uniform meshes, we approximate second derivatives with the second-order central difference, which is given by

$$\frac{d^2 f}{dx^2} \approx \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}. \quad (\text{A.1})$$

Using $\phi_i = \phi(ih)$ and $\rho_i = \rho(ih)$ for one-dimensional problems, (1.6) becomes

$$\phi_{i-1} - 2\phi_i + \phi_{i+1} \approx -h^2 \frac{\rho_i}{\varepsilon_0}. \quad (\text{A.2})$$

Similarly, using $\phi_{k,l} = \phi(kh, lh)$ and $\rho_{k,l} = \rho(kh, lh)$ for two-dimensional problems, we have

$$\phi_{k-1,l} + \phi_{k,l-1} + \phi_{k+1,l} + \phi_{k,l+1} - 4\phi_{k,l} \approx -h^2 \frac{\rho_{k,l}}{\varepsilon_0}. \quad (\text{A.3})$$

Typically, the multi-index in higher-dimensional problems is flattened into a single index. Assuming there are N_x grid points in the x -direction, let $i = k + N_x l$. Then (A.3) can be rewritten as

$$\phi_{i-1} + \phi_{i-N_x} + \phi_{i+1} + \phi_{i+N_x} - 4\phi_i \approx -h^2 \frac{\rho_i}{\varepsilon_0}. \quad (\text{A.4})$$

Dirichlet boundary conditions are imposed by simply setting the values of ϕ at grid points on boundaries. For example, if a grid point $\mathbf{x}_i$ lies on a boundary upon which we impose a Dirichlet boundary condition

$\phi(\mathbf{x}) = f$, then

$$\phi_i = f \quad (\text{A.5})$$

enforces the boundary condition.

For Neumann boundary conditions, the normal derivative is approximated with a central finite difference, which is given by

$$\frac{df}{dx} \approx \frac{f(x+h) - f(x-h)}{2h}. \quad (\text{A.6})$$

In order to implement this, we introduce “ghost nodes” outside the domain. For example, if a Neumann boundary condition of the form $\hat{n} \cdot \nabla \phi(x) = g$ is imposed on the left side of a one-dimensional domain, we require

$$\phi_{-1} - 2\phi_0 + \phi_1 = -h^2 \frac{\rho_0}{\varepsilon_0}, \quad (\text{A.7a})$$

$$\phi_{-1} - \phi_1 = 2hg_0. \quad (\text{A.7b})$$

The ghost node term can be eliminated, yielding

$$-2\phi_0 + \phi_1 = -h^2 \frac{\rho_0}{\varepsilon_0} - 2hg_0, \quad (\text{A.8})$$

and correspondingly for the right side of the domain and for edges in two-dimensional domains. Corners in two-dimensional domains require special treatment, since the normal derivative is not well-defined. Instead, we use the average of the normal derivatives from the two edges. As an example, consider the bottom-left corner of a square domain. Suppose that we impose the boundary conditions

$$\hat{n} \cdot \nabla \phi = g^b, \quad (\text{A.9a})$$

$$\hat{n} \cdot \nabla \phi = g^l, \quad (\text{A.9b})$$

on the bottom and left edges, respectively. Then, using the multi-index notation for clarity, we require

$$\phi_{-1,0} + \phi_{0,-1} + \phi_{1,0} + \phi_{0,1} - 4\phi_{0,0} = -h^2 \frac{\rho_{0,0}}{\varepsilon_0}, \quad (\text{A.10a})$$

$$\phi_{-1,0} - \phi_{1,0} = 2hg_{0,0}^l, \quad (\text{A.10b})$$

$$\phi_{0,-1} - \phi_{0,1} = 2hg_{0,0}^b. \quad (\text{A.10c})$$

Again, the ghost node terms can be eliminated, yielding

$$2\phi_{1,0} + 2\phi_{0,1} - 4\phi_{0,0} = -h^2 \frac{\rho_{0,0}}{\varepsilon_0} - 2hg_{0,0}^l - 2hg_{0,0}^b. \quad (\text{A.11})$$

Neumann boundary conditions on other corners are implemented similarly.

With the solution ϕ_i , the electric field at each grid point is computed using a second-order central difference as

$$\mathbf{E}_i = \left(\frac{\phi_{i-1} - \phi_{i+1}}{2h}, 0, 0 \right) \quad (\text{A.12})$$

in one dimension and

$$\mathbf{E}_i = \left(\frac{\phi_{i-1} - \phi_{i+1}}{2h}, \frac{\phi_{i-N_x} - \phi_{i+N_x}}{2h}, 0 \right) \quad (\text{A.13})$$

in two dimensions in the interior of the domain. On boundary nodes, the electric field is computed using second-order one-sided finite differences.

A.2 Finite Element Methods

In this section, we provide a brief introduction to the application of the finite element method (FEM) to our electrostatic PIC problem. As we do not intend to give a definitive description of the finite element method, many details are omitted for brevity or clarity of presentation. We have endeavored to use standard notation as often as possible [154]. Since, Poisson's equation is solved at some instant of time, we suppress time dependence throughout this section for brevity.

Rather than directly solve (1.6) in some polyhedral domain Ω , the finite element method seeks weak solutions $u \in V \subset H^1(\Omega)$, where $H^1(\Omega)$ is the first-order Sobolev space of functions defined on Ω and V is a subspace that is also a Hilbert space. We have omitted discussion of the boundary conditions as the details of including boundary conditions in finite element solutions are beyond the scope of this work. It is assumed that functions in V implicitly obey the desired boundary conditions. The weak formulation of (1.6) is written as

$$\varepsilon_0 \int_{\Omega} d\mathbf{x} \nabla u \cdot \nabla v = \int_{\Omega} d\mathbf{x} \rho v \quad (\text{A.14})$$

for all $v \in V$.

Since this problem is still computationally intractable, we instead seek weak solutions u_h in a finite-dimensional subspace $S_h \subset V$. Let $\{\psi_1, \psi_2, \dots, \psi_M\}$ be a basis for S_h . Then u_h can be written as

$$u_h = \sum_{i=1}^M z_i \psi_i. \quad (\text{A.15})$$

Defining the $M \times M$ matrix $\mathbf{A}$ and vector $\mathbf{b}$ such that

$$\begin{aligned} A_{ij} &= \varepsilon_0 \int_{\Omega} d\mathbf{x} \nabla \psi_i \cdot \nabla \psi_j, \\ b_i &= \int_{\Omega} d\mathbf{x} \rho \psi_i, \end{aligned}$$

the solution to the matrix problem

$$\mathbf{A}\mathbf{z} = \mathbf{b}, \quad (\text{A.16})$$

where $\mathbf{z} = (z_1, \dots, z_M)^t$, determines the coefficients for which u_h is the desired weak solution.

The choice of the finite element space S_h practically begins by partitioning Ω into a mesh $\mathcal{T}_h$ of polyhedral subdomains $T_i \in \mathcal{T}_h$, called elements. The mesh is required to be admissible, which is true if

1. $\overline{\Omega} = \bigcup_{T_i \in \mathcal{T}_h} T_i$;
2. the interiors of elements do not overlap; and,
3. there are no hanging vertices, edges, or faces.

By a hanging vertex, we refer to a point that is a vertex of one element but not a vertex of another element, and correspondingly for hanging edges and faces.

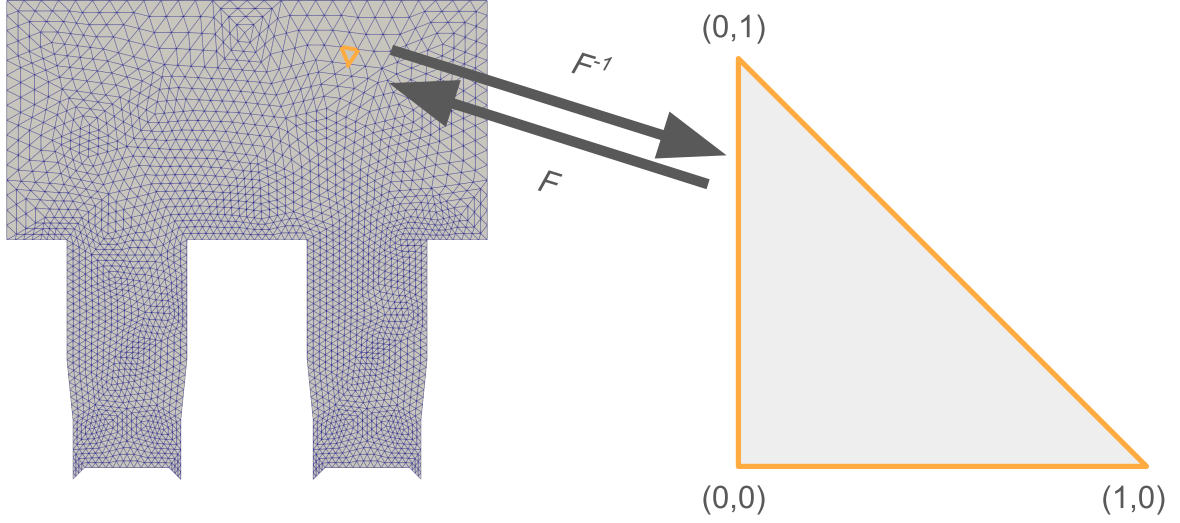


Figure A.1: An example of a reference element and its associated mappings. This example illustrates the use of the interior of a standard simplex as the reference element for triangular mesh elements. Integration over the reference element is simpler to perform, since quadrature rules are readily available. Note that every mesh element is associated with a mapping to the reference element.

A finite element is an element of a mesh imbued with a space of functions Π having finite dimension s and a set Σ containing s linearly independent functionals on Π , such that each $p \in \Pi$ is uniquely determined by the values of each functional in Σ on p . Typically, these functionals evaluate functions or their derivatives at points, so Σ is called the set of interpolation conditions.

It is usually convenient to define a single reference finite element $(T_{\text{ref}}, \Pi_{\text{ref}}, \Sigma)$. The finite element solver in hPIC2 assumes that S_h is an affine finite element space, which means that each mesh element $T_i \in \mathcal{T}_h$ is associated with an affine mapping $F_i : T_{\text{ref}} \rightarrow T_i$ from the reference element to T_i . These mappings are defined such that the restriction of each $v \in S_h$ to T_i is given by

$$v(\mathbf{x}) = p(F_i^{-1}\mathbf{x}) \quad (\text{A.17})$$

for some $p \in \Pi_{\text{ref}}$. An example of a reference element and its associated mappings is illustrated in Fig. A.1.

A number of reference elements are defined in the FEM literature. In hPIC2, meshes comprise a single element type. In two dimensions, the elements are either triangles or parallelograms; in three dimensions, the elements are either tetrahedra or parallelepipeds.

Céa's lemma [155] essentially states that the error in the finite element solution is limited by how well the exact solution is approximated by polynomials. The Bramble-Hilbert lemma [156] then establishes an error estimate for this polynomial interpolation. In the case of second-order elliptic problems such as (1.6), these results are exploited to obtain an estimate for finite element solutions in the L_2 -norm using the Aubin-Nitsche lemma [157], [158] and in the L_∞ -norm [159].

Appendix B

Linearization of Combined Moments for Hybrid Merge

We begin by deriving (4.19). Since

$$n_c = \int_{\mathbb{R}^d} f_c(\mathbf{v}) d\mathbf{v} \quad (\text{B.1})$$

and $f_c = f_f + f_p$, (4.19a) follows directly. Similarly,

$$\mathbf{u}_c = \frac{\int_{\mathbb{R}^d} \mathbf{v} f_c(\mathbf{v}) d\mathbf{v}}{n_c} = \frac{n_f \mathbf{u}_f + n_p \mathbf{u}_p}{n_f + n_p} \quad (\text{B.2})$$

gives (4.19b). For (4.19c), we first compute

$$mn_c E_c = \int_{\mathbb{R}^d} \frac{1}{2} m v^2 f_c(\mathbf{v}) d\mathbf{v} = mn_f E_f + mn_p E_p \quad (\text{B.3})$$

and

$$n_c u_c^2 = \frac{1}{n_c} \left(\int_{\mathbb{R}^3} \mathbf{v} f_c(\mathbf{v}) d\mathbf{v} \right)^2 = \frac{1}{n_f + n_p} (n_f^2 u_f^2 + n_p^2 u_p^2 + 2n_f n_p \mathbf{u}_f \cdot \mathbf{u}_p). \quad (\text{B.4})$$

Hence, (1.21) yields, with some algebraic manipulation,

$$p_c = p_f + p_p + m \frac{\gamma - 1}{2} \frac{n_f n_p}{n_f + n_p} (\mathbf{u}_f - \mathbf{u}_p)^2. \quad (\text{B.5})$$

Then (4.19c) follows with $p = nkT$.

Now, substituting (4.9) and letting $y_j = x_j/n_f V$, (4.19) becomes

$$n_c = n_f \left(1 + \sum_{j=1}^N y_j \right), \quad (\text{B.6a})$$

$$\mathbf{u}_c = \frac{\mathbf{u}_f + \sum_{j=1}^N y_j \mathbf{v}_j}{1 + \sum_{j=1}^N y_j}, \quad (\text{B.6b})$$

$$\frac{kT_c}{m} = \frac{\frac{kT_f}{m} + \frac{\gamma-1}{2} \left(\sum_{j=1}^N y_j v_j^2 - \frac{\sum_{i,j=1}^N y_i y_j \mathbf{v}_i \cdot \mathbf{v}_j}{\sum_{j=1}^N y_j} \right)}{1 + \sum_{j=1}^N y_j} + \frac{\gamma-1}{2} \frac{\sum_{j=1}^N y_j}{\left(1 + \sum_{j=1}^N y_j \right)^2} \left(\mathbf{u}_f - \frac{\sum_{j=1}^N y_j \mathbf{v}_j}{\sum_{j=1}^N y_j} \right)^2. \quad (\text{B.6c})$$

It will prove useful to compute

$$\frac{\partial \mathbf{u}_c}{\partial y_j} = \frac{\mathbf{v}_j \left(1 + \sum_{i=1}^N y_i \right) - \left(\mathbf{u}_f + \sum_{i=1}^N y_i \mathbf{v}_i \right)}{\left(1 + \sum_{i=1}^N y_i \right)^2}, \quad (\text{B.7})$$

which yields

$$\frac{\partial \mathbf{u}_c}{\partial y_j} \Big|_{\mathbf{y}=\mathbf{0}} = \mathbf{v}_j - \mathbf{u}_f. \quad (\text{B.8})$$

References

- [1] I. Langmuir, “Oscillations in ionized gases,” *Proceedings of the National Academy of Sciences*, vol. 14, no. 8, pp. 627–637, 1928.
- [2] M. S. Longair, *Galaxy Formation*. Springer, 2008.
- [3] M.-B. Kallenrode, *Space physics: an introduction to plasmas and particles in the heliosphere and magnetospheres*. Springer Science & Business Media, 2013.
- [4] S. A. Norberg, W. Tian, E. Johnsen, and M. J. Kushner, “Atmospheric pressure plasma jets interacting with liquid covered tissue: Touching and not-touching the liquid,” *Journal of Physics D: Applied Physics*, vol. 47, no. 47, p. 475 203, 2014.
- [5] S. Kotula, M. Lüdemann, J. Philipp, M. Thomas, and C.-P. Klages, “Plasma nitrogenation of polymer surfaces with a new type of combinatorial plasma-printing reactor,” *Plasma Processes and Polymers*, vol. 14, no. 8, p. 1 600 137, 2017.
- [6] P. K. Chu and X. Lu, *Low temperature plasma technology: methods and applications*. CRC press, 2013.
- [7] M. A. Lieberman and A. J. Lichtenberg, “Principles of plasma discharges and materials processing,” *MRS Bulletin*, vol. 30, no. 12, pp. 899–901, 1994.
- [8] J. M. Dawson, “Particle simulation of plasmas,” *Reviews of modern physics*, vol. 55, no. 2, p. 403, 1983.
- [9] H. Bluhm, *Power systems*. Springer, 2006.
- [10] J. P. Freidberg, *Plasma physics and fusion energy*. Cambridge university press, 2008.
- [11] R. Spielman, W. Stygar, J. Seamen, *et al.*, “Pulsed power performance of pbfa z,” in *Digest of Technical Papers. 11th IEEE International Pulsed Power Conference (Cat. No. 97CH36127)*, IEEE, vol. 1, 1997, pp. 709–714.
- [12] J. J. Ramirez, K. Prestwich, D. Johnson, *et al.*, “Performance of the hermes-iii gamma ray simulator,” in *7th Pulsed Power Conference*, IEEE, 1989, pp. 26–31.
- [13] F. F. Chen, *Introduction to Plasma Physics*. Springer Science & Business Media, 2012.
- [14] R. Fitzpatrick, *Plasma Physics: An Introduction*. Crc Press, 2014.
- [15] N. M. Laurendeau, *Statistical thermodynamics: fundamentals and applications*. Cambridge University Press, 2005.
- [16] G. Hagelaar and L. C. Pitchford, “Solving the boltzmann equation to obtain electron transport coefficients and rate coefficients for fluid models,” *Plasma sources science and technology*, vol. 14, no. 4, p. 722, 2005.

- [17] J. Juno, A. Hakim, J. TenBarge, E. Shi, and W. Dorland, “Discontinuous galerkin algorithms for fully kinetic plasmas,” *Journal of Computational Physics*, vol. 353, pp. 110–147, 2018.
- [18] E. L. Shi, G. Hammett, T. Stoltzfus-Dueck, and A. Hakim, “Gyrokinetic continuum simulation of turbulence in a straight open-field-line plasma,” *Journal of Plasma Physics*, vol. 83, no. 3, p. 905 830 304, 2017.
- [19] C. K. Birdsall and A. B. Langdon, *Plasma Physics via Computer Simulation*. CRC press, 2018.
- [20] R. W. Hockney and J. W. Eastwood, *Computer simulation using particles*. crc Press, 2021.
- [21] D. Tskhakaya, K. Matyash, R. Schneider, and F. Taccogna, “The particle-in-cell method,” *Contributions to Plasma Physics*, vol. 47, no. 8-9, pp. 563–594, 2007.
- [22] B. Wang, S. Ethier, W. Tang, *et al.*, “Kinetic turbulence simulations at extreme scale on leadership-class systems,” in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2013, pp. 1–12.
- [23] C.-S. Chang, S. Ku, and H. Weitzner, “Numerical study of neoclassical plasma pedestal in a tokamak geometry,” *Physics of Plasmas*, vol. 11, no. 5, pp. 2649–2667, 2004.
- [24] P. Bertrand, N. Van Tuan, M. Gros, B. Izrar, M. Feix, and J. Gutierrez, “Classical vlasov plasma description through quantum numerical methods,” *Journal of Plasma Physics*, vol. 23, no. 3, pp. 401–422, 1980.
- [25] V. Nguyen, B. Izrar, P. Bertrand, and G. Baumann, “Quantum plasma description and landau damping,” *Physics Letters A*, vol. 83, no. 9, pp. 431–434, 1981.
- [26] A. Eberhardt, A. Banerjee, M. Kopp, and T. Abel, “Investigating the use of field solvers for simulating classical systems,” *Physical Review D*, vol. 101, no. 4, p. 043 011, 2020.
- [27] R. J. LeVeque and R. J. Leveque, *Numerical methods for conservation laws*. Springer, 1992, vol. 214.
- [28] R. J. LeVeque, *Finite volume methods for hyperbolic problems*. Cambridge university press, 2002, vol. 31.
- [29] S. Patankar, *Numerical heat transfer and fluid flow*. CRC press, 2018.
- [30] R. I. Issa, “Solution of the implicitly discretised fluid flow equations by operator-splitting,” *Journal of computational physics*, vol. 62, no. 1, pp. 40–65, 1986.
- [31] B. Cockburn and C.-W. Shu, “The local discontinuous galerkin method for time-dependent convection-diffusion systems,” *SIAM journal on numerical analysis*, vol. 35, no. 6, pp. 2440–2463, 1998.
- [32] B. Cockburn and C.-W. Shu, “Runge–kutta discontinuous galerkin methods for convection-dominated problems,” *Journal of scientific computing*, vol. 16, pp. 173–261, 2001.
- [33] R. A. Gingold and J. J. Monaghan, “Smoothed particle hydrodynamics: Theory and application to non-spherical stars,” *Monthly notices of the royal astronomical society*, vol. 181, no. 3, pp. 375–389, 1977.
- [34] D. S. Harned, “Quasineutral hybrid simulation of macroscopic plasma phenomena,” *Journal of Computational Physics*, vol. 47, no. 3, pp. 452–462, 1982.
- [35] D. W. Swift, “Use of a hybrid code for global-scale plasma simulation,” *Journal of Computational Physics*, vol. 126, no. 1, pp. 109–121, 1996.

- [36] M. Sherlock, “A Monte-Carlo method for Coulomb collisions in hybrid plasma models,” *Journal of Computational Physics*, vol. 227, no. 4, pp. 2286–2292, 2008.
- [37] M. J. Kushner, “Hybrid modelling of low temperature plasmas for fundamental investigations and equipment design,” *Journal of Physics D: Applied Physics*, vol. 42, no. 19, p. 194 013, 2009.
- [38] M. T. Bettencourt, D. A. Brown, K. L. Cartwright, *et al.*, “EMPIRE-PIC: A performance portable unstructured particle-in-cell code,” *Communications in Computational Physics*, vol. 30, no. SAND-2021-2806J, 2021.
- [39] S. Brunner, E. Valeo, and J. A. Krommes, “Collisional delta-f scheme with evolving background for transport time scale simulations,” *Physics of Plasmas*, vol. 6, no. 12, pp. 4504–4521, 1999.
- [40] S. Brunner, E. Valeo, and J. Krommes, “Linear delta-f simulations of nonlocal electron heat transport,” *Physics of plasmas*, vol. 7, no. 7, pp. 2810–2823, 2000.
- [41] L. T. Meredith, M. Rezazadeh, M. Huq, *et al.*, “Hpic2: A hardware-accelerated, hybrid particle-in-cell code for dynamic plasma-material interactions,” *Computer Physics Communications*, vol. 283, p. 108 569, 2023.
- [42] C. R. Trott, D. Lebrun-Grandié, D. Arndt, *et al.*, “Kokkos 3: Programming model extensions for the exascale era,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 805–817, 2022. DOI: [10.1109/TPDS.2021.3097283](https://doi.org/10.1109/TPDS.2021.3097283).
- [43] B. M. Medina, P. Grua, K. L. Cartwright, *et al.*, “Verification and benchmarking relativistic electron beam transport through a background gas,” *Computer Physics Communications*, vol. 288, p. 108 721, 2023.
- [44] G. J. Hagelaar, “How to normalize Maxwell–Boltzmann electrons in transient plasma models,” *Journal of Computational Physics*, vol. 227, no. 2, pp. 871–876, 2007.
- [45] M. Elias and D. Curreli, “An explicit scheme to enforce charge conservation in transient particle-in-cell simulations with Maxwell-Boltzmann electrons,” *Journal of Computational Physics*, vol. 409, p. 109 320, 2020.
- [46] T. Tajima, *Computational plasma physics: with applications to fusion and astrophysics*. CRC press, 2018.
- [47] J. P. Boris *et al.*, “Relativistic plasma simulation-optimization of a hybrid code,” in *Proc. Fourth Conf. Num. Sim. Plasmas*, 1970, pp. 3–67.
- [48] H. Qin, S. Zhang, J. Xiao, J. Liu, Y. Sun, and W. M. Tang, “Why is Boris algorithm so good?” *Physics of Plasmas*, vol. 20, no. 8, p. 084 503, 2013.
- [49] M. T. Robinson, “The binary collision approximation: Background and introduction,” *Radiation Effects and Defects in Solids*, vol. 130, no. 1, pp. 3–20, 1994. DOI: [10.1080/10420159408219767](https://doi.org/10.1080/10420159408219767). eprint: <https://doi.org/10.1080/10420159408219767>. [Online]. Available: <https://doi.org/10.1080/10420159408219767>.
- [50] G. Kawamura, Y. Tomita, and A. Kirschner, “Pic simulation of kinetic effects of plasma and consequences for physical sputtering,” *Journal of nuclear materials*, vol. 415, no. 1, S192–S195, 2011.
- [51] V. Vahedi, G. DiPeso, C. Birdsall, M. Lieberman, and T. Rognlien, “Capacitive RF discharges modelled by particle-in-cell Monte Carlo simulation. I. analysis of numerical techniques,” *Plasma Sources Science and Technology*, vol. 2, no. 4, p. 261, 1993.

- [52] G. Bird, *Molecular gas dynamics and the direct simulation of gas flows*. Clarendon Press, 1994.
- [53] S. Rjasanow and W. Wagner, “A stochastic weighted particle method for the boltzmann equation,” *Journal of Computational Physics*, vol. 124, no. 2, pp. 243–253, 1996.
- [54] S. Araki and R. Martin, “Interspecies fractional collisions,” *Physics of Plasmas*, vol. 27, no. 3, 2020.
- [55] R. Martin, “Conservative bin-to-bin fractional collisions,” in *AIP Conference Proceedings*, AIP Publishing, vol. 1786, 2016.
- [56] G. Cohen and S. Pernet, *Finite Element and Discontinuous Galerkin Methods for Transient Wave Equations*. Springer, 2017.
- [57] R. Hartmann and P. Houston, “Adaptive discontinuous galerkin finite element methods for the compressible euler equations,” *Journal of Computational Physics*, vol. 183, no. 2, pp. 508–532, 2002.
- [58] E. F. Toro, *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.
- [59] C. Runge, “Über die numerische auflösung von differentialgleichungen,” *Mathematische Annalen*, vol. 46, no. 2, pp. 167–178, 1895.
- [60] W. Kutta, *Beitrag zur näherungsweise Integration totaler Differentialgleichungen*. Teubner, 1901.
- [61] J. C. Butcher, *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.
- [62] R. Courant, K. Friedrichs, and H. Lewy, “Über die partiellen differenzengleichungen der mathematischen physik,” *Mathematische annalen*, vol. 100, no. 1, pp. 32–74, 1928.
- [63] S. K. Godunov and I. Bohachevsky, “Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics,” *Matematicheskij sbornik*, vol. 47, no. 3, pp. 271–306, 1959.
- [64] J. VonNeumann and R. D. Richtmyer, “A method for the numerical calculation of hydrodynamic shocks,” *Journal of applied physics*, vol. 21, no. 3, pp. 232–237, 1950.
- [65] X.-D. Liu, S. Osher, and T. Chan, “Weighted essentially non-oscillatory schemes,” *Journal of computational physics*, vol. 115, no. 1, pp. 200–212, 1994.
- [66] S. A. Moe, J. A. Rossmanith, and D. C. Seal, “A simple and effective high-order shock-capturing limiter for discontinuous galerkin methods,” *arXiv preprint arXiv:1507.03024*, 2015.
- [67] A. Harten, P. D. Lax, and B. v. Leer, “On upstream differencing and godunov-type schemes for hyperbolic conservation laws,” *SIAM review*, vol. 25, no. 1, pp. 35–61, 1983.
- [68] S. Davis, “Simplified second-order godunov-type methods,” *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 3, pp. 445–473, 1988.
- [69] V. Rusanov, “Calculation of interaction of non-steady shock waves with obstacles,” *J. Comput. Math. Phys. USSR*, vol. 1, pp. 267–279, 1961.
- [70] R. J. LeVeque, “Time-split methods for partial differential equations,” Ph.D. dissertation, Stanford University, Stanford, CA, Apr. 1982.
- [71] G. Strang, “On the construction and comparison of difference schemes,” *SIAM journal on numerical analysis*, vol. 5, no. 3, pp. 506–517, 1968.
- [72] S. T. Miller and U. Shumlak, “A multi-species 13-moment model for moderately collisional plasmas,” *Physics of Plasmas*, vol. 23, no. 8, 2016.

- [73] H. Grad, “On the kinetic theory of rarefied gases,” *Communications on pure and applied mathematics*, vol. 2, no. 4, pp. 331–407, 1949.
- [74] H. Grad, “Principles of the kinetic theory of gases,” in *Thermodynamik der Gase/Thermodynamics of Gases*, Springer, 1958, pp. 205–294.
- [75] R. Reyes and V. Lomüller, “SYCL: Single-source C++ accelerator programming,” in *Parallel Computing: On the Road to Exascale*, IOS Press, 2016, pp. 673–682.
- [76] L. Sommer, J. Korinth, and A. Koch, “OpenMP device offloading to FPGA accelerators,” in *2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, IEEE, 2017, pp. 201–205.
- [77] D. A. Beckingsale, J. Burmark, R. Hornung, *et al.*, “Raja: Portable performance for large-scale scientific applications,” in *2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, IEEE, 2019, pp. 71–81.
- [78] B. Wang, S. Ethier, W. Tang, *et al.*, “Kinetic turbulence simulations at extreme scale on leadership-class systems,” in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC ’13, Denver, Colorado: Association for Computing Machinery, 2013, pp. 1–12, ISBN: 9781450323789. DOI: [10.1145/2503210.2503258](https://doi.org/10.1145/2503210.2503258). [Online]. Available: <https://doi.org/10.1145/2503210.2503258>.
- [79] R. Bird, P. Gillies, M. Bareford, J. Herdman, and S. Jarvis, “Mini-app driven optimisation of inertial confinement fusion codes,” in *2015 IEEE International Conference on Cluster Computing*, 2015, pp. 768–776. DOI: [10.1109/CLUSTER.2015.132](https://doi.org/10.1109/CLUSTER.2015.132).
- [80] T. R. Younkin, D. L. Green, A. B. Simpson, and B. Wirth, “GITR: An accelerated global scale particle tracking code for wall material erosion and redistribution in fusion relevant plasma–material interactions,” *Computer Physics Communications*, vol. 264, p. 107885, 2021.
- [81] C. W. Smith, G. Diamond, G. Perumpilly, *et al.*, “Unstructured mesh technologies for fusion simulations,” in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC ’19, Denver, Colorado: Association for Computing Machinery, 2019, pp. 1–3.
- [82] M. Bussmann, H. Burau, T. E. Cowan, *et al.*, “Radiative signatures of the relativistic Kelvin-Helmholtz instability,” in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC ’13, Denver, Colorado: ACM, 2013, 5:1–5:12, ISBN: 978-1-4503-2378-9. DOI: [10.1145/2503210.2504564](https://doi.org/10.1145/2503210.2504564). [Online]. Available: <http://doi.acm.org/10.1145/2503210.2504564>.
- [83] R. Anderson, J. Andrej, A. Barker, *et al.*, “MFEM: A modular finite element methods library,” *Computers & Mathematics with Applications*, vol. 81, pp. 42–74, 2021.
- [84] J. T. Drobny and D. Curreli, “RustBCA: A high-performance binary-collision-approximation code for ion-material interactions,” *Journal of Open Source Software*, vol. 6, no. 64, p. 3298, 2021. DOI: [10.21105/joss.03298](https://doi.org/10.21105/joss.03298). [Online]. Available: <https://doi.org/10.21105/joss.03298>.
- [85] Y. Yamamura, N. Matsunami, and N. Itoh, “A new empirical formula for the sputtering yield,” *Radiation Effects*, vol. 68, no. 3, pp. 83–87, 1982.
- [86] Y. Yamamura, “Theory of sputtering and comparison to experimental data,” *Nuclear Instruments and Methods in Physics Research*, vol. 194, no. 1-3, pp. 515–522, 1982.

- [87] W. Eckstein, C. Garcia-Rosales, J. Roth, and W. Ottenberger, “Sputtering data,” 1993.
- [88] J. F. Ziegler, M. D. Ziegler, and J. P. Biersack, “SRIM—the stopping and range of ions in matter (2010),” *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 268, no. 11-12, pp. 1818–1823, 2010.
- [89] A. Mutzke, R. Schneider, W. Eckstein, *et al.*, “SDTrimSP version 6.00,” 2019.
- [90] W. Möller and W. Eckstein, “Tridyn—a TRIM simulation code including dynamic composition changes,” *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 2, no. 1-3, pp. 814–818, 1984.
- [91] J. Drobny, A. Hayes, D. Curreli, and D. N. Ruzic, “F-tridyn: A binary collision approximation code for simulating ion interactions with rough surfaces,” *Journal of Nuclear Materials*, vol. 494, pp. 278–283, 2017, ISSN: 0022-3115. DOI: <https://doi.org/10.1016/j.jnucmat.2017.07.037>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S002231151730301X>.
- [92] D. Reiter, M. Baelmans, and P. Boerner, “The EIRENE and B2-EIRENE codes,” *Fusion science and technology*, vol. 47, no. 2, pp. 172–186, 2005.
- [93] A. Kirschner, V. Philipps, J. Winter, and U. Kögler, “Simulation of the plasma-wall interaction in a tokamak with the Monte Carlo code ERO-TEXTOR,” *Nuclear Fusion*, vol. 40, no. 5, p. 989, 2000.
- [94] A. Kirschner, D. Tskhakaya, G. Kawamura, *et al.*, “Modelling of impurity transport and plasma-wall interaction in fusion devices with the ERO code: Basics of the code and examples of application,” *Contributions to Plasma Physics*, vol. 56, no. 6-8, pp. 622–627, 2016.
- [95] R. D. Falgout and U. M. Yang, “hypre: A library of high performance preconditioners,” in *International Conference on Computational Science*, Springer, 2002, pp. 632–641.
- [96] H. A. Van der Vorst, “Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems,” *SIAM Journal on scientific and Statistical Computing*, vol. 13, no. 2, pp. 631–644, 1992.
- [97] U. M. Yang *et al.*, “BoomerAMG: A parallel algebraic multigrid solver and preconditioner,” *Applied Numerical Mathematics*, vol. 41, no. 1, pp. 155–177, 2002.
- [98] T. Preston-Werner and P. Gedam, *TOML Specification 1.0.0*, version v1.0.0, Jan. 2021. [Online]. Available: <https://toml.io/en/v1.0.0>.
- [99] T. Niina, J. Williams, G. Fraux, *et al.*, *ToruNiina/toml11: version 3.7.1*, version v3.7.1, Mar. 2022. DOI: [10.5281/zenodo.6349353](https://doi.org/10.5281/zenodo.6349353). [Online]. Available: <https://doi.org/10.5281/zenodo.6349353>.
- [100] W. Eckstein, “Charge fractions of reflected particles,” in *Inelastic Particle-Surface Collisions*, E. Taglauer and W. Heiland, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1981, pp. 157–183, ISBN: 978-3-642-87065-1.
- [101] F. Bronold, K. Matyash, D. Tskhakaya, R. Schneider, and H. Fehske, “Radio-frequency discharges in oxygen: I. particle-based modelling,” *Journal of Physics D: Applied Physics*, vol. 40, no. 21, p. 6583, 2007.
- [102] D. Tskhakaya, “One-dimensional plasma sheath model in front of the divertor plates,” *Plasma Physics and Controlled Fusion*, vol. 59, no. 11, p. 114001, 2017.
- [103] D. Tskhakaya, “On recent massively parallelized PIC simulations of the SOL,” *Contributions to Plasma Physics*, vol. 52, no. 5-6, pp. 490–499, 2012.

- [104] T. Takizuka, “Kinetic effects in edge plasma: Kinetic modeling for edge plasma and detached divertor,” *Plasma Physics and Controlled Fusion*, vol. 59, no. 3, p. 034 008, 2017.
- [105] T. Takizuka and H. Abe, “A binary collision model for plasma simulation with a particle code,” *Journal of computational physics*, vol. 25, no. 3, pp. 205–219, 1977.
- [106] V. Vahedi and M. Surendra, “A Monte Carlo collision model for the particle-in-cell method: Applications to argon and oxygen discharges,” *Computer Physics Communications*, vol. 87, no. 1-2, pp. 179–198, 1995.
- [107] C. K. Birdsall, “Particle-in-cell charged-particle simulations, plus Monte Carlo collisions with neutral atoms, PIC-MCC,” *IEEE Transactions on plasma science*, vol. 19, no. 2, pp. 65–85, 1991.
- [108] W. Bland, A. Bouteiller, T. Herault, G. Bosilca, and J. Dongarra, “Post-failure recovery of MPI communication capability: Design and rationale,” *The International Journal of High Performance Computing Applications*, vol. 27, no. 3, pp. 244–254, 2013.
- [109] The HDF Group, *Hierarchical Data Format, version 5*. [Online]. Available: <https://github.com/HDFGroup/hdf5>.
- [110] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit*. Kitware, 2006.
- [111] J. Ahrens, B. Geveci, C. Law, C. Hansen, and C. Johnson, “36-paraview: An end-user tool for large-data visualization,” *The visualization handbook*, vol. 717, pp. 50 038–1, 2005.
- [112] U. Ayachit, *The paraview guide: a parallel visualization application*. Kitware, Inc., 2015.
- [113] W. L. Oberkampf and C. J. Roy, *Verification and validation in scientific computing*. Cambridge university press, 2010.
- [114] M. M. Turner, “Verification of particle-in-cell simulations with monte carlo collisions,” *Plasma Sources Science and Technology*, vol. 25, no. 5, p. 054 007, 2016.
- [115] M. M. Turner, A. Derzsi, Z. Donko, *et al.*, “Simulation benchmarks for low-pressure plasmas: Capacitive discharges,” *Physics of Plasmas*, vol. 20, no. 1, 2013.
- [116] A. V. Bobylev, “Exact solutions of the Boltzmann equation,” in *Akademiia Nauk SSSR Doklady*, vol. 225, 1975, pp. 1296–1299.
- [117] M. Krook and T. T. Wu, “Formation of Maxwellian tails,” *Physical Review Letters*, vol. 36, no. 19, p. 1107, 1976.
- [118] G. A. Sod, “A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws,” *Journal of computational physics*, vol. 27, no. 1, pp. 1–31, 1978.
- [119] M. Nazarov and A. Larcher, “Numerical investigation of a viscous regularization of the euler equations by entropy viscosity,” *Computer Methods in Applied Mechanics and Engineering*, vol. 317, pp. 128–152, 2017.
- [120] R. A. DeVore, “Nonlinear approximation,” *Acta numerica*, vol. 7, pp. 51–150, 1998.
- [121] O. Buneman, “Dissipation of currents in ionized media,” *Physical Review*, vol. 115, no. 3, p. 503, 1959.
- [122] J. Dawson, “One-dimensional plasma model,” *The Physics of Fluids*, vol. 5, no. 4, pp. 445–459, 1962.
- [123] J. M. Dawson, “Particle simulation of plasmas,” *Reviews of Modern Physics*, vol. 55, no. 2, pp. 403–447, Apr. 1983. DOI: [10.1103/RevModPhys.55.403](https://doi.org/10.1103/RevModPhys.55.403).

- [124] F. Assous, T. P. Dulimbert, and J. Segré, “A new method for coalescing particles in PIC codes,” *Journal of Computational Physics*, vol. 187, no. 2, pp. 550–571, 2003.
- [125] G. Lapenta, “Particle rezoning for multidimensional kinetic particle-in-cell simulations,” *Journal of computational physics*, vol. 181, no. 1, pp. 317–337, 2002.
- [126] D. R. Welch, T. C. Genoni, R. E. Clark, and D. V. Rose, “Adaptive particle management in a particle-in-cell code,” *Journal of Computational Physics*, vol. 227, no. 1, pp. 143–155, 2007.
- [127] M. Pfeiffer, A. Mirza, C.-D. Munz, and S. Fasoulas, “Two statistical particle split and merge methods for particle-in-cell codes,” *Computer Physics Communications*, vol. 191, pp. 9–24, 2015.
- [128] M. Vranic, T. Grismayer, J. L. Martins, R. A. Fonseca, and L. O. Silva, “Particle merging algorithm for PIC codes,” *Computer Physics Communications*, vol. 191, pp. 65–73, 2015.
- [129] T. Hall, J. Boerner, Z. Eckert, R. Hooper, and J. Pacheco, “Effect of dynamic particle reweighting on plasma swarm parameters in the particle-in-cell code Aleph,” in *APS Annual Gaseous Electronics Meeting Abstracts*, 2021, CT14–005.
- [130] R. S. Martin and J.-L. Cambier, “Octree particle management for DSMC and PIC simulations,” *Journal of Computational Physics*, vol. 327, pp. 943–966, 2016.
- [131] J. Byers, B. Cohen, W. Condit, and J. Hanson, “Hybrid simulations of quasineutral phenomena in magnetized plasma,” *Journal of Computational Physics*, vol. 27, no. 3, pp. 363–396, 1978.
- [132] D. Hewett, “A global method of solving the electron-field equations in a zero-inertia-electron-hybrid plasma simulation code,” *Journal of Computational Physics*, vol. 38, no. 3, pp. 378–395, 1980.
- [133] J. Davies, A. Bell, M. Haines, and S. Guerin, “Short-pulse high-intensity laser-generated fast electron transport into thick solid targets,” *Physical Review E*, vol. 56, no. 6, p. 7193, 1997.
- [134] R. Mason, “Heating mechanisms in short-pulse laser-driven cone targets,” *Physical review letters*, vol. 96, no. 3, p. 035 001, 2006.
- [135] L. Gargaté, R. Bingham, R. A. Fonseca, and L. O. Silva, “dHybrid: A massively parallel code for hybrid simulations of space plasmas,” *Computer physics communications*, vol. 176, no. 6, pp. 419–425, 2007.
- [136] K. L. Cartwright, J. P. Verboncoeur, and C. K. Birdsall, “Nonlinear hybrid Boltzmann–particle-in-cell acceleration algorithm,” *Physics of Plasmas*, vol. 7, no. 8, pp. 3252–3264, 2000.
- [137] R. J. Mason, “The implicit hybrid/PIC code ANTHEM,” Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2002.
- [138] O. Sen, N. J. Gaul, K. Choi, G. Jacobs, and H. Udaykumar, “Evaluation of kriging based surrogate models constructed from mesoscale computations of shock interaction with particles,” *Journal of Computational Physics*, vol. 336, pp. 235–260, 2017, ISSN: 0021-9991. DOI: <https://doi.org/10.1016/j.jcp.2017.01.046>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021999117300621>.
- [139] B. Cohen, A. Kemp, and L. Divol, “Simulation of laser–plasma interactions and fast-electron transport in inhomogeneous plasma,” *Journal of Computational Physics*, vol. 229, no. 12, pp. 4591–4612, 2010, ISSN: 0021-9991. DOI: <https://doi.org/10.1016/j.jcp.2010.03.001>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S002199911000104X>.

- [140] D. R. Welch, N. Bennett, T. C. Genoni, C. Thoma, and D. V. Rose, “Fast hybrid particle-in-cell technique for pulsed-power accelerators,” *Phys. Rev. Accel. Beams*, vol. 23, p. 110401, 11 Nov. 2020. DOI: [10.1103/PhysRevAccelBeams.23.110401](https://doi.org/10.1103/PhysRevAccelBeams.23.110401). [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevAccelBeams.23.110401>.
- [141] S. Braginskii, “Transport processes in a plasma,” *Reviews of plasma physics*, vol. 1, 1965.
- [142] S. Chapman and T. G. Cowling, *The mathematical theory of non-uniform gases: an account of the kinetic theory of viscosity, thermal conduction and diffusion in gases*. Cambridge university press, 1990.
- [143] P. L. Bhatnagar, E. P. Gross, and M. Krook, “A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems,” *Physical review*, vol. 94, no. 3, p. 511, 1954.
- [144] P. B. Stark and R. L. Parker, “Bounded-variable least-squares: An algorithm and applications,” *Computational Statistics*, vol. 10, pp. 129–129, 1995.
- [145] C. Trott, L. Berger-Vergiat, D. Poliakoff, *et al.*, “The Kokkos EcoSystem: Comprehensive Performance Portability for High Performance Computing,” *Computing in Science Engineering*, vol. 23, no. 5, pp. 10–18, 2021. DOI: [10.1109/MCSE.2021.3098509](https://doi.org/10.1109/MCSE.2021.3098509).
- [146] H. C. Edwards, C. R. Trott, and D. Sunderland, “Kokkos: Enabling manycore performance portability through polymorphic memory access patterns,” *Journal of Parallel and Distributed Computing*, vol. 74, no. 12, pp. 3202–3216, 2014, Domain-Specific Languages and High-Level Frameworks for High-Performance Computing, ISSN: 0743-7315. DOI: <https://doi.org/10.1016/j.jpdc.2014.07.003>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0743731514001257>.
- [147] F. Bassi and S. Rebay, “High-order accurate discontinuous finite element solution of the 2D Euler equations,” *Journal of computational physics*, vol. 138, no. 2, pp. 251–285, 1997.
- [148] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, “The development of discontinuous Galerkin methods,” in *Discontinuous Galerkin Methods*, Springer, 2000, pp. 3–50.
- [149] C. L. Lawson and R. J. Hanson, *Solving least squares problems*. SIAM, 1995.
- [150] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 2018.
- [151] G. A. Radtke, N. Martin, C. H. Moore, A. Huang, and K. L. Cartwright, “Robust verification of stochastic simulation codes,” *Journal of Computational Physics*, vol. 451, p. 110855, 2022.
- [152] E. Esarey, C. B. Schroeder, and W. P. Leemans, “Physics of laser-driven plasma-based electron accelerators,” *Reviews of modern physics*, vol. 81, no. 3, p. 1229, 2009.
- [153] A. L. Peratt, “Advances in numerical modeling of astrophysical and space plasmas,” *Astrophysics and Space Science*, vol. 242, pp. 93–163, 1996.
- [154] D. Braess, *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*. Cambridge University Press, 2007.
- [155] J. Céa, “Approximation variationnelle des problèmes aux limites,” *Annales de l’institut Fourier*, vol. 14, no. 2, pp. 345–444, 1964.
- [156] J. H. Bramble and S. Hilbert, “Estimation of linear functionals on Sobolev spaces with application to Fourier transforms and spline interpolation,” *SIAM Journal on Numerical Analysis*, vol. 7, no. 1, pp. 112–124, 1970.

- [157] J. P. Aubin, “Behavior of the error of the approximate solutions of boundary value problems for linear elliptic operators by Galerkin’s and finite difference methods,” *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze*, vol. 21, no. 4, pp. 599–637, 1967.
- [158] J. Nitsche, “Ein kriterium für die quasi-optimalität des ritzschen verfahrens,” *Numerische Mathematik*, vol. 11, no. 4, pp. 346–348, 1968.
- [159] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*. SIAM, 2002.