INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

OPTIMIZED 1D-1V VLASOV-POISSON SIMULATIONS USING FOURIER-HERMITE SPECTRAL DISCRETIZATIONS

by

Joseph Wade Schumer

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Nuclear Engineering)
in The University of Michigan
1997

Doctoral Committee:

Associate Professor James P. Holloway, Chairperson
Professor John P. Boyd
Professor Edward Larsen
Professor Y. Y. Lau

Dedicated to my parents, Ronald and Gloria Schumer, without whose love, support, and prayers this would not have been possible.

ACKNOWLEDGEMENTS

I wish to thank Professor James Holloway for his insight and guidance during my years at the University of Michigan. His wit, patience, and "down-to-earth" attitude provided the perfect atmosphere in which to enjoy the beauty of the physics.

Family members, whose love, patience, and companionship helped me through the past 27 years, include my parents to whom this dissertation is dedicated, my grandmother Beatrice Irene Naes, my brother Matthew Boyd Schumer (who is nearly an M.D....I just beat you, bro!), my grandparents who passed many years ago (Buford and Ruby Schumer, Leo James Naes) who could not see my accomplishments but from above, my uncle Jim and cousin Robert Naes. I would also like to welcome and thank the most recent additions to the Schumer and Naes families, including my sister-in-law Kristi (Marshall) Schumer in 1992, my new nephew Jonathan Marshall Schumer in October 1996, and the entire Memphis group ala Jim Naes: Lori, Kandace, Kayla, Karlee, and Kyle.

At the University of Michigan, I would like to first thank my "advisorial grandfather" Professor Ziya Akcasu for the many conversations ranging from the Wigner "6j" formula to the nationality of goats on a rocky Mediterranean isle (Turkish or Greek?...let the feta decide). Other professors at the University of Michigan who aided my progress through stimulating lectures or informal "in the hall" conversations include Professors Y. Y. Lau, Ron Gilgenbach, Bill Martin, Terry Kammash, Hal Marshall of LASC, Ed Larsen, and John Boyd of AOSS. During my undergradu-

ate years at the University of Missouri-Rolla, I benefited greatly from the advice and teaching of Professors Gary Mueller, Tom Dolan, Nick Tsoulfanidis, Arvind Kumar, and Joel Kramme. For their guidance and great lunchtime conversations during my internships at Oak Ridge National Laboratory-Fusion Energy Division, I wish to express my gratitude to Steve Hirshman, James Rome, Stan Milora, Don Batchelor, Lew Hedrick, and John Sheffield. At the University of Texas-Austin, I would like to thank Phil Morrison, Brad Shadwick, and Vernon Wong for keeping me busy during the summer of 1993 with "bracketology" and the like.

Graduate students and comrades who have given companionship, collaboration, conciliation, and love: Dr. John "Bucky" Luginsland, Denise "Stealth" Paraventi, Ed "POA1" Moore, David "IBT" Stuenkel, Dr. Todd "Tmonster" Urbatsch, Dr. Mark "Pencilneck" Walter, Susan "Ginger" Winchester and the USPAS gang, my cat, Lucky, and her "mom", Marcia Prewitt, The Buttheads of St Louis, The Anderson Boys (Scott and Rex, my officemates), Dr. Ayman Hawari, Carla Barrett, Anthony Ricci (my weight-lifting partner), Lisa "Poophead" Manne and ORSERS everywhere. Good luck to you all in future...I hope we always stay in touch.

This work was supported by the Department of Energy Magnetic Fusion Science Fellowship and a Graduate Research Assistantship through the University of Michigan with funds from the National Science Foundation on contract NSF ECS-935834.

PREFACE

Future understanding of physical phenomena in collisionless plasmas necessitates the development of accurate and optimized simulation tools. Particle-in-cell (PIC) methods, although based on simple algorithms and well-suited for the complex geometries of "real world" applications, do not properly resolve tenuous regions of phase space. Taking the experience of previously developed kinetic codes which evolve the phase space distribution function $f(x, u, t)$ using the Vlasov-Poisson system, I combine some of the more promising features, including spectral representation of the distributions and velocity-space filtering, into one method.

In this thesis, a 1d-1v spatially periodic, thermally-warm, charged particle distribution is represented with one of two different Fourier-Hermite (FH) basis sets. Coefficients of these basis sets are the primary unknowns evolved through time using the FH-transformed and Gaussian-filtered Vlasov-Poisson equations. The process of filtering helps to prevent "filamentation," a phenomena of collisionless plasmas, from destroying the numerical accuracy of the scheme while preserving the form of the Poisson equation. In addition, an $O(\Delta t^2)$-accurate time-splitting algorithm is applied, separately modeling the advection and acceleration of particles and yielding a faster algorithm by avoiding a costly convolution sum for the acceleration term. Optimizing the spectral accuracy of a Hermite basis by properly choosing the velocity scale-length is shown to yield orders of magnitude reduction in errors. Comparisons of these two different FH schemes will be performed, comparing them

against one another, against a standard PIC scheme, and against a similar filtered Fourier-Fourier method which also uses a splitting technique. In these comparisons, conservation properties and physical accuracy in linear regimes have determined the best method to be the filtered Fourier-Hermite algorithm using symmetric Hermite normalizations.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

LIST OF APPENDICES

CHAPTER I

METHODOLOGY

1.1 Introduction

A clearer physical understanding of collisionless plasmas, the kinetic processes inherent to them, and their technological implications, such as particle and energy transport in tokamaks, laser wake-field acceleration, and power limits in microwave generators, can be obtained through simulation techniques [Nrc.1]. Analytically, even in one-dimensional (1d-1v) systems, the non-linear growth rates and saturation levels of electrostatic instabilities can only be approximated theoretically [Ber.1, Dav.1, Fri.1, Pen.1, Ska.1]. We therefore require expeditious algorithms which can preserve as many physical constants as possible, maintain numerical stability, and reliably reproduce results from analytic theory in linear regimes so we may have faith in their non-linear predictions.

Self-consistent simulations of charged-particle dynamics using particle-in-cell (PIC) or cloud-in-cell (CIC) algorithms have been developed [Daw.1, Bir.2, others] and, quite frequently, used. Indeed, these simulation methods have become standard technologies in the design and evaluation of neutral and non-neutral plasma systems. However, PIC/CIC codes, based on the modeling of plasmas using 10^2 to 10^7 macroparticles, are inherently noisy. In plasmas with tenuous velocity-space profiles,

there can be few "numerical" particles coinciding with the phase velocities of the electrostatic waves; hence, plasmas having these delicate structures call for more precise modeling.

The goal of this dissertation is to develop a numerical kinetic method for 1d-1v systems which can accurately resolve delicate phase-space distributions, efficiently evolve these distributions while maintaining numerical stability, and make reliable linear and non-linear physical predictions.

To accomplish this task, we may study the non-relativistic evolution of charged-particle distributions f_α in phase space (x, u), described by the Vlasov-Maxwell (VM) system of equations [Nic.1]. In this work, the numerical algorithm centers around a system of equations which can model the VM system in one-dimension (1d-1v), namely the *filtered Vlasov-Poisson (VP) equations* for the species α,

$$\frac{\partial f_\alpha(x,u,t)}{\partial t} + u\frac{\partial f_\alpha}{\partial x} + \frac{q_\alpha}{m_\alpha}E(x,t)\frac{\partial f_\alpha}{\partial u} = -v_{o\alpha}^2\frac{\partial^2 f_\alpha}{\partial x \partial u} \tag{1.1.1}$$

$$\frac{\partial E(x,t)}{\partial x} = \sum_\alpha \frac{q_\alpha}{\epsilon_o}\int_{-\infty}^{\infty} f_\alpha(x,u,t)\, du\,. \tag{1.1.2}$$

This system, used for the numerical study of filtered solutions of the Vlasov equation, contains interesting and subtle physics in its own right, and its origin will be revealed later in Section 1.4. For now, it is sufficient to say that these equations, solved using a splitting scheme [Che.1, others], allow a fast, flexible, and accurate algorithm.

Gaussian filtering [Kli.2], which produces the $v_{o\alpha}^2$ term on the right-hand side of Equation 1.1.1, exactly smooths the distribution $f_\alpha(x, u, t)$ in velocity space, thereby sustaining for long times the accuracy of the method at the finest velocity scales of the simulation. Because velocity filamentation occurs naturally in collisionless plasmas (see Appendix A), we are required to pay special attention to these fine scales. The Gaussian filter was chosen by Klimas because it can produce filtered

solutions of the unfiltered Vlasov equation yet does not affect the charge densities or currents; hence, the electric field dynamics are correct and the simulation is more resistant to errors at the shortest velocity scales. The filtered Vlasov-Poisson system is introduced in Section 1.4.

In Section 1.5, we employ a splitting technique which decouples the advection and acceleration phase space mappings into two separate first-order partial differential equations, providing a way to optimize each mapping with different schemes while incurring only a moderate accuracy expense. Accuracy of the splitting method is shown to be $O(\Delta t^2)$ given either a constant E-field or time-varying E-field during the acceleration phase.

In addition to splitting and filtering the Vlasov-Maxwell system of equations, this work utilizes, compares, and optimizes two spectrally-transformed VP systems in which the details of the distribution functions $f_\alpha(x,u,t)$ are carried in an associated matrix $\mathbf{f}_\alpha$ of spectral coefficients (f_α^{mn}). Spectral methods are ideally suited to physical problems in which fine-scale evolution plays a significant role, such as turbulent fluid flows or collisionless plasmas [Can.1]; they also tend to generate conservative and non-dispersive schemes. In this dissertation, the phase-space dependence of the distribution $f_\alpha(x,u,t)$ is considered to be periodic in space and, at least initially, assumed to be a sum of Maxwellians in velocity. Therefore, we choose a Fourier basis in x and one of two different Hermite bases in u (symmetric and asymmetric normalizations, see Chapter II).

The Hermite basis is a natural choice for Maxwellian-like velocity profiles because the lowest order expansion function is then a Gaussian function [Gra.2]. To optimize the spectral accuracy of the Hermite function representation in these simulations, we introduce a novel species-dependent velocity scale U_α, based on the theoretical work

of Boyd [Boy.1], which yielded orders of magnitude reduction in errors for numerical solution of the linearized Vlasov-Poisson system in previous work by Holloway [Hol.2].

Because the symmetric and asymmetric normalizations of the Hermite functions yield slightly different algorithms, analysis of these two methods will center on efficiency, scaling of errors with expansion order, fidelity of spectral expansion coefficients with time, and conservation of primary physical quantities, such as particles, momentum, and total energy in the fully discrete system. Numerical comparisons between a similar Fourier-Fourier (FF) method [Kli.3], a standard periodic PIC code ES1 [Bir.2], and the two Fourier-Hermite (FH) methods will be performed, based on their respective modeling of the Landau damping phenomena in a uniform Maxwellian plasma and their modeling of the two-stream instability in a system with a classic "bump-on-tail" profile [Den.1, Dem.1, others].

In Chapter II, the Fourier-Hermite weighted-residuals method for the filtered and split Vlasov-Poisson system is derived. Chapter III illustrates the conservation properties of both of the FH methods. Chapter IV shows the collisionless plasma simulations dealing with two different velocity profiles: Maxwellian (stable damped modes) and bump-on-tail (unstable growing modes). Comparisons to linear kinetic theory, FF results, and PIC simulations are shown. Chapter V finishes the dissertation with a discussion of all of the results, recommendations for optimal FH simulations, and suggestions for future work in this area.

1.2 Historical Background

Plasma simulation methods can be generally divided into three groups: magneto-hydrodynamic (MHD) fluid methods, kinetic methods, and hybrid methods. MHD methods [Rob.1] evolve low-order velocity moments such as particle densities, cur-

rents, and temperatures using discretizations of the continuity equation, the momentum balance equation, and the energy balance equation for charged fluids, coupled self-consistently with Maxwell's equations. Kinetic methods, which include particle-in-cell (PIC), Vlasov solvers, and Fokker-Planck solvers, are required for problems with more complicated electromagnetic-particle interactions in plasmas with evolving phase-space profiles. PIC methods [Daw.2, Mor.3, Bir.1] directly integrate the equations of motion for a large collection of charged particles moving under the force of their own self-generated fields or externally applied fields; while PIC is easy to implement, Vlasov solvers, in contrast, calculate the evolution of distribution functions, are free of artificial discrete particle noise, and are better suited for warm or tenuous plasmas. Fokker-Planck solvers [Kil.1] follow in the line of Vlasov solvers, but are intended for collisional plasmas dominated by forward-peaked scattering. Hybrid methods, which are now under development, meld promising features of the above methods into algorithms which model a particular set of physical mechanisms; for example, electrons could be simulated using a kinetic method while the sluggish ions could be modeled accurately with a MHD scheme [Nun.1].

Other than spectral Vlasov solvers, which will be outlined further below, one may choose to use standard finite difference schemes [Bye.1], finite elements schemes [Gar.1], or methods which integrate the distribution along "characteristic orbits" [Ber.2, Che.1]. All three inherently require either a low-order interpolation to calculate derivatives or to map the distribution back onto a fixed grid. This numerical smoothing does reduce filamentation, but at the cost of physical accuracy (these tend to be dissipative methods). Conservation of particles, momentum, and energy is, in general, only approximate.

Spectral methods for solving the Vlasov-Poisson system, including Fourier, Her-

mite, Hilbert, and Chebyshev discretizations, in one or both dimensions of phase space, have been implemented in various forms [Kli.2, Arm.1, Ghi.2, Sho.1, others]. A Fourier-Fourier (FF) discretization in (x,u) phase space, with an $O(\Delta t^2)$ splitting technique to decouple the advection and acceleration terms during the time-integration, has been used by Klimas and Farrell [Kli.2, Kli.3] and Ghizzo et al [Ghi.1]. Klimas and Farrell also introduced a Gaussian velocity-space filter to combat filamentation; however, the filtering term can generate a numerical instability for short time-steps (see Appendix B) and Fourier expansions in velocity cannot conserve particle momentum. Fourier-Hermite (FH) discretizations [Arm.1, Sho.2] in (x,u) using asymmetric Hermite normalizations have also been implemented, but without splitting, filtering, or velocity-scaling of the Hermite functions (no symmetric Hermite scheme for plasma kinetic simulations has been developed). Hence, costly convolution sums over the $E\partial_u f$ term were performed, poorly-resolved fine-scales developed at the level of the velocity grid, and the non-optimal spectral expansions required between 500 to 1000 Hermite modes to achieve moderate accuracy levels. Some of these FH algorithms incorporated artificial damping or monotonically decreasing Hermite expansion order $N_u(t)$ to combat the errors at the fine scales [Har.1, Joy.1, Gra.1, Kno.2, others]; however, artificial damping changes the interesting collisionless physics and decreasing N_u eventually leaves the simulation with no velocity resolution whatsoever. Hermite methods were originally dismissed because of their poor resolution properties, but recent work with scaled Hermites has shown with proper selection of the scale length, they can be quite competitive when modeling functions with Gaussian-shaped profiles [Tan.1].

1.3 The Next Step

A Gaussian filter and a splitting scheme have been applied to the Vlasov-Poisson system before; however, only Klimas and Farrell in their Fourier-Fourier scheme have used both methods. The major contributions of this dissertation are continuation of these ideas with two different Hermite spectral velocity discretizations and improvement upon other Hermite-based schemes by scaling and filtering velocity-space to enhance spectral accuracy.

1.4 Filtered Vlasov-Maxwell Equations

In three-dimensions (3d-3v), the collisionless Boltzmann equation, otherwise known in plasma physics as the Vlasov equation, coupled with Maxwell's equations for the electromagnetic fields, is written for species α

$$\frac{\partial f_\alpha(\mathbf{x},\mathbf{v},t)}{\partial t} + \mathbf{v}\cdot\frac{\partial f_\alpha}{\partial \mathbf{x}} + \frac{q_\alpha}{m_\alpha}[\mathbf{E}(\mathbf{x},t)+\mathbf{v}\times\mathbf{B}(\mathbf{x},t)]\cdot\frac{\partial f_\alpha}{\partial \mathbf{v}} = 0 \tag{1.4.1}$$

$$\nabla\times\mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{1.4.2}$$

$$c^2\nabla\times\mathbf{B} = \frac{\partial \mathbf{E}}{\partial t} + \sum_\alpha \frac{q_\alpha}{\epsilon_o}\iiint \mathbf{v} f_\alpha(\mathbf{x},\mathbf{v},t)\,d\mathbf{v} \tag{1.4.3}$$

$$\nabla\cdot\mathbf{E} = \sum_\alpha \frac{q_\alpha}{\epsilon_o}\iiint f_\alpha(\mathbf{x},\mathbf{v},t)\,d\mathbf{v} \tag{1.4.4}$$

$$\nabla\cdot\mathbf{B} = 0 \tag{1.4.5}$$

where $\mathbf{E}(\mathbf{x},\mathbf{t})$ and $\mathbf{B}(\mathbf{x},\mathbf{t})$ are the electromagnetic fields generated by charged particles of mass m_α and charge q_α in free-space with permittivity ϵ_o and speed of light c (rationalized mks units). In 1d-1v with $\mathbf{B} = B_x\hat{x}$, these equations reduce to

$$\frac{\partial f_\alpha(x,u,t)}{\partial t} + u\frac{\partial f_\alpha}{\partial x} + \frac{q_\alpha}{m_\alpha}E(x,t)\frac{\partial f_\alpha}{\partial u} = 0 \tag{1.4.6}$$

$$\frac{\partial E(x,t)}{\partial x} = \sum_{\alpha} \frac{q_\alpha}{\epsilon_o} \int_{-\infty}^{\infty} f_\alpha(x,u,t)du \tag{1.4.7}$$

$$\frac{\partial E(x,t)}{\partial t} = -\sum_{\alpha} \frac{q_\alpha}{\epsilon_o} \int_{-\infty}^{\infty} u f_\alpha(x,u,t)du \tag{1.4.8}$$

and exactly describe the dynamics of collisionless periodic plasmas with their self-consistent electrostatic fields. External time-varying E-fields may be applied by adding their contribution to the self-consistent fields using $E(x,t) = E_{self}(x,t) + E_{external}(x,t)$. In the description shown in this dissertation, (1) there are no magnetic fields, (2) no externally-applied E-fields, and (3) Equation 1.4.8 is made equivalent to Equation 1.4.7 with the constraint $\int E(x,t)\,dx = 0$ for all time [Kli.1]. Given a physically-relevent initial condition $f_\alpha(x,u,0)$ (i.e. analytic in (x,u) phase-space), we wish to find $f_\alpha(x,u,t)$ by solving Equations 1.4.6 and 1.4.7.

Velocity-space filamentation, or secular increase of velocity derivatives, is a natural collisionless phenomena found in solutions of the Vlasov equation (see Appendix A). However, it is not numerically desirable since short wavelength oscillations on the discrete grid are poorly resolved by any numerical kinetic method, and hence, are poorly modeled. To combat filamentation, Klimas [Kli.2] velocity-smoothed the distribution function $f_\alpha(x,u,t)$ using the convolution,

$$\bar{f}_\alpha(x,u,t) = \mathbf{M}_\alpha * f_\alpha = \int_{-\infty}^{\infty} M_\alpha(u-u') f_\alpha(x,u',t)du' \tag{1.4.9}$$

where the normalized Gaussian filter to be used is written

$$M_\alpha(u-u') = \sqrt{\frac{1}{2\pi v_{o\alpha}^2}}\, e^{-(u-u')^2/2v_{o\alpha}^2}\,. \tag{1.4.10}$$

and $v_{o\alpha}$ is the species-dependent velocity filter width. Convolving $\mathbf{M}_\alpha$ with the Vlasov-Poisson system (Equations 1.4.6, 1.4.7) and Ampere's Law (Equation 1.4.8), and using the relations

$$u' M_\alpha(u-u') = v_{o\alpha}^2 \frac{\partial M_\alpha(u-u')}{\partial u} + u M_\alpha(u-u') \tag{1.4.11}$$

$$\frac{\partial M_\alpha(u-u')}{\partial u} = -\frac{\partial M_\alpha(u-u')}{\partial u'} \tag{1.4.12}$$

yields the filtered Vlasov-Poisson system, which was introduced previously in Section 1.1 and which we will solve in Chapter II:

$$\text{filtered Vlasov}: \frac{\partial \bar{f}_\alpha(x,u,t)}{\partial t} + u\frac{\partial \bar{f}_\alpha}{\partial x} + \frac{q_\alpha}{m_\alpha}\bar{E}(x,t)\frac{\partial \bar{f}_\alpha}{\partial u} = -v_{o\alpha}^2\frac{\partial^2 \bar{f}_\alpha}{\partial x \partial u} \tag{1.4.13}$$

$$\text{Poisson's}: \quad \frac{\partial \bar{E}(x,t)}{\partial x} = \sum_\alpha \frac{q_\alpha}{\epsilon_o}\int_{-\infty}^{\infty} \bar{f}_\alpha(x,u,t)\,du \tag{1.4.14}$$

$$\text{Ampere's}: \quad \frac{\partial \bar{E}(x,t)}{\partial t} = -\sum_\alpha \frac{q_\alpha}{\epsilon_o}\int_{-\infty}^{\infty} u\bar{f}_\alpha(x,u,t)\,du\,. \tag{1.4.15}$$

The additional $v_{o\alpha}^2$ term in Equation 1.4.13 is the only visible change to this system of equations. The "filtered" E-field $\bar{E}$ is calculated by solving Poisson's equation with the filtered distribution function $\bar{f}_\alpha(x,u,t)$. The forms of Poisson's equation and Ampere's Law are unchanged, and in fact the velocity moments

$$\int_{-\infty}^{\infty} u^p f\,du = \int_{-\infty}^{\infty} u^p \bar{f}\,du \;,\; p=0,1 \tag{1.4.16}$$

are invariant under the filter, implying that the dynamics of the electric field in the filtered simulations will remain exactly the *same* as in the unfiltered simulations, i.e. $\bar{E} = E$. The filter improves the spectral accuracy of method by making $\bar{f}(x,u,0)$ entire [Boy.1], thereby enhancing the spectral accuracy of the initial representation and helping to eliminate the need for $500-1000$ velocity modes as in other Hermite schemes [Har.1, Joy.1]. As stated earlier, this filter has been applied in a FF method by Klimas and Farrell; however, their filtered FF method exhibits a numerical instability for small temporal discretizations (see Appendix B). The FH methods discussed in this dissertation are not limited by such a filtering instability.

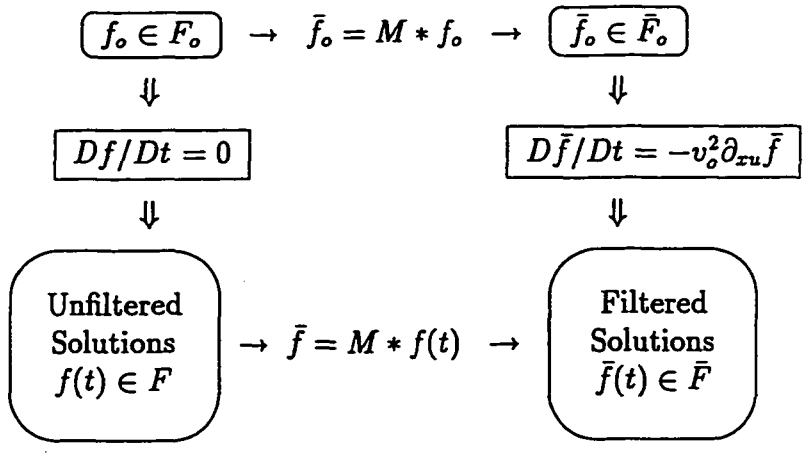

Figure 1.1: Diagram outlining the validity of solutions generated by the filtered Vlasov-Poisson system.

1.4.1 Validity of the filtered Vlasov-Poisson system

The utility of Equation 1.4.13 is that filtered solutions of the regular Vlasov equation, which are guaranteed to be smooth, naturally satisfy the filtered Vlasov equation.

Define F_o as the set of all initial distribution functions f_o which are analytic in (x, u) phase-space. The Vlasov operator D/Dt

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + \frac{q}{m}E(x,t)\frac{\partial}{\partial u} \tag{1.4.17}$$

implies a mapping of initial conditions f_o in set F_o to solutions $f(t)$ in solution set F (see left-hand side of Figure 1.1). As stated earlier, we wish to perform this mapping numerically; unfortunately, filamentation introduces discretization errors into the solution $f(t)$.

In lieu of solving $Df/Dt = 0$, we instead solve the filtered Vlasov-Poisson equations which were derived by convolving a Gaussian function with the unfiltered

Vlasov-Poisson system

$$M * \frac{Df}{Dt} = \frac{D\bar{f}}{Dt} + v_o^2 \frac{\partial^2 \bar{f}}{\partial x \partial u} = 0 \,. \tag{1.4.18}$$

An exact solution $f(t)$ of the unfiltered Vlasov equation satisfies this filtered Vlasov equation after being filtered, that is $\bar{f}(t) = M * f(t)$. Thus, we may obtain filtered solutions of the unfiltered Vlasov-Poisson system by solving the filtered Vlasov-Poisson system with filtered initial conditions. This communative property is schematically shown in Figure 1.1.

One question that comes to mind is "Can solutions $\bar{f}(t)$ of the filtered Vlasov-Poisson system be unfiltered to yield physical solutions $f(t)$?" The answer is "Yes" under the condition that for some real constant C the filtered solution $\bar{f}(t)$ satisfies the bound

$$\left|\bar{f}(\nu, t)\right| < C e^{-\nu^2 v_o^2} e^{-\gamma|\nu|} \tag{1.4.19}$$

where $\bar{f}(\nu, t)$ is the Fourier transform of $\bar{f}(u, t)$ in velocity-space, ν is the Fourier mode number, and γ is a real constant. If Equation 1.4.19 is satisfied, then we may filter-deconvolve by dividing $\bar{f}(\nu, t)$ with the Fourier-transformed filter $\hat{M} \sim e^{-\nu^2 v_o^2}$ to yield an unfiltered solution $f(t)$ that is non-singular in a strip of width γ about the real-axis, i.e. $f(\nu, t) \sim O(e^{-\gamma|\nu|})$. Note, although delta function distributions $\delta(u - u_o)$ would be interesting to study, we will only consider initially analytic functions.

For simplicity, we will omit the barred notation over $f(x, u, t)$ and explicitly set $v_o = 0$ for any non-filtered simulations using this set of filtered Vlasov-Poisson equations.

1.5 Splitting of the Vlasov-Poisson Equations

1.5.1 Heuristics of the Splitting Scheme

The solution of Vlasov equation is equivalent to the statement of conserved particle distribution values

$$f(x(t),u(t),t) = f(x(0),u(0),0) \tag{1.5.1}$$

along characteristic orbits defined by the ordinary differential equations

$$\frac{dx(t)}{dt} = u(t) \tag{1.5.2}$$

$$\frac{du(t)}{dt} = \frac{q}{m}E(x(t),t) \tag{1.5.3}$$

and parameterized by the time t. Note, we have dropped the species dependence in this discussion for simplicity. These ODEs for $\mathbf{z} = (x,u)$ are derived from the canonical Poisson bracket [Gol.1] given by $\dot{\mathbf{z}} = \{\mathbf{z},h\} = \partial_x \mathbf{z}\,\partial_u h - \partial_u \mathbf{z}\,\partial_x h$, where $h(x,u) = \frac{1}{2}u^2 + \frac{q}{m}\phi(x)$ is the specific single-particle Hamiltonian and $\phi(x)$ is the electrostatic potential.

The Vlasov-Poisson system underlies the self-consistent dynamics of a distribution of such orbits, and may itself be obtained (see Appendix C) using a non-canonical Poisson bracket [Mar.1, Mor.1] given by $\dot{F} = [F,H]$ with the corresponding total Hamiltonian

$$H(f,E) = \underbrace{\frac{1}{2}\iint u^2 f(x,u)\,dxdu}_{H_{\text{kinetic}}} + \underbrace{\frac{\epsilon_o}{2m}\int E^2(x,t)\,dx}_{H_{\text{field}}} . \tag{1.5.4}$$

To approximate the Vlasov equation with a splitting scheme, we may derive separate mappings $\mathbf{M}_x(t)$ and $\mathbf{M}_u(E,t)$ for the advection $(u\partial_x f)$ and acceleration terms $(E\partial_u f)$, respectively, to advance the distribution $\mathbf{f}(t) = f(x,u,t)$ forward one time-step Δt,

$$\mathbf{f}(t_o+\Delta t) = \mathbf{M}_x\left(\frac{\Delta t}{2}\right)\mathbf{M}_u(E,\Delta t)\mathbf{M}_x\left(\frac{\Delta t}{2}\right)\mathbf{f}(t_o) . \tag{1.5.5}$$

The advection mapping $\mathbf{M}_x(t)$ is the solution of the differential equation coming from the bracket of filtered distribution f with the *kinetic* Hamiltonian

$$\dot{f} = [f, H_{\text{kinetic}}] = \Omega_x f, \tag{1.5.6}$$

$$\text{where} \qquad \Omega_x = -u\frac{\partial}{\partial x} - v_{o\alpha}^2 \frac{\partial^2}{\partial x \partial u} . \tag{1.5.7}$$

More specifically,

$$\frac{d\mathbf{M}_x(t)}{dt} = \Omega_x \mathbf{M}_x(t) , \ \mathbf{M}_x(0) = \mathbf{I}$$

$$\Rightarrow \qquad \mathbf{M}_x \equiv \left[\mathbf{I} + t\Omega_x + \frac{t^2}{2}\Omega_x^2 + O(t^3)\right] \tag{1.5.8}$$

where $\mathbf{I}$ is the identity mapping.

Similarly, the acceleration mapping $\mathbf{M}_u(E,t)$ is the solution of the differential equation coming from the bracket of f with the *field* Hamiltonian

$$\dot{f} = [f, H_{\text{field}}] = \Omega_u f, \tag{1.5.9}$$

$$\text{where} \qquad \Omega_u = -\frac{q}{m} E(x,t)\frac{\partial}{\partial x} . \tag{1.5.10}$$

Solving this as before, we see

$$\frac{d\mathbf{M}_u(E,t)}{dt} = \Omega_u \mathbf{M}_u(E,t), \ \mathbf{M}_u(E,0) = \mathbf{I}$$

$$\Rightarrow \qquad \mathbf{M}_u \equiv \left[\mathbf{I} + t\Omega_u + \frac{t^2}{2}(\Omega_u^2 + \dot{\Omega}_u) + O(t^3)\right] . \tag{1.5.11}$$

Inserting the mappings $\mathbf{M}_x(t)$ and $\mathbf{M}_u(E,t)$ into Equation 1.5.5, we find

$$\mathbf{f}(t_o + \Delta t) = \left[I + \frac{\Delta t}{2}\Omega_x + \frac{\Delta t^2}{8}\Omega_x^2\right]\left[I + \Delta t\Omega_u + \frac{\Delta t^2}{2}(\Omega_u^2 + \dot{\Omega}_u)\right]$$

$$(\times)\left[I + \frac{\Delta t}{2}\Omega_x + \frac{\Delta t^2}{8}\Omega_x^2\right]\mathbf{f}(t_o) + O(\Delta t^3) . \tag{1.5.12}$$

As written, it appears that the operators Ω_u and $\dot{\Omega}_u$ should use updated field and current information taken from the distribution f after the first advection mapping

$\mathbf{M}_x(t)$ in Equation 1.5.5. However, the splitting scheme, as developed and used previously [Che.1, Kli.2, others], requires $\dot{\Omega}_u = 0$ in order to obtain $O(\Delta t^2)$ accuracy. The accuracy of the splitting scheme is developed below.

1.5.2 Accuracy Evaluation of the Splitting Scheme

To evaluate the accuracy of the splitting scheme (Equation 1.5.12), we must first write an exact mapping for the filtered Vlasov Poisson system (Equations 1.4.13 and 1.4.14) in an equivalent form. Writing the Vlasov equation in terms of the integro-differential operators Ω_x and Ω_u, defined in Equations 1.5.7 and 1.5.10, acting on the distribution $\mathbf{f}(t)$, we find

$$\frac{\partial \mathbf{f}(t)}{\partial t} = [\Omega_x + \Omega_u]\,\mathbf{f}(t)\,. \tag{1.5.13}$$

The time-dependence of the E-field is stated implicitly in the integral operator Ω_u. Taylor expanding Equation 1.5.13, we may advance the initial system state $\mathbf{f}(t_o)$ forward in time one time-step Δt using a Taylor expansion

$$\mathbf{f}_{exact}(t_o + \Delta t) = \mathbf{f}(0) + \Delta t\,\dot{\mathbf{f}}(t)\Big|_{t=t_o} + \frac{\Delta t^2}{2}\ddot{\mathbf{f}}(t)\Big|_{t=t_o} + O(\Delta t^3) \tag{1.5.14}$$

$$= \left[I + (\Omega_x + \Omega_{uo})\Delta t + \left((\Omega_x + \Omega_{uo})^2 + \dot{\Omega}_{uo}\right)\frac{\Delta t^2}{2} + O(\Delta t^3)\right]\mathbf{f}(t_o) \tag{1.5.15}$$

where the operators Ω_{uo} and $\dot{\Omega}_{uo} = -\dot{E}(x, t_o)\partial_u$ use E-field information from the initial system state $\mathbf{f}(t_o)$.

We may now compare this expansion to the one derived in previous section. Multiplying the factors in Equation 1.5.12 and combining terms of common order $O(\Delta t^p)$, $p = 0, 1, 2, 3$, we get

$$\mathbf{f}_{split}(\Delta t) = \left[I + (\Omega_x + \Omega_{ua})\Delta t + [\dot{\Omega}_{ua} + (\Omega_x + \Omega_{ua})^2]\frac{\Delta t^2}{2}\right]\mathbf{f}(t_o) + O(\Delta t^3) \tag{1.5.16}$$

where Ω_{ua} and $\dot{\Omega}_{ua}$ use the field and current information from the system state *after* the first advection at "time" t_a (see Figure 1.2).

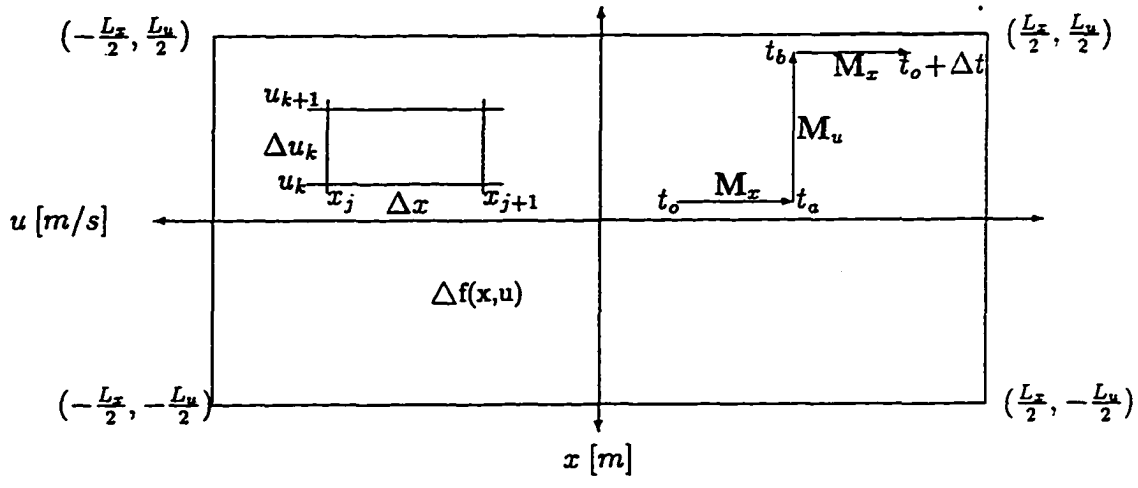

Figure 1.2: 1-D/1-v phase space grid and effective particle motion in the splitting scheme

Subtracting Equations 1.5.15 and 1.5.16, we see the difference is

$$\begin{aligned} \mathbf{f}_{exact}(\Delta t) - \mathbf{f}_{split}(\Delta t) &= \Delta t(\Omega_{uo} - \Omega_{ua}) \\ &\quad + \frac{\Delta t^2}{2}\left[(\dot{\Omega}_{uo} - \dot{\Omega}_{ua}) + (\Omega_{uo} + \Omega_x)^2 - (\Omega_{ua} + \Omega_x)^2)\right] \\ &\quad + O(\Delta t^3)\,. \end{aligned} \tag{1.5.17}$$

In the splitting method suggested by Cheng and Knorr [Che.1], the E-field is calculated at "time" t_a and held *constant* during the acceleration phase. Then because $E(x,t_a) = E(x,0) + \frac{\Delta t}{2}\dot{E}(x,0) + O(\Delta t^2)$, the acceleration operators Ω_{ua} and $\dot{\Omega}_{ua}$ satisfy

$$\Omega_{ua} = \Omega_{uo} + \frac{\Delta t}{2}\dot{\Omega}_{uo} + O(\Delta t^2) \tag{1.5.18}$$

$$\dot{\Omega}_{ua} = 0\,. \tag{1.5.19}$$

The first-order error in Equation 1.5.18 from calculating the E-field at "time" t_a cancels the second order error that arises from holding the E_a constant in Equation 1.5.17. Therefore, overall, Cheng and Knorr's splitting method has a Δt error given by

$$\mathbf{f}_{exact}(\Delta t) - \mathbf{f}_{split}(\Delta t) = O(\Delta t^3) \,. \tag{1.5.20}$$

This is the splitting method we shall use in the simulations included in this dissertation.

While the splitting algorithm described above has been used before, there is another splitting algorithm which may be worthy of attention (it is a pure-Hamiltonian splitting). This alternative $O(\Delta t^2)$ accurate splitting scheme, developed by myself with inspiration from Professor Holloway, includes the time-variation of the electric field throughout the acceleration phase. Using Poisson's equation and Ampere's Law at time t_o to calculate the E-field $E(x, t_o)$ and the current density $J(x, t_o)$, we in effect set $\Omega_{ua} = \Omega_{uo}$ and $\dot{\Omega}_{ua} = \dot{\Omega}_{uo}$ during the $\mathbf{M}_u(E, t)$ mapping and cancel the $O(\Delta t)$ and $O(\Delta t^2)$ errors in Equation 1.5.17. The explicit time-dependence of the E-field during the acceleration mapping is given by

$$E(x,t) = E(x,t_o)\, \cos\omega_p(x)t - \frac{J(x,t_o)}{\omega_p(x)}\, \sin\omega_p(x)t \tag{1.5.21}$$

where the $\omega_p(x)$ is the local plasma frequency (see Appendix D). This alternative splitting scheme is not used in this work because repeated evaluations of sines and cosines would slow down the method and yet add no accuracy to the scheme.

1.5.3 The Splitting Error

The $O(\Delta t^3)$ errors are not canceled in the splitting scheme. The difference of the exact and split $O(\Delta t^3)$ terms is

$$O(\Delta t^3) = \Big[\tilde{\Omega}_{uo} - \frac{1}{12}\left(\dot{\Omega}_{uo}(\Omega_x + \Omega_{uo}) + (\Omega_x + \Omega_{uo})\dot{\Omega}_{uo} + \Omega_x\Omega_{uo}\Omega_x + \Omega_{uo}^2\Omega_x + \Omega_x\Omega_{uo}^2\right) + \frac{1}{24}\left(\Omega_{uo}\Omega_x^2 + \Omega_x^2\Omega_{uo}\right) + \frac{1}{6}\Omega_{uo}\Omega_x\Omega_{uo}\Big]\,\mathbf{f}(t_o)\,. \quad (1.5.22)$$

If the operators commuted (i.e. $\Omega_x\Omega_u = \Omega_u\Omega_x$, etc.), this term would be $\tilde{\Omega}_{uo} - \frac{1}{3}\Omega_x\dot{\Omega}_{uo} \neq 0$, so the errors would still be $O(\Delta t^3)$.

For the alternative splitting method using time-varying fields during the acceleration mapping, we find

$$O(\Delta t^3) = \Big[\frac{1}{2}\left(\dot{\Omega}_{uo}\Omega_x - \Omega_x\dot{\Omega}_{uo}\right) + \frac{1}{2}\left(\Omega_x\Omega_{uo}\Omega_x + \Omega_x\Omega_{uo}^2 + \Omega_{uo}^2\Omega_x\right) - \frac{1}{4}\left(\Omega_x^2\Omega_{uo} + \Omega_{uo}\Omega_x^2\right) - \Omega_{uo}\Omega_x\Omega_{uo}\Big]\,\mathbf{f}(t_o)\,. \quad (1.5.23)$$

If the operators Ω_x and Ω_{uo} commuted, this expression would be zero, giving the alternative method an overall error of $O(\Delta t^4)$. However, the operators do not, in fact, commute, so we are constrained to an $O(\Delta t^2)$ accurate scheme.

1.6 The Splitting Algorithm

Rather than performing the explicit mappings in Equation 1.5.5, we advance the distribution $f(x,u,t)$ forward one time step by solving the following sequence of differential equations (DE), each taking its initial data from the previous step:

$$(1)\quad \Omega_x\left[\tfrac{\Delta t}{2}\right]f: \quad \frac{\partial f(x,u,t)}{\partial t} = -u\frac{\partial f}{\partial x} - v_o^2\frac{\partial^2 f}{\partial x\partial u},\quad f = f(x,u,t_o) \quad (1.6.1)$$

$$(2)\quad \text{ECALC}: \quad \frac{\partial E(x,t_a)}{\partial x} = \frac{q}{\epsilon_o}\int_{-\infty}^{\infty} f(x,u,t_a)\,du \quad (1.6.2)$$

$$(3)\quad \Omega_u[t;\Delta t]f: \quad \frac{\partial f(x,u,t)}{\partial t} = -\frac{q}{m}E(x,t_a)\frac{\partial f}{\partial u},\quad f = f(x,u,t_a) \quad (1.6.3)$$

$$(4)\quad \Omega_x\left[\tfrac{\Delta t}{2}\right]f: \quad \frac{\partial f(x,u,t)}{\partial t} = -u\frac{\partial f}{\partial x} - v_o^2\frac{\partial^2 f}{\partial x\partial u},\quad f = f(x,u,t_b) \quad (1.6.4)$$

The effective algorithmic sequence is shown schematically in Figure 1.2. Solution of the first DE Ω_x advects the distribution $\Delta t/2$ from time t_o to "time" t_a. Note, the temporal positions t_a and t_b are not physically realizable points in time; they are used only for labeling the algorithm steps. After the E-field is calculated using this updated information, the second DE Ω_u is solved, accelerating the distribution from "time" t_a to "time" t_b. To finish the sequence, we solve the Ω_x equation again, using the information at "time" t_b, thereby completing the unit time-step by advecting the distribution $\Delta t/2$ to time $t_o + \Delta t$.

Note, we may combine two adjacent unit time-steps in the form

$$f(x,u,t_o+2\Delta t) = \mathbf{M}_x\left(\frac{\Delta t}{2}\right)\mathbf{M}_u(t,\Delta t)\underbrace{\mathbf{M}_x(\Delta t)}_{\mathbf{M}_x^2(\Delta t/2)}\mathbf{M}_u(t,\Delta t)\mathbf{M}_x\left(\frac{\Delta t}{2}\right)f(x,u,t_o) \quad (1.6.5)$$

thereby reducing the computational effort of the whole algorithm. The accuracy of the method is still $O(\Delta t^2)$ provided the half time-step advection mappings are applied at the beginning and end of the simulation.

CHAPTER II

SPECTRAL DISCRETIZATION OF THE FILTERED VLASOV-POISSON SYSTEM

Spectral methods offer the prospect of finding high accuracy solutions for differential equations using relatively few degrees-of-freedom. In physical situations where the fine scale structures play a significant role, such as in collisionless plasma dynamics, it is necessary to develop such methods.

In this chapter, we introduce the Fourier basis and the two velocity-scaled Hermite bases (symmetric and asymmetric normalizations) to be used in the simulations. The species-dependent velocity scale U_α will be used as an optimization parameter and is on the order of the plasma thermal velocity $v_{th,\alpha}$. Velocity scaling of Hermites functions has been shown to improve their accuracy in the spectral representation of Gaussian-shaped functions [Tan.1, Boy.1]. In previous analyses of the linearized Vlasov-Poisson system, both asymmetric and symmetric Hermite representations using an optimal velocity scale yielded orders of magnitude lower errors when calculating eigenvalues of that system [Hol.2] than when not using a velocity scale at all. Others have used the asymmetric Hermite representation [Arm.1, Sho.2, Eng.1] but there has been no documented use of the symmetric Hermite conventions in plasma kinetics simulations; in these previous kinetic analyses using Hermite functions, ve-

locity space was either not scaled or the relation $U = v_{th}$ was chosen.

Also in this chapter, we derive the advection and acceleration mappings for the splitting scheme, starting with the filter Vlasov-Poisson system of equations. The advection mapping $\mathbf{M}_x$, introduced in Section 1.5, is performed by solving a Fourier-Hermite transformed differential equation for the coefficients $f_\alpha^{mn}(t)$ of the distribution $f_\alpha(x, u, t)$, where the spatial mode number m denotes the Fourier index, the velocity mode number n denotes the Hermite index, and α is the species index for the distribution. The acceleration mapping $\mathbf{M}_u$, also introduced earlier, is performed by solving a similar differential equation for the Hermite coefficients $f_\alpha^n(x, t)$ in x-space. By performing the acceleration mapping in x-space, we find a significant computational savings (see Subsection 2.3.4). The advection and acceleration mappings are both performed in transformed velocity space, saving an $O(N_x N_u^3 N_\alpha)$ operations from avoiding inverse Hermite transforms between time-steps. In Subsection 2.3.5, the splitting method is shown to be more efficient than a comparable "unsplit" method.

In this work, we will compare the velocity-scaled asymmetric and symmetric Hermite methods and find that using the optimal U-scale can improve conservation properties (see Chapter III), lower errors with respect to linear theory for a fixed discretization, and reduce the needed number of unknowns for a fixed precision level (see Chapter IV).

2.1 Fourier conventions

In these simulations, we assume that the distributions $f_\alpha(x, u, t)$ are spatially periodic. A Fourier spectral basis is, therefore, the natural choice for representation in x-space. The Fourier functions $\Phi_k(x) = [\Phi^k(x)]^* = e^{ikx(2\pi/L)}$ in the system domain

$\left[-\frac{L}{2} \leq x \leq \frac{L}{2}\right]$ satisfy the orthogonality relation

$$\frac{1}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} \Phi^m(x)\,\Phi_n(x)\,dx = \delta_n^m \tag{2.1.1}$$

where δ_n^m is the Kronecker delta function. This allows the Fourier weighted-residuals representation for a function $g(x)$

$$g(x) = \sum_{m=-\infty}^{\infty} g^m \Phi_m(x) \Longleftrightarrow g^m = \frac{1}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} g(x)\Phi^m(x)\,dx\,. \tag{2.1.2}$$

2.2 Hermite conventions

Velocity profiles of plasmas near thermal equilibrium, in general, have a Maxwell-Boltzmann shape [Tip.1, Nic.1]. For the velocity dependence of the distributions $f_\alpha(x,u,t)$, we choose basis functions which have a Gaussian as their 0^{th}-order function: the orthonormal Hermite functions. In addition, we will compare algorithms derived from the two following basis sets of bi-orthonormal Hermite polynomials:

- **symmetrically-weighted Hermites (symH)**

$$\Psi_n(v) = \Psi^n(v) = \frac{e^{-v^2/2}H_n(v)}{\pi^{1/4}\sqrt{2^n n!}} \tag{2.2.1}$$

- **asymmetrically-weighted Hermites (asymH)**

$$\Psi_n(v) = \frac{e^{-v^2}H_n(v)}{\sqrt{\pi 2^n n!}} \qquad \Psi^n(v) = \frac{H_n(v)}{\sqrt{2^n n!}} \tag{2.2.2}$$

where $H_n(v)$ is the n^{th} Hermite polynomial normalized so that $H_n(v) \sim 2^n v^n$ as $v \rightarrow \infty$ [Gra.2] and $v = u/U_\alpha$ is a dimensionless velocity, normalized by a species-dependent velocity scale U_α. The velocity scale U_α will be used as an optimization parameter.

Both Hermite bases are orthonormal in the infinite domain

$$\int_{-\infty}^{\infty} \Psi^m(v)\,\Psi_n(v)\,dv \;=\; \delta_n^m \tag{2.2.3}$$

and satisfy the recursion relations

$$\begin{bmatrix} v\Psi^n(v) \\ v\Psi_n(v) \end{bmatrix} = \sqrt{\frac{n+1}{2}} \begin{bmatrix} \Psi^{n+1}(v) \\ \Psi_{n+1}(v) \end{bmatrix} + \sqrt{\frac{n}{2}} \begin{bmatrix} \Psi^{n-1}(v) \\ \Psi_{n-1}(v) \end{bmatrix} . \quad (2.2.4)$$

In addition, the two Hermite normalizations generate different banded derivative relations

$$\text{Symmetric}: \begin{bmatrix} \frac{d\Psi^n(v)}{dv} \\ \frac{d\Psi_n(v)}{dv} \end{bmatrix} = -\sqrt{\frac{n+1}{2}} \begin{bmatrix} \Psi^{n+1}(v) \\ \Psi_{n+1}(v) \end{bmatrix} + \sqrt{\frac{n}{2}} \begin{bmatrix} \Psi^{n-1}(v) \\ \Psi_{n-1}(v) \end{bmatrix}$$

$$\text{Asymmetric}: \begin{bmatrix} \frac{d\Psi^n(v)}{dv} \\ \frac{d\Psi_n(v)}{dv} \end{bmatrix} = \begin{bmatrix} \sqrt{2n}\Psi^{n-1}(v) \\ -\sqrt{2(n+1)}\Psi_{n+1}(v) \end{bmatrix} . \quad (2.2.5)$$

These relations will be used to derive the Hermite-based algorithms in Section 2.3.

A distribution $g(u)$ may be written using a Hermite weighted-residuals representation

$$g(u) = \sum_{n=0}^{\infty} g^n \Psi_n \left(\frac{u}{U}\right) \Longleftrightarrow g^n = \frac{1}{U} \int_{-\infty}^{\infty} g(u) \Psi^n \left(\frac{u}{U}\right) du . \quad (2.2.6)$$

For spectral accuracy of the asymmetric Hermite representation (i.e. $|g^n| < \frac{1}{n^p} \; \forall \; p$), the results in Boyd [Boy.1] require the bound

$$|g(u)| < \exp\left(\frac{-u^2}{2U^2}\right) . \quad (2.2.7)$$

For example, when using asymmetric Hermite expansions of a Maxwellian distribution with thermal width v_{th}, the velocity scale must satisfy $U > U_{min} = \sqrt{2}v_{th}/2$. This analytic restriction shown in Equation 2.2.7 for the asymmetric expansion is slightly cumbersome because the maximum resolved velocity in the simulation is determined by the largest root λ_{N_u} of $H_{N_u+1}(\lambda)$ multiplied by U, limiting the resolution for fixed expansion order N_u. However, Equation 2.2.7 is not required for the symmetric expansion; for the symmetric expansion, the velocity profile of the

distribution must be infinitely differentiable and exponentially decaying as $|u| \to \infty$ (for example, $f(u)$ must fall off as $|g(u)| \leq \exp(-u/U)$ for some $U > 0$). One of the goals of this dissertation is to assess the benefits and downfalls of each Hermite method over a wide range of velocity scales U, especially with these scaling limits in mind.

2.3 Fourier-Hermite discretization

The Fourier-Hermite weighted-residuals representation for the α^{th} plasma species distribution $f_\alpha(x,u,t)$ is written,

$$\mathbf{F}^{N_x,N_u}: \; f_\alpha(x,u,t) = \sum_{m=-\frac{N_x}{2}}^{\frac{N_x}{2}-1} \sum_{n=0}^{N_u} f_\alpha^{mn}(t)\Phi_m(x)\Psi_n(v) \tag{2.3.1}$$

where $v = u/U_\alpha$ and U_α is a velocity scale. The functions $\Phi_m(x)$ and $\Psi_n(v)$ are the Fourier and Hermite weight functions, respectively, as introduced in Sections 2.1 and 2.2. The coefficients $f_\alpha^{mn}(0)$ are calculated initially by integration of the initial distribution functions $f_\alpha(x,u,0)$ with the basis functions $\Phi^m(x)$ and $\Psi^n(v)$, as shown in Section 2.4, and are advanced through time using the equations derived below in Subsections 2.3.1, 2.3.3, and 2.3.2. After initially calculating the Hermite coefficients $f_\alpha^n(x,t)$, no further Hermite transforms are required for this method (avoiding $O(N_u^2)$ operations per coefficient per time-step).

Previous unfiltered plasma kinetics methods with Hermite polynomials [Arm.2. Sho.2, others] used the asymmetric Hermite polynomials ($\Psi_n(u) \neq \Psi^n(u)$) without a velocity scale U_α as performed in this dissertation; no kinetic method based on the symmetric Hermite normalization has been developed. In the following sections, the filtered Vlasov-Poisson equations will be Fourier-Hermite transformed into forms suitable for application of the splitting method. It will be shown that optimization of

the velocity scale in either symmetric or asymmetric Hermite simulations improves the physical accuracy in the modeling of damping and growth rates as compared to linear plasma theory.

2.3.1 Fourier-Hermite Transformed Advection Mapping

Assuming a spatial periodic plasma distribution in the region $-\frac{L}{2} \leq x \leq \frac{L}{2}$ and using the orthogonality and recursion relations as shown in Section 2.2, we may Fourier-Hermite transform the filtered advection equation (Equation 1.5.7) to yield a differential equation for the coefficients $f_\alpha^{mn}(t)$,

$$\frac{1}{U_\alpha L}\int\int\left(\frac{\partial f_\alpha(x,u,t)}{\partial t}+u\frac{\partial f_\alpha}{\partial x}+v_{o\alpha}^2\frac{\partial^2 f_\alpha}{\partial x\partial u}\right)\Phi^m(x)\Psi^n(v)\,dxdu \tag{2.3.2}$$
$$=\frac{\partial f_\alpha^{mn}(t)}{\partial t}+\frac{i2\pi mU_\alpha}{L}\left[\Lambda_+^2\sqrt{\frac{n+1}{2}}f_\alpha^{m,n+1}(t)+\Lambda_-^2\sqrt{\frac{n}{2}}f_\alpha^{m,n-1}(t)\right]=0$$

where $v = u/U_\alpha$. The filtering coefficients $\Lambda_\pm$ are defined by

$$\Lambda_\pm \equiv \begin{cases} \sqrt{1\pm\frac{v_{o\alpha}^2}{U_\alpha^2}} & , \text{symH}(\pm) \\ 1 & , \text{asymH}(+) \\ \sqrt{1-\frac{2v_{o\alpha}^2}{U_\alpha^2}} & , \text{asymH}(-) \end{cases} \tag{2.3.3}$$

and are all equal to 1 for $v_{o\alpha} = 0$ (no filtering). This system of differential equations is solved using a 4^{th}-order Runge-Kutta (RK4) algorithm [Pre.1] to advance the distribution $f_\alpha(x,u,t)$ forward $\frac{\Delta t}{2}$ from time t_o to "time" t_a for the first advection mapping $\mathbf{M}_x$ and again from "time" t_b to time $t_o + \Delta t$ for the second advection mapping (refer to Equation 1.5.5). The solution of the system of equations shown in Equation 2.3.2 is called the "X-shift." As stated earlier in Section 1.5, we may combine the second X-shift from the last unit step with the first X-shift of the present unit step, for a considerable savings in computer run time (see Subsection 2.3.4).

One may ask, "Why use a high-order RK scheme if the splitting error is $O(\Delta t^3)$?" The answer is straightforward. Kinetic methods are Courant-limited by a few fast

moving particles and therefore require a time-stepping scheme with a large absolute stability region in the $\omega\Delta t$ complex plane (ω is an eigenvalue of $\dot{f}(t) = \omega f(t)$.) The lower order RK schemes have much smaller stability regions [Can.1, Dur.1]. In fact, an RK2 scheme is unstable for imaginary (oscillatory) eigenvalues. An RK3 scheme could be used for solving Equation 2.3.2, however, its absolute stability limit for imaginary eigenvalues is 1.63 times smaller than that of RK4. To obtain the same stabilty and accuracy, an RK3-based scheme would require a 46% increase in the number of operations even after realizing a savings of one right-hand-side evaluation. A more detailed operation count comparison of the RK4 and RK3 schemes is shown in Subsection 2.3.4.

Another important point to note is the truncation error introduced by the X-shift through the $n = N_u$ equation for non-zero Fourier modes m. The coefficients $f_\alpha^{m,N_u+1}(t)$ are set to zero in this algorithm. In the exact system with an infinite number of Hermite coefficients, the coefficients $f_\alpha^{m,N_u+1}(t)$ are, of course, *not* zero. Fortunately, the truncation error is small because we have a spectrally accurate representation for $f_\alpha(x,u,t)$; the error is

$$f_\alpha^{m,N_u+1} \propto O(N_u^{-\frac{1}{4}} e^{-w(2N_u+1)^\beta}) \tag{2.3.4}$$

where $\beta = \frac{1}{2}$ for functions with singularities in the complex-v plane and w is the distance from the real axis to the nearest singularity [Boy.1]. The use of filtering ($v_{o\alpha} > 0$) decreases the truncation error further by keeping $f_\alpha(x,u,t)$ entire as a function of u, making the error $f_\alpha^{m,N_u+1} \propto O(e^{-pN_u})$, for some constant $p > 0$. It is this rapid decrease in magnitude of the Hermite coefficients that makes the combination of Hermite methods *with* filtering so attractive and motivates the investigations presented in this dissertation.

An exact advection mapping is derived in Appendix D and is shown there only for completeness. The $O(N_u^2)$ operations per coefficient cost of such a mapping is prohibitive compared to 30 operations per coefficient for the RK4 algorithm. The accuracy gained by an exact advection would, of course, be destroyed by the splitting scheme in any event.

2.3.2 Fourier-Hermite Transformed Acceleration Mapping

The algorithm for the acceleration mapping $\mathbf{M}_u(t)$ is slightly different in nature. To avoid the numerically expensive convolution sum resulting from the multiplication of $E(x,t)\partial_u f_\alpha(x,u,t)$, we only Hermite transform Equation 1.5.10, thereby deriving a differential equation for the Hermite coefficients $f_\alpha^n(x,t)$ in x-space. Using the asymmetric Hermite basis with $v = u/U_\alpha$, we get

$$\frac{1}{U_\alpha}\int\left(\frac{\partial f_\alpha(x,u,t)}{\partial t} - \frac{q_\alpha}{m_\alpha}E(x,t)\frac{\partial f_\alpha}{\partial u}\right)\Psi^n(v)\,du \qquad (2.3.5)$$
$$= \frac{\partial f_\alpha^n(x,t)}{\partial t} + \frac{q_\alpha\sqrt{2n}}{m_\alpha U_\alpha}E(x,t)f_\alpha^{n-1}(x,t) = 0$$

after integration by parts and use of the derivative relation for the asymmetric Hermites. Using the symmetric Hermite basis, we get

$$\frac{\partial f_\alpha^n(x,t)}{\partial t} + \frac{q_\alpha E(x,t)}{m_\alpha U_\alpha}\left[\sqrt{\frac{n+1}{2}}f_\alpha^{n+1}(x,t) - \sqrt{\frac{n}{2}}f_\alpha^{n-1}(x,t)\right] = 0\,. \qquad (2.3.6)$$

Equations 2.3.5 and 2.3.6, solutions of which are called the "V-shifts", are both solved using a RK4 method to perform the acceleration mapping $\mathbf{M}_u(t)$ on the distribution $f_\alpha(x,u,t_a)$ at time t_a, moving it forward one Δt to time t_b (see Equation 1.5.5). We avoid the expensive $E\partial_u f$ convolution sum by solving the V-shift in x-space, giving us a computational savings of order $O(N_x)$ (see Section 2.3.4). An inverse Fourier transform must be applied to the coefficients f_α^{mn} after the first X-shift before the V-shift can be applied; subsequently, a Fourier transform on the coefficients $f_\alpha^n(x,t)$

is required after the V-shift to ready them for the second X-shift. The E-field $E(x, t_a)$ is calculated by performing a Poisson solve using the equations in Subsection 2.3.3 and held *constant* during the V-shift. Note: if the alternative splitting method from Section 1.5 was used, then the current density $J(x, t_a)$ can be calculated using equations also derived in Subsection 2.3.3. In the X-shift and V-shift, we always work with Hermite coefficients $f_\alpha^n(x, t)$ or $f_\alpha^{mn}(t)$. Because Hermite transforms are not necessary during the entire algorithm, we save an $O(N_x N_u^3 N_\alpha)$ operations per time-step over methods that do require transforms in velocity space.

An exact acceleration mapping for the asymmetric Hermite algorithm is shown in Appendix D. Again, as in the exact advection mapping, the computational cost is prohibitive (an $O(N_u)$ operations per coefficient, compared to $O(1)$ for the RK4 scheme) and would allow no gains in accuracy due to the splitting errors of the method. However, it is interesting to note that a spatially-uniform problem can be *exactly* solved (plasma oscillations) using the asymmetric Hermite method.

There is no truncation error in the asymmetric Hermite V-shift; the error comes only from the X-shift truncation. In contrast, the symmetric Hermite X-shift and V-shift have very similar formulas, both having $N_u + 1$ terms that are truncated. Enhancing spectrally accuracy of the symmetric Hermite expansion with velocity scaling or filtering, we may again have confidence that the truncation error from the V-shift is not too great.

2.3.3 Poisson's Equation and Ampere's Law Evaluations

Before V-shifting, we need to evaluate the electric field $E(x, t_a)$ at time t_a (see Figure 1.2). If the alternative splitting scheme was to be used (Subsection 1.5.2), the current density $J(x, t_a)$ at time t_a is also required. Inserting Equation 2.3.1 into

Poisson's equation 1.4.14 and using Fourier-asymmmetric Hermites, we get

$$\frac{\partial E(x,t_a)}{\partial x} = \sum_{m=-\frac{N_x}{2}}^{\frac{N_x}{2}-1} \left[\sum_\alpha \frac{q_\alpha U_\alpha}{\epsilon_o} f_\alpha^{m0}(t_a) \right] \Phi_m(x) \,. \tag{2.3.7}$$

Using the Fourier-symmetric Hermite representation, we get

$$\frac{\partial E(x,t_a)}{\partial x} = \sum_{m=-\frac{N_x}{2}}^{\frac{N_x}{2}-1} \left[\sum_\alpha \frac{q_\alpha U_\alpha}{\epsilon_o} \left(\sum_{n=0}^{N_u} I_{0n} f_\alpha^{mn}(t_a) \right) \right] \Phi_m(x) \tag{2.3.8}$$

where the coefficients I_{0n} are given by

$$I_{0n} = \int_{-\infty}^{\infty} \Psi_n(\lambda)\, d\lambda = \begin{cases} \frac{\sqrt{2}\pi^{1/4}}{(n/2)!}\sqrt{\frac{n!}{2^n}}, & n \text{ even} \\ 0, & n \text{ odd} \,. \end{cases} \tag{2.3.9}$$

Using the E-field Fourier representation, $E(x,t_a) = \sum_m E^m(t_a)\Phi_m(x)$, we may perform differentiation with respect to x

$$\frac{\partial E(x,t_a)}{\partial x} = \sum_{m=-\frac{N_x}{2}}^{\frac{N_x}{2}-1} \left(\frac{2\pi i m}{L} E^m(t_a) \right) \Phi_m(x) \tag{2.3.10}$$

and then identify the coefficients $E^m(t_a)$ in Equations 2.3.7 and 2.3.8. For the asymmetric Hermite algorithm, we find the E-field Fourier modes are

$$E^m(t_a) = \left(\frac{-iL}{2\pi m \epsilon_o} \right) \sum_\alpha q_\alpha U_\alpha f_\alpha^{m0}(t_a), \quad m \neq 0 \tag{2.3.11}$$

and for symmetric Hermites, we find

$$E^m(t_a) = \left(\frac{-iL}{2\pi m \epsilon_o} \right) \sum_\alpha q_\alpha U_\alpha \left(\sum_{n=0}^{N_u} I_{0n} f_\alpha^{mn}(t_a) \right), \quad m \neq 0 \,. \tag{2.3.12}$$

The 0^{th} Fourier mode $E^0(t_a)$ is set to zero for all time during the simulations shown in this dissertation. These coefficients $E^m(t_a)$ are then inverse Fourier transformed to $E(x,t_a)$ for use in the V-shift. If an external time-varying E-field is desired, the 0^{th} Fourier mode may be varied appropriately.

To calculate the current density, we insert the Fourier-Hermite representation in Equation 2.3.1 into Ampere's Law (Equation 1.4.8) using the asymmetric Hermite conventions to yield

$$J^m(t_a) = \sum_\alpha \left(\frac{U_\alpha^2 q_\alpha}{\sqrt{2}\epsilon_o} \right) f_\alpha^{m1}(t_a) \,. \tag{2.3.13}$$

Using symmetric Hermites, we find

$$J^m(t_a) = \sum_\alpha \left(\frac{U_\alpha^2 q_\alpha}{\sqrt{2}\epsilon_o} \right) \sum_{n=1}^{N_u} I_{1n} f_\alpha^{mn}(t_a) \tag{2.3.14}$$

where

$$I_{1n} = \int_{-\infty}^{\infty} \lambda \Psi_n(\lambda)\, d\lambda = \begin{cases} \frac{2\sqrt{2}\pi^{1/4}}{\left(\frac{n-1}{2}\right)!}\sqrt{\frac{n!}{2^n}}, & n \text{ odd} \\ 0, & n \text{ even} \,. \end{cases} \tag{2.3.15}$$

The coefficients I_{0n} and I_{1n} can be evaluated recursively from

$$I_{0n} = \sqrt{\frac{n-1}{n}} I_{0,n-2}, \qquad I_{00} = \sqrt{2}\pi^{1/4} \tag{2.3.16}$$

$$I_{1n} = \sqrt{\frac{n}{n-1}} I_{1,n-2}, \qquad I_{11} = 2\pi^{1/4} \,. \tag{2.3.17}$$

If necessary for the alternative splitting scheme, the Fourier coefficients $J^m(t_a)$ may be inverse transformed into $J(x, t_a)$ before the V-shift.

2.3.4 Operation count for the methods

In order to design an efficient and accurate numerical method, it is important to first quantify the operational cost. Memory access speeds also play a significant role in the design of array sizes and algorithms, but will not be considered here since they are architecture dependent. In this analysis, the computational time required for each "multiply" $\otimes$ and each "add" $\oplus$ is assumed to be equal.

For the X-shift, we may represent the system of equations shown in Equation 2.3.2 with the formula

$$\dot{f}_\alpha^{mn}(t) = A_\alpha^{m,n+1} f_\alpha^{m,n+1}(t) + B_\alpha^{m,n} f_\alpha^{m,n-1}(t) \tag{2.3.18}$$

where the A's and B's are appropriately defined to include all of the constants of the X-shift. The RK4 scheme requires four right-hand-side (RHS) evaluations of Equation 2.3.18 with an additional 6⊗ multiplications and 7⊕ additions of overhead. Given the number of coefficients $N_x N_u N_\alpha$, the total operational count for the X-shift is

$$\text{Xshift Ops} = [4(2\otimes + \oplus) + 6\otimes + 7\oplus]\, N_x N_u N_\alpha = 25 N_x N_u N_\alpha \,. \tag{2.3.19}$$

An RK3-based X-shift would require $19 N_x N_u N_\alpha$ total operations per time-step.

The asymmetric and symmetric V-shifts may be represented by the compact forms,

$$\dot{f}_\alpha^n(x,t) = \Gamma_\alpha^n E(x,t)\dot{f}_\alpha^{n-1}(x,t)\,, \text{ asymH} \tag{2.3.20}$$

$$= \left[-\frac{\Gamma_\alpha^{n+1}}{2}\dot{f}_\alpha^{n+1}(x,t) + \frac{\Gamma_\alpha^n}{2}\dot{f}_\alpha^{n-1}(x,t)\right] E(x,t)\,, \text{ symH} \tag{2.3.21}$$

where the $\Gamma_\alpha^n = q_\alpha\sqrt{2n}/(m_\alpha U_\alpha)$. In addition to the four RHS evaluations and the 6⊗, and the 7⊕ for the RK4 evaluation, we now have two FFT's to get the coefficients into X-space and back to Fourier space. If N_x is a power of 2, the FFT and inverse FFT cost $5 N_x N_u N_\alpha \log_2 N_x$ each. The total operational cost for the asymmetric Hermite V-shift is then

$$\text{Vshift Ops} = [4(2\otimes) + 6\otimes + 7\oplus + 10\log_2 N_x]\, N_x N_u N_\alpha \tag{2.3.22}$$

$$= [21 + 10\log_2 N_x] N_x N_u N_\alpha \tag{2.3.23}$$

and for the symmetric Hermite V-shift is

$$\text{Vshift Ops} = [4(3\otimes + \oplus) + 6\otimes + 7\oplus + 10\log_2 N_x]\, N_x N_u N_\alpha \tag{2.3.24}$$

$$= [29 + 10\log_2 N_x] N_x N_u N_\alpha \tag{2.3.25}$$

per time step. An RK3-based V-shift would require $5 N_x N_u N_\alpha$ fewer operations for the asymmetric method and $7 N_x N_u N_\alpha$ fewer operations for the symmetric method.

Additionally, avoiding the convolution sum (CS) by using an x-multiply (XM) in the V-shift (a cost of N_x multiplications per coefficient), yields a time savings of

$$\frac{t_{CS}}{t_{XM}} \approx \frac{8N_x}{4+10\log_2 N_x} = 8 \text{ for } N_x = 64 \,. \tag{2.3.26}$$

The E-field calculation requires $(2N_\alpha + 5\log_2 N_x)N_x$ operations using the asymmetric Hermite expansion and $[2N_uN_\alpha + 5\log_2 N_x]N_x$ for the symmetric Hermites expansion, including the one inverse FFT of the E-field modes into x-space.

Using these separate operation counts, the total number of operations per time-step for each method are

$$\begin{aligned} \text{Asym Ops} &= [46+10\log_2 N_x]N_xN_uN_\alpha + (2N_\alpha + 5\log_2 N_x)N_x \\ &\approx 2(23+5\log_2 N_x)N_xN_uN_\alpha \end{aligned} \tag{2.3.27}$$

$$\begin{aligned} \text{Sym Ops} &= [54+10\log_2 N_x]N_xN_uN_\alpha + (2N_uN_\alpha + 5\log_2 N_x)N_x \\ &\approx 2(28+5\log_2 N_x)N_xN_uN_\alpha \,. \end{aligned} \tag{2.3.28}$$

In this estimate, we only really need one X-shift per time-step (combine two adjacent X-shifts into one over the Δt time-step) if the unit algorithm is not halted for monitoring of conservation variables or distribution values. An RK3-based splitting scheme could save approximately $12N_xN_uN_\alpha$ operations per time-step. However, if we consider the stability limits [Dur.1]

$$[\omega\Delta t]_{RK4} = 1.63[\omega\Delta t]_{RK3} \tag{2.3.29}$$

we would require an average 1.63 additional time-steps to achieve the same stability as an RK4 method, yielding a 46% increase in the number of operations.

2.3.5 An unsplit Hermite scheme

Is it possible to gain an advantage from using a unsplit Hermite method? We may still avoid the expensive convolution sum from the $E(x,t)\partial_u f$ term by leaving the en-

tire Vlasov equation in terms of the distribution representation $f_\alpha^n(x,t)$. Writing the symmetric Hermite transformed Vlasov-Poisson equations in x-space (asymmetric Hermites could be similarly cast)

$$\dot{f}_\alpha^n(x,t) = \left[A_\alpha^{n+1}\frac{\partial f_\alpha^{n+1}}{\partial x} + B_\alpha^n\frac{\partial f_\alpha^{n-1}}{\partial x}\right] - \left[\frac{\Gamma_\alpha^{n+1}}{2}\dot{f}_\alpha^{n+1} - \frac{\Gamma_\alpha^n}{2}\dot{f}_\alpha^{n-1}\right]E(x,t) \qquad (2.3.30)$$

$$E^m(t_o) = \left(\frac{-iL}{2\pi m\epsilon_o}\right)\sum_\alpha q_\alpha U_\alpha\left(\sum_{n=0}^{N_u} I_{0n}f_\alpha^{mn}(t_o)\right) \overset{\text{IFFT}}{\Rightarrow} E(x,t) \qquad (2.3.31)$$

where $A_\alpha^n = U_\alpha\Lambda_+^2\sqrt{n/2}$, $B_\alpha^n = A_\alpha^n\Lambda_-^2/\Lambda_+^2$, and $\Gamma_\alpha^n = q_\alpha\sqrt{2n}/(m_\alpha U_\alpha)$. This equation may also be advanced forward in time from t_o to $t_o+\Delta t$ using an RK4 method while avoiding the expensive convolution sum and remaining in transformed Hermite coefficient space. The spatial derivatives in Equation 2.3.30 can be performed by

$$f_\alpha^n(x,t) \overset{IFFT}{\Rightarrow} f_\alpha^{mn}(t) \rightarrow \frac{2\pi im}{L}f_\alpha^{mn} \overset{FFT}{\Rightarrow} \frac{\partial f_\alpha^n(x,t)}{\partial x}. \qquad (2.3.32)$$

at a cost of approximately $10N_x\log_2 N_x$ operations per coefficient. The total operational cost is estimated to be $[84+40\log_2 N_x]N_xN_uN_\alpha$ for this unsplit method using RK4, which is much greater than $[56+10\log_2 N_x]N_xN_uN_\alpha$ for the symmetric Hermite split method with RK4. For $N_x = 64$, the splitting scheme yields an RK4-based method that is nearly 3 times faster! The RK4-based splitting scheme, even with its $O(\Delta t^3)$ error, is still superior to the RK4-based unsplit scheme.

However, we are *now* free to try other explicit time-stepping methods which use solutions from past time-steps (e.g. Adams-Bashforth) thereby avoiding the repeated evaluations of the right-hand side in Equation 2.3.30 which are required by RK4. The AB3 and AB4 schemes are given by [Dur.1]

$$F(t_o+\Delta t) = F(t_o) + \Delta t\left[a\dot{F}(t_o) + b\dot{F}(t_o-\Delta t) + c\dot{F}(t_o-2\Delta t) + d\dot{F}(t_o-3\Delta t)\right] \qquad (2.3.33)$$

where $F(t) = [f_\alpha^n(x,t)]$ and the constants are AB3: $(a,b,c,d) = \frac{1}{12}(23,-16,5,0)$ and AB4: $(a,b,c,d) = \frac{1}{24}(55,-59,37,-9)$. Only one RHS evaluation is required per time-step, although the storage cost is higher. For large arrays $F(t_o - p\Delta t)$, the time to retrieve stored values may be significant.

Disregarding the storage retrieval time in the operational cost estimates, the unsplit AB methods cost approximately $[15 + 10\log_2 N_x]N_xN_uN_\alpha$ operations per time-step. This is clearly cheaper than $[56 + 10\log_2 N_x]N_xN_uN_\alpha$ operations for the split RK4 method. However, the stability limits $\omega\Delta t$ imposed by the AB3 (0.72) and AB4 (0.43) methods are much smaller than that for RK4 (2.82). For comparable stability *and* accuracy in the unsplit method, the estimated workload for $N_x = 64$ is

$$\frac{W_{AB3}}{W_{RK4}} \sim \left(\frac{T^{ops}_{AB3}}{T^{ops}_{RK4,unsplit}}\right)\left(\frac{[\omega\Delta t]_{RK4}}{[\omega\Delta t]_{AB3}}\right) = \frac{75\cdot 2.82}{116\cdot 0.72} = 2.53\,. \qquad (2.3.34)$$

In this case, the unsplit AB3 method is 2.5 times slower than the split RK4 method. The need for adequate stability, in addition to the excess computations from performing FFT's for the spatial derivative, make the unsplit Hermite method inferior to the split Fourier-Hermite method demonstrated in this dissertation.

2.4 Calculation of the initial Fourier-Hermite coefficients

The coefficients $f_\alpha^{mn}(0)$ are given by integration of the distribution function $f_\alpha(x,u,0)$ with the Fourier and Hermite basis functions, $\Phi^m(x)$ and $\Psi^n(v)$,

$$f_\alpha^{mn}(0) = \frac{1}{LU_\alpha}\int_{-\infty}^{\infty}\int_{\frac{-L}{2}}^{\frac{L}{2}} f_\alpha(x,u,0)\Phi^m(x)\Psi^n(v)\,dxdv \qquad (2.4.1)$$

where $v = u/U_\alpha$, U_α is the Hermite velocity scale factor, and L is the spatial length of the system. We wish to formulate a discrete approximation of this double integral.

On a discrete x-grid defined by N_x equispaced grid points given by $X : [x_j = x_{j-1} + \Delta x,\ x_o = -L/2,\ \Delta x = L/N_x]$, we may use the trapezoidal rule to evaluate

the Fourier orthogonality relation (Equation 2.1.1),

$$\frac{1}{L}\sum_{j=0}^{N_x-1}\Phi^m(x_j)\,\Phi_n(x_j)\Delta x=\frac{1}{N_x}\sum_{j=0}^{N_x-1}\Phi^m(x_j)\,\Phi_n(x_j)=\delta_n^m\ . \tag{2.4.2}$$

Using this, we can calculate the m^{th} Fourier coefficient $f_\alpha^m(u,0)$ from the distribution sampled on this equispaced x-grid

$$f_\alpha^m(u,0)=\frac{1}{N_x}\sum_{j=0}^{N_x-1}f_\alpha(x_j,u,0)\Phi^m(x_j)\ . \tag{2.4.3}$$

Note that the minimum physical wavelength that can be simulated by any numerical code in a discrete spatial grid is $\lambda_{min}=2\Delta x$. The maximum wavelength is, of course, the actual system length L. Fast Fourier Transforms (FFT's) will be used to perform the sums in Equation 2.4.3 at a computational cost on the order of $5N_x\log_2 N_x$ operations if N_x is a power of 2.

In a system with a discrete mesh in velocity u, we may evaluate the integral (Equation 2.2.6) for the coefficients $f_\alpha^n(x,0)$ using the Gauss-Hermite quadrature rule,

$$\int_{-\infty}^{\infty}g(\lambda)w(\lambda)d\lambda=\sum_{k=0}^{N_u}g(\lambda_k)w_k+R_{N_u+1}\ ,\ w(\lambda)=e^{-\lambda^2}\ . \tag{2.4.4}$$

The quadrature error R_{N_u+1} is zero for all $g\in\mathbf{P}_{2N_u+1}$ (i.e. g is a polynomial of order $2N_u+1$). The weights w_k satisfying Equation 2.4.4 are given in standard mathematics references [Abr.1] to be

$$w_k=\frac{2^{N_u}(N_u!)\sqrt{\pi}}{(N_u+1)[H_{N_u}(\lambda_k)]^2} \tag{2.4.5}$$

where the Gauss-Hermite collocation points λ_k are the N_u+1 roots of the Hermite polynomial, $H_{N_u+1}(\lambda)=0$. The quadrature error is given by

$$R_{N_u+1}=\frac{(N_u+1)!\sqrt{\pi}}{2^{N_u+1}(2N_u+2)!}\frac{d^\ell g(z)}{dz^\ell}\ ,\quad \ell=2N_u+2,\ -\infty<z<\infty\ . \tag{2.4.6}$$

With the Hermite scale U_α, the maximum resolved velocity in the system is $u_{max} = \lambda_{N_u} U_\alpha$, where $\lambda_{N_u} \sim \sqrt{N_u}$ [Abr.1]. Interior roots of the Hermite polynomial are *not* equispaced in $[-u_{max}, u_{max}]$ but have the highest velocity resolution in the center of the system near $u = 0$.

Using these relations, distribution values $f_\alpha(x, u_k, 0)$ sampled on the N_u+1 Gauss-Hermite points $u_k = \lambda_k U_\alpha$ may be used to calculate the N_u Hermite coefficients $f_\alpha^n(x, 0)$

$$f_\alpha^n(x,0) = \sum_{k=0}^{N_u} \bar{w}_k f(x, U_\alpha \lambda_k, 0) \Psi^n(\lambda_k) \tag{2.4.7}$$

where λ_k are the Hermite collocation points and the weights $\bar{w}_k$ are defined

$$\bar{w}_k = \begin{cases} w_k e^{\lambda_k^2}, & \text{asymmetric Hermites} \\ w_k e^{\lambda_k^2/2}, & \text{symmetric Hermites}. \end{cases} \tag{2.4.8}$$

Using Equations 2.4.3 and 2.4.7, we may calculate the Fourier-Hermite coefficients $f_\alpha^{mn}(0)$ from an initial distribution $f_\alpha(x, u, 0)$ given on a discrete (x, u) grid in phase space using

$$f_\alpha^{mn}(0) = \frac{1}{N_x} \sum_{j=0}^{N_x-1} \sum_{k=0}^{N_u} \bar{w}_k f_\alpha(x_j, u_k, 0) \Phi^m(x_j) \Psi^n(\lambda_k) \tag{2.4.9}$$

This calculation is only performed once in any simulation.

2.4.1 Filtering the initial distributions

The initial distributions $f_\alpha(x, u, 0)$ are filtered using Equation 1.4.9. By a change of variables and Gauss-Hermite quadrature (see Equation 2.4.4), we may approximately evaluate this integral. Defining the substitution

$$v = \frac{u - u'}{\sqrt{2} v_{o\alpha}} \quad \Rightarrow \quad u' = u - \sqrt{2} v_{o\alpha} v \tag{2.4.10}$$

$$dv = -\frac{du'}{\sqrt{2} v_{o\alpha}} \quad \Rightarrow \quad du' = -\sqrt{2} v_{o\alpha} dv \tag{2.4.11}$$

we may integrate Equation 1.4.9 using the quadrature rule,

$$\bar{f}_\alpha(x,u,0) = \frac{1}{\sqrt{2\pi}v_{o\alpha}}\int_{-\infty}^{\infty} e^{-(u-u')^2/2v_o^2} f(x,u',0)\,du' \qquad (2.4.12)$$

$$= \frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty} e^{-v^2} f(x,u-\sqrt{2}v_{o\alpha}v,0)\,dv \qquad (2.4.13)$$

$$\approx \left(\frac{1}{\sqrt{\pi}}\right)\sum_{k'=0}^{N_u} f(x,u-\sqrt{2}v_{o\alpha}v_{k'},0)\,w_{k'} \qquad (2.4.14)$$

where $w_{k'}$ are the Gauss-quadrature weights defined in Equation 2.4.5 and the $v_{k'}$ are N_u+1 roots of $H_{N_u+1}(v)=0$. This filtering summation is more costly than in the Klimas Fourier-Fourier method [Kli.2] without a fast Hermite transform, but it is only required once in any simulation.

As stated earlier in Section 1.4, the filtered coefficients of the distribution exhibit superior spectral convergence. High-order Hermite coefficients are orders of magnitude smaller than low-order coefficients; this enhancement of the coefficients initially and during simulations is illustrated in Figures 2.1 and 2.2.

2.4.2 Exact evaluation of asymmetric Hermite coefficients

For an initial plasma distribution made up of Gaussian-shaped beams, we may exactly transform *and* filter the functions $f_\alpha(x,u,0)$ using filtered asymmetric Hermite polynomials. There is no known complement for the symmetric Hermites. The n^{th} filtered asymmetric Hermite coefficient is given by the integral

$$\bar{f}^n(x,0) = \frac{1}{U}\int_{-\infty}^{\infty} \bar{f}(x,u',0)\Psi^n\left(\frac{u'}{U}\right)\,du' \qquad (2.4.15)$$

or equivalently,

$$\bar{f}^n(x,0) = \frac{1}{U}\int_{-\infty}^{\infty} f(x,u,0)\psi^n(u)\,du \qquad (2.4.16)$$

where $\psi^n(u)$ is the filtered asymmetric Hermite function, written

$$\psi^n(u) = \int_{-\infty}^{\infty} \frac{e^{-(u'-u)^2/2v_o^2}}{\sqrt{2\pi}v_o}\Psi^n\left(\frac{u'}{U}\right)\,du' \,. \qquad (2.4.17)$$

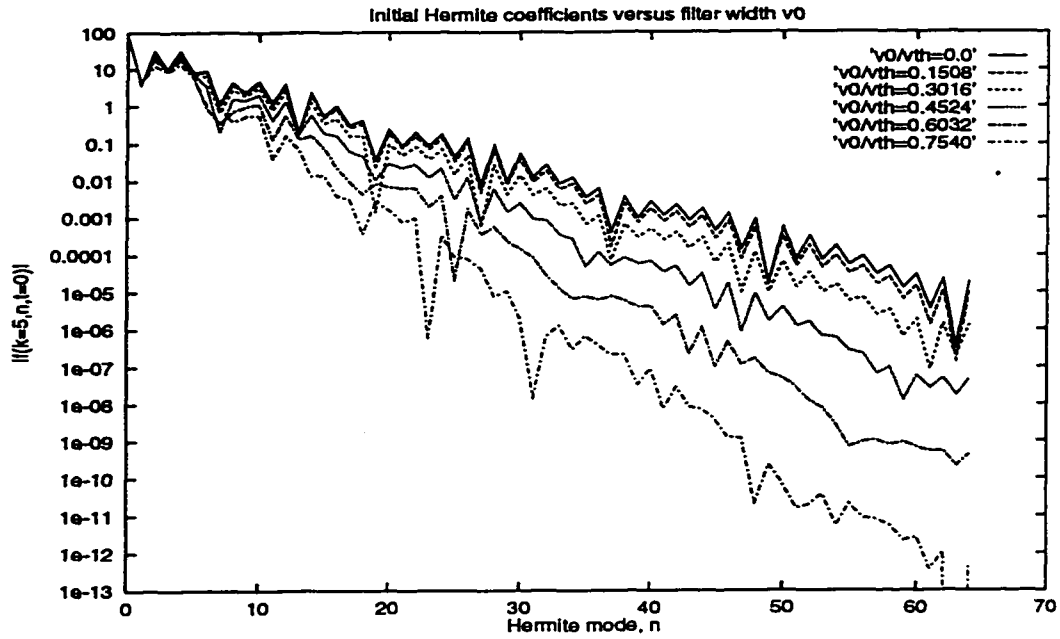

Figure 2.1: Illustration of enhanced spectral convergence of the initial Fourier Hermite coefficients $f^{kn}(0)$ due to Gaussian filtering of the initial distribution $f(x,u,0)$.

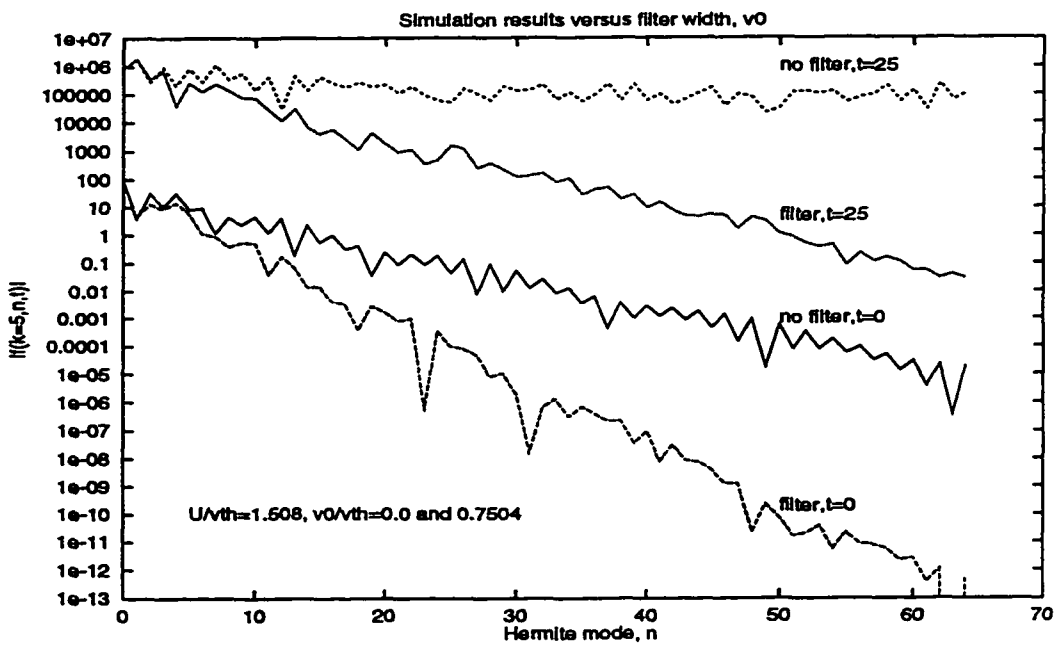

Figure 2.2: Illustration of enhanced spectral convergence due to filtering. These results are from the symmetric Hermite simulations of Chapter IV, spanning 25000 time-steps.

From Gradshteyn and Ryzhik [Gra.2], we find the integral

$$\int_{-\infty}^{\infty} e^{-(x-y)^2} H_n(\alpha x)dx = \sqrt{\pi}(1-\alpha^2)^{\frac{n}{2}} H_n\left[\frac{\alpha y}{\sqrt{1-\alpha^2}}\right] \tag{2.4.18}$$

for which the substitutions $x = u'/\sqrt{2}v_o$, $y = u/\sqrt{2}v_o$, and $\alpha = \sqrt{2}v_o/U$ inserted into Equation 2.4.17 yield

$$\psi^n(u) = [\Lambda]^n \Psi^n\left(\frac{u}{\Lambda U}\right) \tag{2.4.19}$$

where $\Lambda(v_o, U) = \sqrt{1 - 2v_o^2/U^2}$. Inserting this formula into Equation 2.4.16 shows the filtered Hermite basis ψ is the unfiltered basis Ψ with a *new* normalization on a *new* grid defined by velocity scale ΛU,

$$\bar{f}^n(x,0) = \frac{\Lambda^n}{U}\int_{-\infty}^{\infty} f(x,u,0)\Psi^n\left(\frac{u}{\Lambda U}\right)\, du\,. \tag{2.4.20}$$

Since the filtering factor $\Lambda \leq 1$, the coefficients $\bar{f}^n$ decay more rapidly than the unfiltered coefficients f^n (for fixed velocity scale ΛU), yielding a superior spectrally accurate representation for the distribution $f(x,u,0)$ in u-space.

Given an initial single-beam Maxwellian distribution with arbitrary initial spatial dependence $g(x,0)$, drift velocity v_d, and thermal velocity v_{th}, written

$$f(x,u,0) = g(x,0)\frac{e^{-(u-v_d)^2/v_{th}^2}}{\sqrt{\pi}v_{th}} \tag{2.4.21}$$

we may again use Equation 2.4.18 to evaluate Equation 2.4.20 and yield the filtered coefficients for the drifting Maxwellian beam

$$\bar{f}^n(x,0) = \frac{g(x,0)\Lambda^n}{\sqrt{\pi}v_{th}U_\alpha}\int_{-\infty}^{\infty} e^{-(u-v_d)^2/v_{th}^2}\Psi^n\left(\frac{u}{\Lambda U}\right)\, du \tag{2.4.22}$$

$$= \frac{g(x,0)}{U}\left[\frac{\gamma}{U}\right]^n \Psi^n\left(\frac{v_d}{\gamma}\right) \tag{2.4.23}$$

where $\gamma = \sqrt{(\Lambda U)^2 - (v_{th})^2} = \sqrt{U^2 - 2v_o^2 - v_{th}^2}$. Here, we used the substitutions $x = u/v_{th}$, $y = v_d/v_{th}$, and $\alpha = v_{th}/\Lambda U$.

One may be concerned by an imaginary γ in Equation 2.4.23; however, if $v_d = 0$ then imaginary γ is never a concern because $\Psi^n(0) = 0$ for odd n. For the $v_d = 0$ case, we are only constrained by the Hermite quadrature convergence limit of $U > v_{th}/\sqrt{2}$. If the initial distribution has a finite drift velocity (bump-on-tail or double-humped velocity profile) then odd Hermite modes are present and we must adhere to the limit $U > \sqrt{2v_o^2 + v_{th}^2}$ to avoid imaginary γ values. The limit $\sqrt{2v_o^2 + v_{th}^2}$ is the filter-broadened thermal velocity.

Equation 2.4.23 allows an estimate of the truncation error, assuming that $f(u)$ is a spatially uniform, drifting Maxwellian distribution given by $f(u) = e^{-(u-v_d)^2/v_{th}^2}$ for some thermal velocity v_{th} and drift velocity v_d. The truncation error for such a distribution is

$$\epsilon_{trunc}[\bar{f}(u)] = |\bar{f}(u) - \bar{f}_N(u)| = \left| \sum_{n=N+1}^{\infty} \bar{f}^n \Psi_n(v) \right| \tag{2.4.24}$$

$$< M \left| \bar{f}^{N+1} \Psi_{N+1}(v) \right| \tag{2.4.25}$$

for some constant M. For spectrally converging coefficients $\bar{f}^n$, the leading term in the series $\bar{f}^{N+1}$ is dominant since, asymptotically, the Hermite coefficients fall off like $O(e^{-an^{1/2}})$ for $a > 0$ and $n \to \infty$. The subscripted asymmetric Hermites are known to be bounded for all n and v by $|\Psi_n(v)| < 0.816e^{-v^2/2}$ [Abr.1]. So the truncation error is bounded

$$\epsilon_{trunc}[\bar{f}(u)] < 0.816M \left| \bar{f}^{N+1} e^{-u^2/2U^2} \right| \tag{2.4.26}$$

$$< \frac{0.816M}{U} \left| \left[\frac{\gamma}{U}\right]^{N+1} \Psi^{N+1}\left(\frac{v_d}{\gamma}\right) \right| \tag{2.4.27}$$

where $\gamma = \sqrt{U^2 - 2v_o^2 - v_{th}^2}$. This truncation error bound versus a normalized velocity scale U/v_{th} and Hermite expansion order N_u is illustrated in Figure 2.3. The optimal U scale lies between 1.05 and 1.1, yielding orders of magnitude reduction

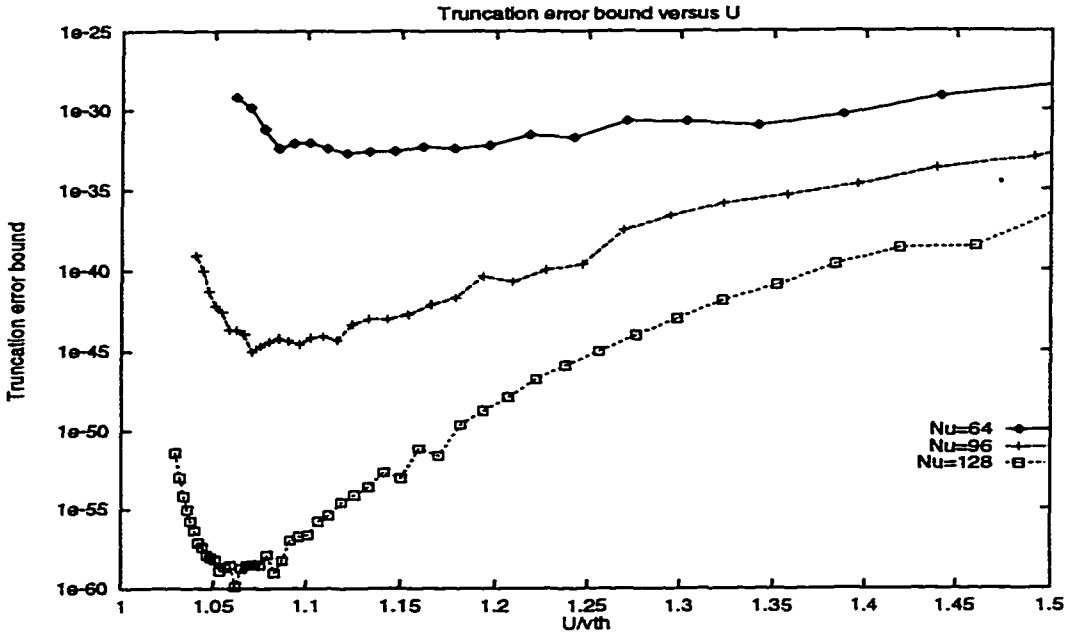

Figure 2.3: Truncation error estimates versus a normalized velocity scale U/vth for the asymmetric Hermite method. Thermal velocity, drift velocity, and filter width were arbitrarily fixed at $v_{th} = 1.32619 \times 10^7$ m/s, $v_d = 5.0 \times 10^7$ m/s, and $v_o = 0$.

in the minimum truncation error with either decreasing U or increasing N_u (actual scaling of the coefficients is $\log_e[f^{N_u}(0)] \approx O(N_u)$). In addition, the truncation error has a singularity when $\gamma = 0$; hence, we must keep $U > \sqrt{v_{th}^2 + 2v_o^2}$.

CHAPTER III

CONSERVATION PROPERTIES

Every numerical algorithm developed to model a physical system must, at least, approximate known conservation laws for that system. For collisionless plasmas, this is an especially interesting challenge because they have an infinite number of conserved quantities [Mor.2]. Although a discrete model for a physical system could never capture all of the conservation laws, we would hope that a numerical kinetic method could inherently conserve important measurable quantities such as particle number, momentum, and total energy. In this section, we derive relations for these three quantities in terms of the spectral coefficients f_α^{mn} and compare the conservation properties of the asymmetric and symmetric Hermite methods.

3.1 Particles

The total particle number for the species α is given by the integral of $f_\alpha(x,u,t)$ over all phase space. For the Fourier-Hermite methods, this is written

$$\begin{aligned} \mathrm{n}_\alpha &= \iint f_\alpha(x,u,t)\,dxdu && (3.1.1)\\ &= \sum_m \sum_n f_\alpha^{mn}(t) \iint \Phi_m(x)\Psi_n(v)\,dxdu && (3.1.2)\\ &= LU_\alpha \sum_{n=0}^{N_u} f_\alpha^{0n}(t) \int_{-\infty}^{\infty} \Psi_n(v)\,dv && (3.1.3) \end{aligned}$$

where we used $\Phi^0(x) = 1$, $v = u/U_\alpha$, and orthogonality relations outlined in Section 2.1. The Fourier scale L is the length of the spatial system.

In the following subsections, we will show that particles are always conserved in the asymmetric Hermite method. In the symmetric Hermite method, particles are conserved if the Hermite expansion order N_u is even. Note, the formulas derived below for calculating the particle number are valid for non-zero $v_{o\alpha}$ because the integral of the distribution is equivalent to the integral of the filtered distribution.

3.1.1 Asymmetric Hermite Particle Conservation

Using Equation 3.1.3, we may use the asymmetric Hermite orthogonality relations outlined in Section 2.2 to yield,

$$\mathrm{n}_\alpha = LU_\alpha f_\alpha^{00}(t) \tag{3.1.4}$$

since $\Psi_0(v) = 1$. The asymmetric Hermite X-shift and V-shift in Equations 2.3.2 and 2.3.5 show the time-derivative of the coefficient $f_\alpha^{00}(t)$ is zero for every time step, hence *particles are conserved.* Filtering does not affect the 0^{th} moment of the distribution, so this derivation is valid for non-zero $v_{o\alpha}$. The splitting error and the RK4 $O(\Delta t^5)$ error do not affect particle conservation.

3.1.2 Symmetric Hermite Particle Conservation

Again using Equation 3.1.3, we find

$$\mathrm{n}_\alpha = LU_\alpha \sum_{n,\ \mathrm{even}}^{N_u} f_\alpha^{0n}(t) I_{0n} \tag{3.1.5}$$

where the coefficients I_{0n} are non-zero for n even and are given recursively in Equation 2.3.16. Again, as in the asymmmetric Hermite method, the X-shift does not affect the particle number which is concentrated in the $m = 0$ Fourier mode. For

analysis of the particle number change in the V-shift, we begin by first Taylor expanding $\mathrm{n}(x,t_b)$ about the initial condition before the V-shift $\mathrm{n}(x,t_a)$,

$$\delta \mathrm{n}_\alpha(x) = \mathrm{n}_\alpha(x,t_b) - \mathrm{n}_\alpha(x,t_a) = \Delta t\, \dot{\mathrm{n}}_\alpha(x,t_a) + \frac{\Delta t^2}{2}\ddot{\mathrm{n}}_\alpha(x,t_a) + O(\Delta t^3)\,. \quad (3.1.6)$$

To calculate the 1^{st}-order change, we take the time-derivative of the local particle number $\mathrm{n}_\alpha(x)$ for species α, using the formula for $\dot{f}^n_\alpha(x,t)$ from the V-shift

$$\dot{\mathrm{n}}_\alpha(x,t) = -\frac{q_\alpha E(x,t_a)}{\sqrt{2}m_\alpha} \sum_{n=0,\,\mathrm{even}}^{N_u} I_{0n}\left[\sqrt{n+1}f^{n+1}_\alpha(x,t) - \sqrt{n}f^{n-1}_\alpha(x,t)\right]\,. \quad (3.1.7)$$

Rearranging this sum in terms of the coefficients $f^n_\alpha(x,t)$, we find at time t_a

$$\dot{\mathrm{n}}_\alpha(x,t_a) = -\frac{q_\alpha E(x,t_a)}{\sqrt{2}m_\alpha} \sum_{n=1,\,\mathrm{odd}}^{N_u-1} f^n_\alpha(x,t_a)\left[\sqrt{n}I_{0,n-1} - \sqrt{n+1}I_{0,n+1}\right] \quad (3.1.8)$$

$$+\; I_{0,N_u-1}\sqrt{N_u}f^{N_u}_\alpha(x,t)\,. \quad (3.1.9)$$

Using the recursion relation for I_{0n}, we can show that $\sqrt{n}I_{0,n-1} - \sqrt{n+1}I_{0,n+1} = 0$. The only remaining term is

$$\dot{\mathrm{n}}_\alpha(x,t_a) = \sqrt{N_u}I_{0,N_u-1}\, f^{N_u}_\alpha(x,t_a)\,. \quad (3.1.10)$$

Taking higher order time-derivatives of this relation, we would get terms like $\ddot{\mathrm{n}}_\alpha(x)$ in the Taylor expansion proportional to I_{0,N_u-1}; hence, $\delta\mathrm{n}_\alpha(x) \propto I_{0,N_u-1}$. If the Hermite expansion order N_u is odd then $I_{0,N_u-1} \neq 0$ and particles are not conserved to $O(\Delta t)$. If N_u is even, then, since $I_{0,N_u-1}=0$, *particles are conserved* to all orders in Δt in the limit of continuous time. The RK4-scheme limits particle conservation in the symmetric method to $O(\Delta t^5)$.

By increasing N_u in the odd N_u case, the errors in particle conservation would be reduced because $f^{N_u}_\alpha(x,t_a)$ is expected to decrease exponentially in N_u. Filtering also enhances particle conservation by improving spectral accuracy.

3.2 Momentum

The system momentum is given by the integral

$$p = \sum_\alpha m_\alpha \iint u f_\alpha(x,u,t)\,dxdu \tag{3.2.1}$$

$$= \sum_\alpha m_\alpha \sum_m \sum_n f_\alpha^{mn}(t) \iint u\Phi_m(x)\Psi_n(v)\,dxdu \tag{3.2.2}$$

$$= \sum_\alpha m_\alpha U_\alpha^2 \sum_n f_\alpha^{0n}(t) \int_{-\infty}^{\infty} v\Psi_n(v)\,dv \tag{3.2.3}$$

after using $\Phi^0(x) = 1$, $v = u/U_\alpha$, and orthogonality relations outlined in Section 2.1.

In the following subsections, we show that momentum is always conserved in the asymmetric Hermite method, while in the symmetric Hermite method momentum is conserved to $O(\Delta t)$ for even N_u and to $O(\Delta t^2)$ for odd N_u. Note, that the formulas derived below for calculating the momentum are valid for non-zero $v_{o\alpha}$ because the 1^{st} velocity moment of the distribution is invariant under the filter.

3.2.1 Asymmetric Hermite Momentum Conservation

Using Equation 3.2.3, we may evaluate the integral $\int v\Psi_n(v)dv$ to yield

$$p(t) = \frac{L}{\sqrt{2}} \sum_\alpha m_\alpha U_\alpha^2 f_\alpha^{01}(t) \tag{3.2.4}$$

where we used $u = \Psi^1(u)/\sqrt{2}$ and the orthogonality relations in Section 2.2. During the X-shift, the $m=0$ Fourier modes are unchanged, so momentum is constant.

During the V-shift however, we need to see if the value of $\delta p = p(t_b) - p(t_a)$ is zero. We know $\delta p = \dot{p}(t_a)\Delta t$ during the V-shift since $\ddot{p}(t)=0 \Rightarrow \dot{p}(t) =$ constant (see Equation 2.3.5). The time-derivative of Equation 3.2.4

$$\dot{p}(t) = \frac{1}{\sqrt{2}} \int \sum_\alpha m_\alpha U_\alpha^2 \dot{f}_\alpha^1(x,t)\,dx \tag{3.2.5}$$

$$= \int E(x,t_a) \sum_\alpha q_\alpha U_\alpha f_\alpha^0(x,t_a)\,dx \tag{3.2.6}$$

where we used the V-shift (Equation 2.3.5) to calculate $\dot{f}_\alpha^1(x,t)$. Since $\rho(x,t_a) = \sum_\alpha q_\alpha U_\alpha f_\alpha^0(x,t_a)$ we have $\dot{p} = E(x,t_a)\rho(x,t_a)$. Performing the integral over x,

$$\dot{p}(t_a) = \int_{-\frac{L}{2}}^{\frac{L}{2}} E(x,t_a)\rho(x,t_a)\,dx = \frac{\epsilon_o}{2}\int_{-\frac{L}{2}}^{\frac{L}{2}} \frac{\partial E^2(x,t_a)}{\partial x}\,dx = 0 \quad . \tag{3.2.7}$$

using Gauss' Law and periodicity of the fields. Hence, *momentum is conserved* in the V-shift and overall in this method in the limit of continuous time. The RK4-scheme limits momentum conservation in the asymmetric method to $O(\Delta t^5)$.

3.2.2 Symmetric Hermite Momentum Conservation

Again using Equation 3.2.3, the total momentum in the symmetric Hermite method is given by

$$p(t) = L\sum_\alpha m_\alpha U_\alpha^2 \sum_{n=1,\,\text{odd}}^{N_u} f_\alpha^{0n}(t) I_{1n} \tag{3.2.8}$$

where the coefficients I_{1n} are non-zero for n odd and are given recursively in Equation 2.3.17. The X-shift does not affect the $m=0$ Fourier modes, so momentum is conserved during this stage of evolution.

For the V-shift, we can analyze the conservation of momentum by Taylor expanding $p(t_b)$ about the initial condition $p(t_a)$,

$$\delta p = p(t_b) - p(t_a) = \dot{p}(t_a)\Delta t + \ddot{p}(t_a)\frac{\Delta t^2}{2} + O(\Delta t^3)\,. \tag{3.2.9}$$

To calculate the 1^{st}-order term, we may write

$$p_\alpha(x,t) = \sum_\alpha m_\alpha U_\alpha^2 \sum_{n=1,\,\text{odd}}^{N_u} f_\alpha^n(x,t) I_{1n} \tag{3.2.10}$$

$$\dot{p}_\alpha(x,t) = -\frac{m_\alpha q_\alpha U_\alpha}{\sqrt{2}} \sum_{n=1,\,\text{odd}}^{N_u} I_{1n}\left[\sqrt{n+1} f_\alpha^{n+1}(x,t) - \sqrt{n} f_\alpha^{n-1}(x,t)\right] E(x,t_a) \tag{3.2.11}$$

after using the V-shift for $\dot{f}_\alpha^n(x,t)$ (Equation 2.3.6). Rearranging the sum over the

Hermite modes n in terms of the coefficient $f_\alpha^n(x,t)$, we have

$$\dot{p}_\alpha(x,t) = -\frac{m_\alpha q_\alpha U_\alpha}{\sqrt{2}}\left[\sqrt{N_u}\, I_{1,N_u-1}\, f_\alpha^{N_u}(x,t) + \sum_{n=0,\text{ even}}^{N_u}\left(\sqrt{n} I_{1,n-1} - \sqrt{n+1} I_{1,n+1}\right) f_\alpha^n(x,t)\right] E(x,t_a)\,. \tag{3.2.12}$$

From the recursion relations for I_{0n} and I_{1n} (Equations 2.3.16 and 2.3.17), we may derive the relation $I_{1n} = \sqrt{2n} I_{0,n-1}$. From this we find $\sqrt{n} I_{1,n-1} = \sqrt{2n} I_{0n}$ and $\sqrt{n+1} I_{1,n+1} = \sqrt{2}(n+1) I_{0n}$; inserting these formulas into $\dot{p}_\alpha(x,t)$, we get

$$\dot{p}_\alpha(x,t) = -m_\alpha q_\alpha U_\alpha \left[\sqrt{\frac{N_u}{2}}\, I_{1,N_u-1}\, f_\alpha^{N_u}(x,t) + \sum_{n=0,\text{ even}}^{N_u} I_{0n} f_\alpha^n(x,t)\right] E(x,t_a)\,. \tag{3.2.13}$$

If we identify the charge density $\rho(x,t_a) = q_\alpha U_\alpha \sum_n I_{0n} f_\alpha^n(x,t_a)$ (see Equation 2.3.8), make use of $\partial_x E(x,t_a) = \rho(x,t_a)/\epsilon_o$, and integrate over x, the $O(\Delta t)$ change in total momentum at "time" t_a is

$$\dot{p}(t_a) = -\sum_\alpha m_\alpha \int q_\alpha U_\alpha \sqrt{\frac{N_u}{2}}\, I_{1,N_u-1}\, f_\alpha^{N_u}(x,t_a) E(x,t_a)\, dx \tag{3.2.14}$$

$$-\sum_\alpha \frac{\epsilon_o m_\alpha}{2} \int \frac{\partial E^2(x,t_a)}{\partial x}\, dx$$

$$= -\sum_\alpha m_\alpha q_\alpha U_\alpha \sqrt{\frac{N_u}{2}}\, I_{1,N_u-1} \int f_\alpha^{N_u}(x,t_a) E(x,t_a)\, dx\,. \tag{3.2.15}$$

where the $\frac{\partial E^2(x,t_a)}{\partial x}$ term disappears due to the periodicity of the E-field.

To find the $O(\Delta t^2)$ term, we take the time-derivative of Equation 3.2.13 and repeat the above analysis, we find

$$\ddot{p}_\alpha(x,t) = -m_\alpha q_\alpha^2 \left[\sqrt{\frac{N_u}{2}}\, I_{1,N_u-1}\, f_\alpha^{N_u-1}(x,t) + \sum_{n=0,\text{ even}}^{N_u} I_{0n}\left(\sqrt{n+1} f_\alpha^{n+1}(x,t) - \sqrt{n} f_\alpha^{n-1}(x,t)\right)\right] E^2(x,t_a) \tag{3.2.16}$$

$$= -m_\alpha q_\alpha^2 \left[\sqrt{\frac{N_u}{2}} I_{1,N_u-1} f_\alpha^{N_u-1}(x,t) + I_{0,N_u-1} f_\alpha^{N_u}(x,t)\right] E^2(x,t_a)\,. \tag{3.2.17}$$

Adding the 1^{st} and 2^{nd} order contributions together in the Taylor expansion, we find

$$\delta p \propto \Delta t I_{1,N_u-1} f_\alpha^{N_u}(x,t_a) + \frac{\Delta t^2}{2}\left[I_{1,N_u-1} f_\alpha^{N_u-1}(x,t_a) + I_{0,N_u-1} f_\alpha^{N_u}(x,t_a)\right] \ . \quad (3.2.18)$$

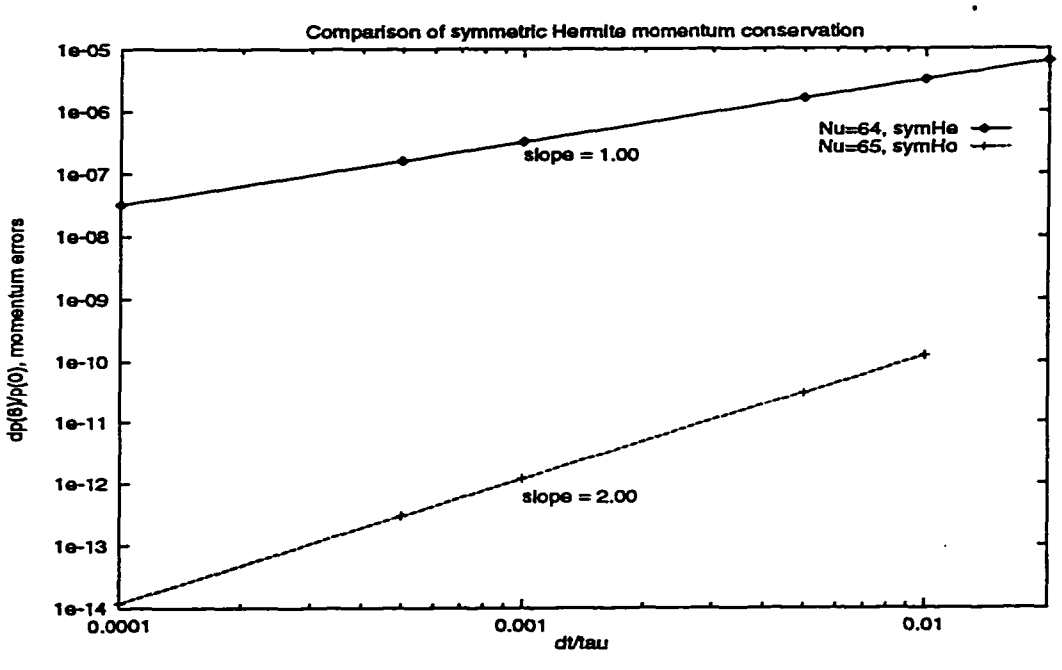

Figure 3.1: Simulations results showing momentum conservation versus Δt for symmetric Hermite methods with even and odd Hermite expansion orders.

Due to the parity of I_{0n} and I_{1n}, the momentum change during the V-shift is never zero. If the Hermite expansion order N_u is even, then $I_{1,N_u-1} \neq 0$ and the error is $O(\Delta t)$. If N_u is odd then the coefficients $I_{1,N_u-1} = 0$ and $I_{0,N_u-1} \neq 0$, so the error is $O(\Delta t^2)$. This scaling is illustrated in Figure 3.1. Again, as in the symmetric Hermite particle conservation, increasing the expansion order N_u will improve the conservation of momentum by making $f_\alpha^{N_u}(x,t)$ smaller. The slopes of the momentum conservation errors on the log-log plot are calculated to be 0.99912 (i.e. $O(\Delta t)$) for N_u even and 2.00004 (i.e. $O(\Delta t^2)$ for N_u odd. These results were calculated from simulation data from the standard bump-on-tail simulations described later in Section 4.2.

Note, if we have particle conservation by making N_u even, momentum conservation is limited to $O(\Delta t)$. If we get $O(\Delta t^2)$ accurate momentum conservation by making N_u odd, particle conservation is $O(\Delta t)$. This is an annoying limitation of the symmetric Hermite method.

We may also use filtering to increase spectral accuracy and gain orders of magnitude reduction in the momentum conservation errors. In Figure 3.2, the errors in momentum and energy (to be derived in the next few sections) are shown versus the filter width $v_{o\alpha}$. These results were calculated from even-order symmetric Hermite simulation data that will be shown in more detail in Chapter IV.

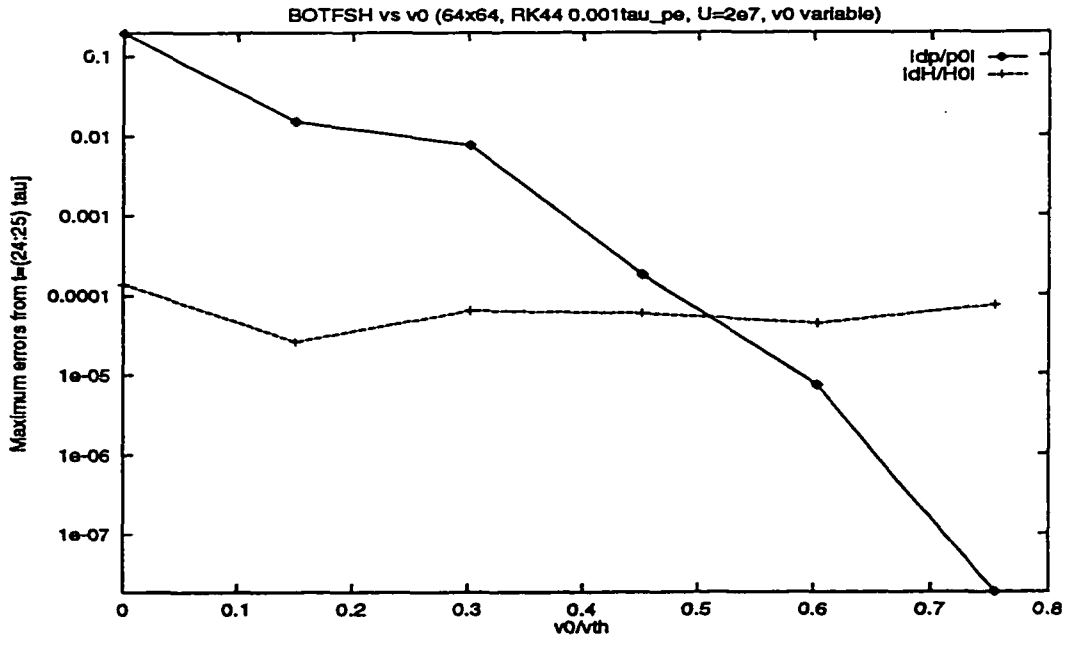

Figure 3.2: Simulations results showing momentum and energy conservation versus $v_{o\alpha}$ for even-order symmetric Hermite methods. Filtering does not affect energy conservation noticeably, but decreases the errors in momentum conservation by 7 orders-of-magnitude.

3.3 Total Energy

The total energy of the system is the sum of the kinetic and field energies,

$$H(t) = \sum_\alpha \frac{m_\alpha}{2} \iint u^2 f_\alpha(x,u,t)\,dxdu + \frac{\epsilon_o}{2}\int |E(x,t)|^2\,dx \qquad (3.3.1)$$

$$= \sum_\alpha \frac{m_\alpha}{2} \sum_m \sum_n f_\alpha^{mn}(t) \iint u^2 \Phi_m(x)\Psi_n(v)\,dxdu \qquad (3.3.2)$$

$$+ \frac{\epsilon_o}{2}\sum_m E^m(t)\int E(x,t)\Phi_m(x)\,dx$$

$$= L\sum_\alpha \frac{m_\alpha U_\alpha^3}{2}\sum_n f_\alpha^{0n}(t)\int_{-\infty}^{\infty} v^2\Psi_n(v)\,dv + \frac{\epsilon_o L}{2}\sum_m |E^m(t)|^2 \qquad (3.3.3)$$

where we have again used $\Phi_0(x) = 1$ and $v = u/U_\alpha$. In the algorithms shown in Chapter II, we use filtered coefficients $\bar{f}_\alpha^{0n}(t)$ although the "bar" is dropped for convenience. However, we really need to calculate the kinetic energy with the unfiltered distribution; therefore, we must pay special attention to the difference between kinetic energy and "filtered kinetic energy". For comparison, the unfiltered and filtered kinetic energies are defined as

$$\mathrm{KE}[f_\alpha] = L\sum_\alpha \frac{m_\alpha}{2}\int_{-\infty}^{\infty} u^2 f_\alpha^0(u,t)\,du \qquad (3.3.4)$$

$$\mathrm{KE}[\bar{f}_\alpha] = L\sum_\alpha \frac{m_\alpha}{2}\int_{-\infty}^{\infty} u^2 \bar{f}_\alpha^0(u,t)\,du\,. \qquad (3.3.5)$$

Inserting the filter convolution defined in Equation 1.4.9 for $\bar{f}_\alpha^0(u,t)$ and rearranging the integrals over u and u', we may redefine the filtered kinetic energy integral in terms of the unfiltered coefficients,

$$\mathrm{KE}[\bar{f}_\alpha] = L\sum_\alpha \frac{m_\alpha}{2}\int f_\alpha^0(u',t)\left[\int u^2 F(u-u')\,du\right]du' \qquad (3.3.6)$$

$$= L\sum_\alpha \frac{m_\alpha}{2}\int_{-\infty}^{\infty} f_\alpha^0(u',t)[(u')^2 + v_{o\alpha}^2]\,du' \qquad (3.3.7)$$

$$= L\sum_\alpha \frac{m_\alpha U_\alpha^3}{2}\sum_n f_\alpha^{0n}(t)\int_{-\infty}^{\infty}\left[v^2 + \frac{v_{o\alpha}^2}{U_\alpha^2}\right]\Psi_n(v)\,dv\,. \qquad (3.3.8)$$

The unfiltered kinetic energy in terms of the filtered coefficients $\bar{f}_\alpha^{0n}(t)$ is

$$\mathrm{KE}[f_\alpha] = \mathrm{KE}[\bar{f}_\alpha] - \underbrace{\frac{Lv_{o\alpha}^2}{2}\sum_\alpha m_\alpha U_\alpha \sum_n \bar{f}_\alpha^{0n}(t)\int_{-\infty}^{\infty}\Psi_n(v)\,dv}_{\frac{v_{o\alpha}^2}{2}\sum_\alpha m_\alpha n_\alpha} . \tag{3.3.9}$$

The difference $\left[\frac{v_{o\alpha}^2}{2}\sum_\alpha m_\alpha n_\alpha\right]$ is the kinetic energy added by an effective cold drifting beam with drift velocity $v_{o\alpha}$ (the filter width). To calculate the kinetic energy during simulations, we compute it from Equation 3.3.3 using filtered coefficients and correct it using Equation 3.3.9, if necessary.

In the following sections, we find that the total energy in the asymmetric Hermite method is conserved with an order $O(\Delta t^3)$ error due to splitting. In the symmetric Hermite method, the total energy conservation error is either $O(\Delta t)$ for odd Hermite expansion order N_u or order $O(\Delta t^3)$ for even Hermite expansion order (the latter case is limited by the splitting error).

3.3.1 Asymmetric Hermite Energy Conservation

Using the substitutions $\Psi_n(v)=1$ and $v^2=\frac{1}{2}(\sqrt{2}\Psi^2+\Psi^0)$, Equation 3.3.3 becomes

$$\begin{aligned} H(t) &= \frac{L}{4}\sum_\alpha m_\alpha U_\alpha^3\left[\sqrt{2}f_\alpha^{02}(t)+f_\alpha^{00}(t)\right]+\frac{\epsilon_o L}{2}\sum_m |E^m(t)|^2 \\ &\quad -\frac{Lv_{o\alpha}^2}{2}\sum_\alpha m_\alpha U_\alpha f_\alpha^{00}(t) \qquad (3.3.10) \\ &= \frac{L}{4}\sum_\alpha m_\alpha U_\alpha^3\left[\sqrt{2}f_\alpha^{02}(t)+\Lambda^2 f_\alpha^{00}(t)\right]+\frac{\epsilon_o L}{2}\sum_m |E^m(t)|^2 \qquad (3.3.11) \end{aligned}$$

where $\Lambda^2 = (1-\frac{2v_{o\alpha}^2}{U_\alpha^2})$. We will use this formula to calculate the total energy for the asymmetric Hermite method.

The change in potential energy over the full Δt time step is

$$\begin{aligned} \delta H_{\text{field}} &= \frac{\epsilon_o L}{2}\int\left[E^2(x,\Delta t)-E^2(x,0)\right]dx \qquad (3.3.12) \\ &= \frac{\epsilon_o L}{2}\int (E^2(x,\Delta t)-E^2(x,t_b))+(E^2(x,t_a)-E^2(x,0))\,dx \qquad (3.3.13) \end{aligned}$$

because the E-field does not change during the V-shift from time t_a to t_b but changes through both X-shifts from time 0 to t_a and from time t_b to Δt via spatial motion of the distribution. Taylor expanding $E^2(x, \Delta t)$, $E^2(x, t_b)$, and $E^2(x, 0)$ about the initial condition $E^2(x, t_a)$ and grouping terms of similar order, we get

$$\delta H_{\text{field}} = -\epsilon_o \int \Delta t E(x, t_a) J(x, t_a) + \frac{\Delta t^2}{2} E(x, t_a) \dot{J}(x, t_a)\, dx + O(\Delta t^3) \qquad (3.3.14)$$

where the Fourier modes of the current density $J(x, t_a)$ are defined in Equation 2.3.13. From here on, the goal is to similarly formulate the change in kinetic energy.

During the X-shift, the $m=0$ Fourier modes are unaffected, so the kinetic energy is constant during this step; however, during the V-shift (Equation 2.3.5), the kinetic energy changes according to

$$\delta H_{\text{kinetic}} = \sum_\alpha \frac{m_\alpha U_\alpha^3}{4} \int \left[\sqrt{2}(f_\alpha^2(x, \Delta t) - f_\alpha^2(x, 0)) + \Lambda^2(f_\alpha^0(x, \Delta t) - f_\alpha^0(x, 0))\right] dx\,. \qquad (3.3.15)$$

Since the coefficient $f_\alpha^0(x, t)$ does not change during the V-shift, the second term is zero and we will see no filtering effects on energy conservation in the asymmetric Hermite method. Taylor expanding $f_\alpha^2(x, \Delta t)$ about the initial condition $f_\alpha^2(x, 0)$ and using the V-shift formula for the time-derivatives, we find after some algebra a form for the change in the kinetic Hamiltonian

$$\delta H_{\text{kinetic}} = \epsilon_o \int \Delta t E(x, t_a) J(x, t_a) + \frac{\Delta t^2}{2} E(x, t_a) \dot{J}(x, t_a)\, dx \qquad (3.3.16)$$

which nearly cancels the Taylor expanded field Hamiltonian, differing only by $O(\Delta t^3)$. Hence, we find that the total energy for the system is conserved to $O(\Delta t^3)$, only limited by the splitting error. An illustration of asymmetric Hermite energy conservation versus time-step Δt is shown in Figure 3.3.

3.3.2 Symmetric Hermite Energy Conservation

Using Equation 3.3.3 and the filter correction, we find the total energy of the system is given in terms of the Fourier-symmetric Hermite coefficients by

$$\begin{aligned} H(t) &= \frac{L}{2}\sum_\alpha m_\alpha U_\alpha^3 \sum_{n=0,\,\text{even}}^{N_u} I_{2n} f_\alpha^{0n}(t) + \frac{\epsilon_o L}{2}\sum_m |E^m(t)|^2 \\ &\quad -\frac{L v_{o\alpha}^2}{2}\sum_\alpha m_\alpha U_\alpha \sum_n I_{0n} f_\alpha^{0n}(t) \qquad (3.3.17) \\ &= \frac{L}{2}\sum_\alpha m_\alpha U_\alpha^3 \sum_{n=0,\,\text{even}}^{N_u} (2n+\Lambda_-^2) I_{0n} f_\alpha^{0n}(t) + \frac{\epsilon_o L}{2}\sum_m |E^m(t)|^2 \qquad (3.3.18) \end{aligned}$$

where we have defined the coefficients $I_{2n} = \int v^2 \Psi_n(v) dv = (2n+1) I_{0n}$. This integral relation was derived using the recursion relation for symmetric Hermites, the definition of I_{0n}, and a little bit of algebra.

The kinetic energy is again constant during the X-shift. The change in potential energy over the full time step is again given according to Equation 3.3.14. During the V-shift however, the change in kinetic energy is given by

$$\delta H_{\text{kinetic}} = \frac{1}{2}\sum_\alpha m_\alpha U_\alpha^3 \sum_{n=0,\,\text{even}}^{N_u} (2n+\Lambda_-^2) I_{0n} \int \left(f_\alpha^n(x,t_b) - f_\alpha^n(x,t_a)\right) dx \,. \qquad (3.3.19)$$

Taylor-expanding $f_\alpha^n(x,t_b)$, using the V-shift for the time-derivatives of $f_\alpha^n(x,t)$, we see that the $O(\Delta t)$ change is

$$\begin{aligned} \delta H_{\text{kinetic}} &= -\Delta t \sum_\alpha \frac{q_\alpha U_\alpha^2}{2\sqrt{2}} \int E(x,t_a) \qquad (3.3.20) \\ &\quad (\times) \left[\sum_{n=0}^{N_u} (2n+\Lambda_-^2) I_{0n} \left(\sqrt{n+1} f_\alpha^{n+1}(x,t) - \sqrt{n} f_\alpha^{n-1}(x,t)\right)\right] dx \,. \end{aligned}$$

The $O(\Delta t^2)$ change in kinetic energy is the time-derivative of this formula. Arranging the sum over n, using the recursion relations for I_{0n} and I_{1n}, and using the formula for the current density $J(x,t) = \sum_\alpha \frac{q_\alpha U_\alpha^2}{\epsilon_o} \sum_n f_\alpha^n(x,t) I_{1n}$ we get the formula

$$\delta H_{\text{kinetic}} = -\epsilon_o \int \Delta t E(x,t_a) J(x,t_a) + \frac{\Delta t^2}{2} E(x,t_a)\dot{J}(x,t_a)\, dx \qquad (3.3.21)$$
$$+I_{1,N_u} E(x,t_a)\left[O(\Delta t\, f_\alpha^{N_u}(x,t_a)) + O(\Delta t^2\, f_\alpha^{N_u-1}(x,t_a))\right]$$
$$+O(\Delta t^3)$$

The first term cancels the change in potential energy shown in Equation 3.3.14. If N_u is even, then $I_{1,N_u}=0$ and the error in energy conservation is $O(\Delta t^3)$, limited by the splitting scheme; if N_u is odd, the change in energy is $O(\Delta t)$. Symmetric Hermite energy conservation versus time-step Δt is shown in Figure 3.3 and compared to the asymmetric Hermite method. The slopes of the local energy error on the log-log plot

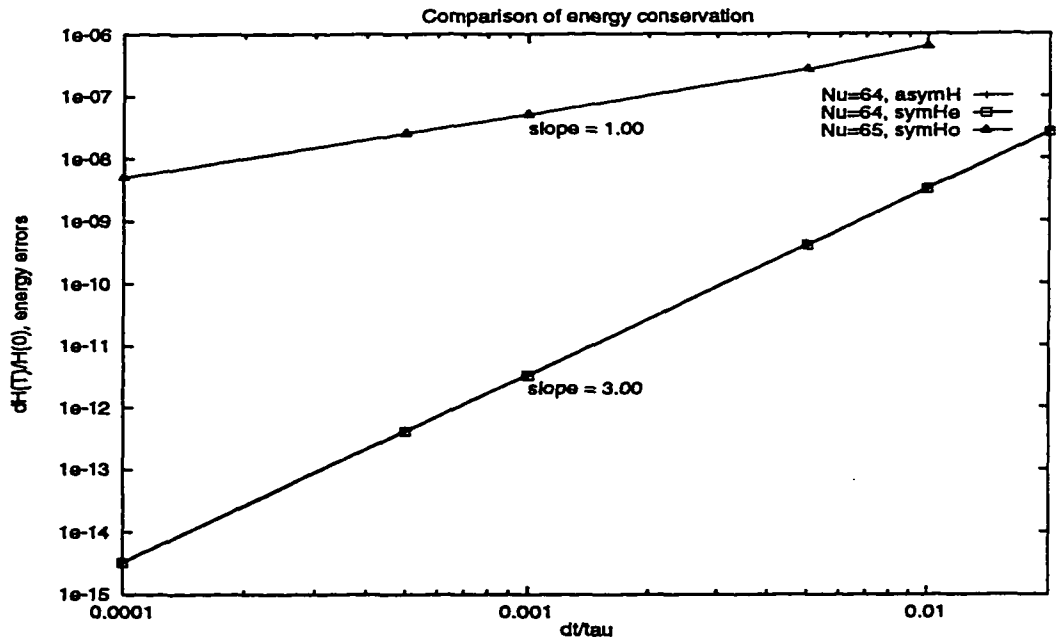

Figure 3.3: Simulation results showing energy conservation versus Δt from asymmetric and symmetric Hermite methods (both even and odd expansion order N_u are shown). The asymmetric Hermite and even symmetric Hermite data overlap.

are calculated to be 2.999095 (i.e. $O(\Delta t^3)$) for N_u even and 1.001267 (i.e. $O(\Delta t)$) for

N_u odd. These calculations were performed on data from the standard bump-on-tail simulations described later in Section 4.2.

CHAPTER IV

SIMULATIONS

Landau damping and the growth of some electrostatic waves, two phenomena in collisionless plasmas with similar physical mechanisms [Nic.1], are the result of a resonant coupling between the phase velocity of an electrostatic wave and the local velocity profile of charged particles. Both phenomena have been predicted theoretically, measured experimentally, and observed in computer simulations (for examples, see [Che.1, Sti.1, Bir.2]). As stated earlier, the goal of this dissertation is to develop efficient and accurate computer methods which can observe these, and other, phenomena in collisionless plasmas; in this chapter, we will demonstrate that the Fourier-Hermite methods are well-suited for the modeling of these 1d-1v collisionless phenomena.

We begin by assessing the ability of the Fourier-Hermite algorithms to capture the physics of Landau damping; this is done by modeling the evolution of an initially perturbed, yet stable, equilibrium Maxwellian velocity profile [Pen.1] as described in Section 4.1. Likewise, by analyzing the growth and saturation of electrostatic waves in collisionless plasmas, we continue our study of the FH algorithms in Section 4.2 by perturbing an unstable, equilibrium bump-on-tail velocity profile and following the subsequent evolution of the plasma. In both sections, fidelity of the solutions will be

based on comparisons to linear kinetic theory, the Klimas Fourier-Fourier code, and the results of one-dimensional (1d-1v) PIC simulations using ES1 [Bir.2].

4.1 Landau Damping Simulations

In this section, the initial input distribution to be used is a simple Maxwellian in velocity space with some spatially-periodic profile $g_\alpha(x,0)$ for species α

$$f_\alpha(x,u,t) = \frac{g_\alpha(x,0)}{\sqrt{\pi}v_{th,\alpha}} \exp\left[-\frac{u^2}{v_{th,\alpha}^2}\right] \tag{4.1.1}$$

defined by the thermal velocity $v_{th,\alpha}$ and an arbitrary spatial profile $g_\alpha(x,0)$ to be given below. In these simulations, we will analyze an electron plasma, charge-neutralized by setting the 0^{th}-order Fourier mode of the E-field to zero. Physically relevent quantities, such as electron charge and mass, permittivity of free space, and the speed of light, are defined in MKS units with the values listed in Table 4.1. Because only one species is used in these simulations, the species subscript α is dropped for convenience. A sample Maxwellian profile is shown in Figure 4.1.

The initial spatial dependence $g(x,0)$ is uniform in x with a small cosinusoidal perturbation $\epsilon\cos(Kx)$, and is written in the form

$$g(x,0) = \mathrm{n}\left[1 + \epsilon\cos\left(\frac{2\pi kx}{L}\right)\right] \tag{4.1.2}$$

where n is the number of particles, ϵ is the perturbation amplitude, k is the stimulated mode number, and L is the system length. From this formula, we see that the wavenumber to be stimulated is $K = 2\pi k/L$. To simplify the analysis, only one mode will be stimulated in any simulation.

The standard input deck for the Landau damping simulations is shown in Table 4.1. The number density, thermal width, mass, and charge were fixed for all simulations using the Maxwellian distribution. In these simulations, all times are

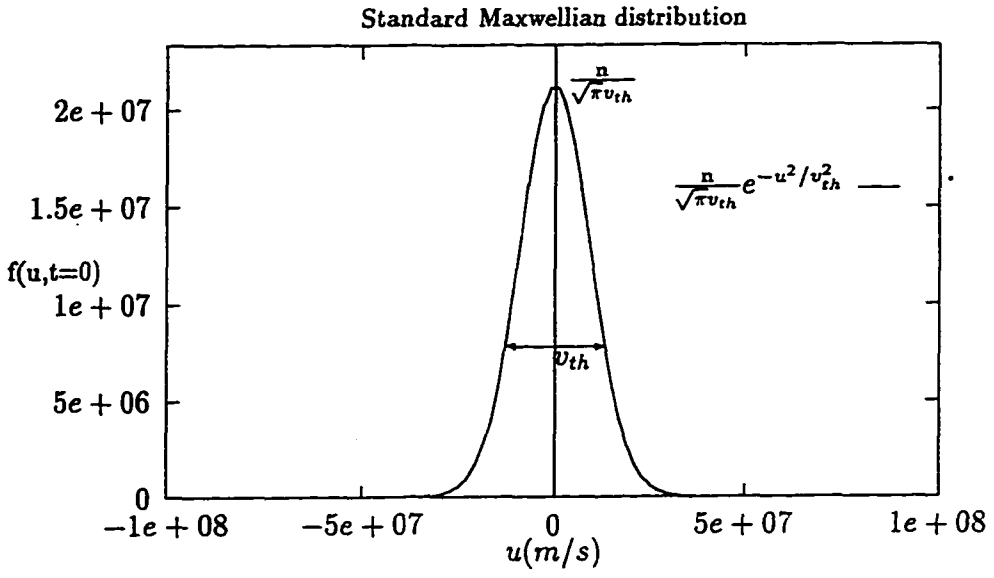

Figure 4.1: A Maxwellian velocity profile.

Parameter	Symbol	Value used [units]
number density	n	5×10^{14} [sheets/m]
thermal velocity	v_{th}	1.32619×10^{7} [m/s]
electron mass	m_e	$9.1095e \times 10^{-31}$ [kg]
electron charge	e	-1.60219×10^{-19} [C]
permittivity	ϵ_o	$8.8541878 \times 10^{-12}$ [F/m]
spatial resolution	N_x	64
velocity resolution	N_u	64 or 65
temporal resolution	Δt	0.001 [τ_{pe}, sec]
spatial length	L	1 [m]
velocity scale	U	1.0×10^{7} [m/s]
velocity filter width	v_o	0.0 [m/s]
perturbation amplitude	ϵ	10^{-5}
perturbation mode number	k	15

Table 4.1: Standard values for simulation of Landau damping in an electron plasma with a Maxwellian velocity profile

given in units of plasma period τ_{pe}, which for the standard electron density n is

$$\tau_{pe} = \frac{2\pi}{\omega_{pe}} = 2\pi\sqrt{\frac{\epsilon_o m_e}{e^2 \mathrm{n}}} \approx 4.98085 \text{ ns} . \tag{4.1.3}$$

For numerical testing, the spatial resolution N_x (the number of x grid points or

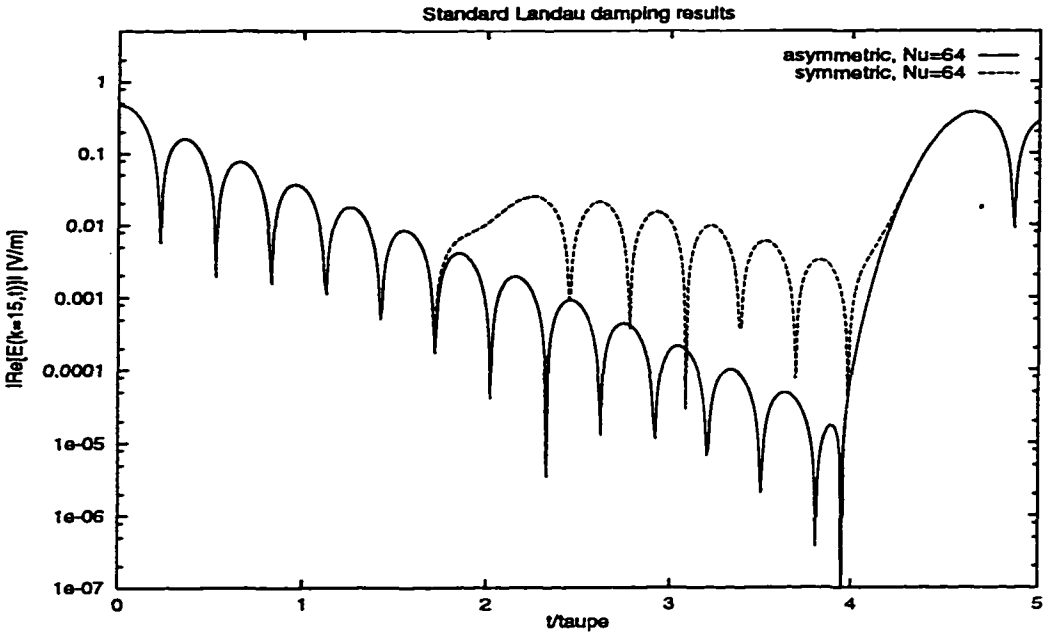

Figure 4.2: A plot of Landau damping, using the standard parameters as in Table 4.1 for both the asymmetric and symmetric Hermite methods. Recursion of the E-field is evident in this figure.

Fourier modes), the velocity resolution N_u (the number of u grid points or Hermite modes), the time step Δt, the Hermite scale factor U, and the filter width v_o will be varied, allowing accuracy comparisons of the simulation results to linear Landau damping theory. The stimulated mode k and the perturbation amplitude ϵ will also be varied in order to generate Landau damping dispersion relations.

Symmetric and asymmetric Hermite simulations using this standard input deck resulted in E-field mode evolutions shown in Figure 4.2, showing $|\Re[E(k=15,t)]|$ versus time t. Assuming an E-field dependence of

$$E(k,t) \sim e^{i\omega t+\gamma t} \rightarrow \Re[E(k,t)] \sim e^{\gamma t}\cos(\omega t) \tag{4.1.4}$$

we may calculate the oscillation frequency ω and damping or growth rate γ. From peaks 2 and 4 (i.e. one full oscillation), the oscillation frequency and damping rate

where found to be $\omega = 1.675\omega_{pe}$ and $\gamma = -0.3942\omega_{pe}$, respectively. Both methods exhibit the Landau damping phenomena; however, after some time, we see that the E-field grows sharply, returning nearly to its initial value (here, $|E(k, t_{final})| = 0.78|E(k, t_{initial})|$). This sharp rise in the amplitude of the E-field is known as *recursion*, a well-known numerical problem seen in the simulation of Landau damping [Gra.1, Kno.2, Joy.1, others]. For the standard input parameters, the symmetric Hermite method recurs before the asymmetric Hermite method. However, the symmetric Hermite methods, as we will soon see, can outperform the asymmetric Hermite method with proper selection of the velocity scale U and filter width v_o.

4.1.1 Recursion time versus velocity resolution

Recursion of the E-field in these Fourier-Hermite simulations is a failure of the Hermite spectral discretization to resolve all possible velocities u. Because we have a discrete grid in velocity generated by the Hermite discretization, we have a corresponding discrete spectrum of eigenvalues λ coming from the linearized Vlasov-Poisson system. For a stable velocity profile, such as the Maxwellian distribution, these eigenvalues generate only oscillatory solutions (i.e. $\lambda = \omega$). Two neighboring modes ω_1 and ω_2 of equal amplitude A separated by $\delta = \omega_2 - \omega_1 \approx k\Delta u$ can, at best, only destructively interfere and contribute to the field like

$$E \propto Ae^{\bar{\omega}} \left[e^{it\delta} + e^{-it\delta}\right] \tag{4.1.5}$$

where $\bar{\omega} = \frac{\omega_2+\omega_1}{2}$. At the time $T_{recur}(\omega_1, \omega_2) = \pi/\delta$, known as the *recursion time*, these modes cancel each other. After this time, their contribution to the field grows. Landau damping requires a continuum of particle velocities in order to couple to the phase velocity of the electrostatic wave and to subsequently phase-mix away the initially stimulated electric field mode [Hol.2]. Without some source of damp-

ing (generation of non-oscillatory eigenvalues) through either collisions or numerical damping, recursion cannot be avoided. However, we will see that increasing the Hermite expansion order N_u can lengthen the time to recursion. Also, proper selection of the velocity scale U or the filter width v_o can increase the recursion time for a given Hermite expansion order.

Since the recursion time is inversely proportional to the separation $\delta \approx k\Delta u$ between the modes, decreasing the wavenumber k or increasing the velocity resolution can delay the onset of recursion. Delay of recursion by increasing the Hermite expansion order N_u is illustrated in Figure 4.3, with results from the symmetric Hermite method. We do not see a linear increase in the recursion time with N_u because the velocity resolution for the Hermite method goes like

$$\Delta u \approx \frac{u_{max}}{N_u} \sim \frac{U\sqrt{N_u}}{N_u} = \frac{U}{\sqrt{N_u}} \tag{4.1.6}$$

and so the recursion time goes as $T_{\text{recur}} \sim \sqrt{N_u}/U$.

We may also increase velocity resolution (and therefore, the recursion time) by decreasing the Hermite velocity scale U. A plot of recursion time versus U is shown in Figure 4.4 for both the asymmetric and symmetric methods. Some important features to note in this figure are:

- For both methods, the recursion time increases linearly as the velocity scale U decreases, until some minimum velocity scale value denoted U_{opt}.
- For the symmetric Hermite method, U_{opt} is determined by the point at which the maximum resolved velocity $u_{max} \propto U$ begins to become too small to accurately represent the Maxwellian profile. For $U_{opt} = 0.377v_{th}$, we find $f(u_{max})/f(0) \approx 10^{-7}$.

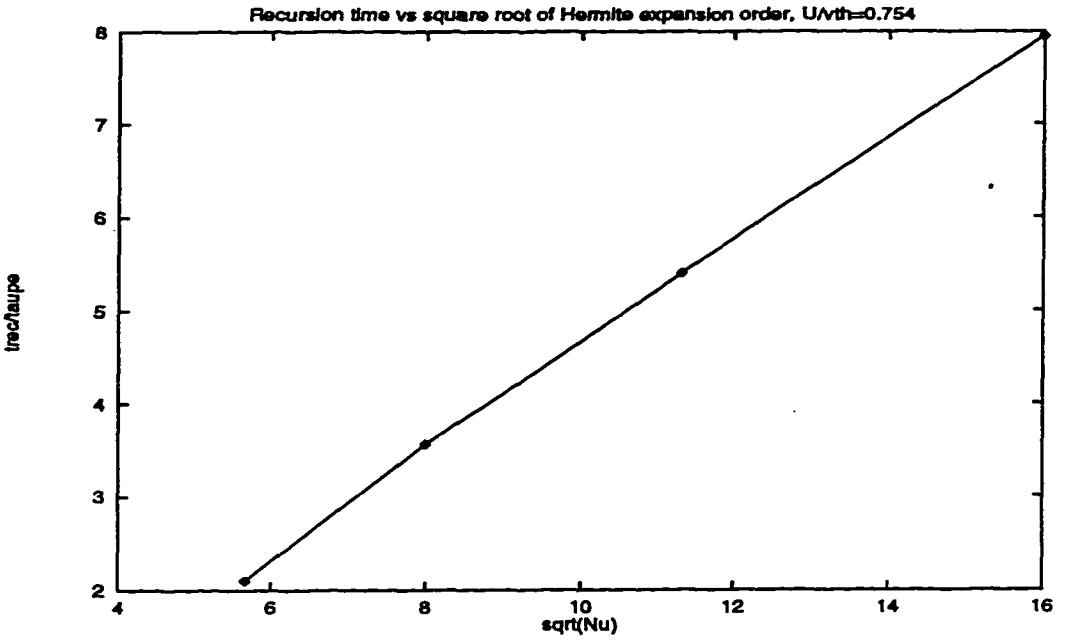

Figure 4.3: A plot of Landau damping recursion time versus the square root of the Hermite expansion order N_u for the symmetric Hermite method. The asymmetric Hermite method yields similar results.

- For the asymmetric Hermite method, if the velocity scale violates the bound $U/v_{th} < \sqrt{2}/2$ then the Hermite coefficients diverge. This U_{min} limit determines the U_{opt} value for asymmetric Hermites.

- For $N_u = 64$, the optimal U scale for the asymmetric Hermites is approximately *twice* that of the symmetric Hermites. Symmetric Hermites can attain higher velocity resolution since they are not limited by the asymmetric divergence bound $U_{min} = v_{th}/\sqrt{2}$.

Previous Fourier-Hermite algorithms did not utilize a variable velocity scale U; by allowing U to be decreased below $U = 1$, we can increase the recursion time up to 3 times during Landau damping simulations with no additional computational effort.

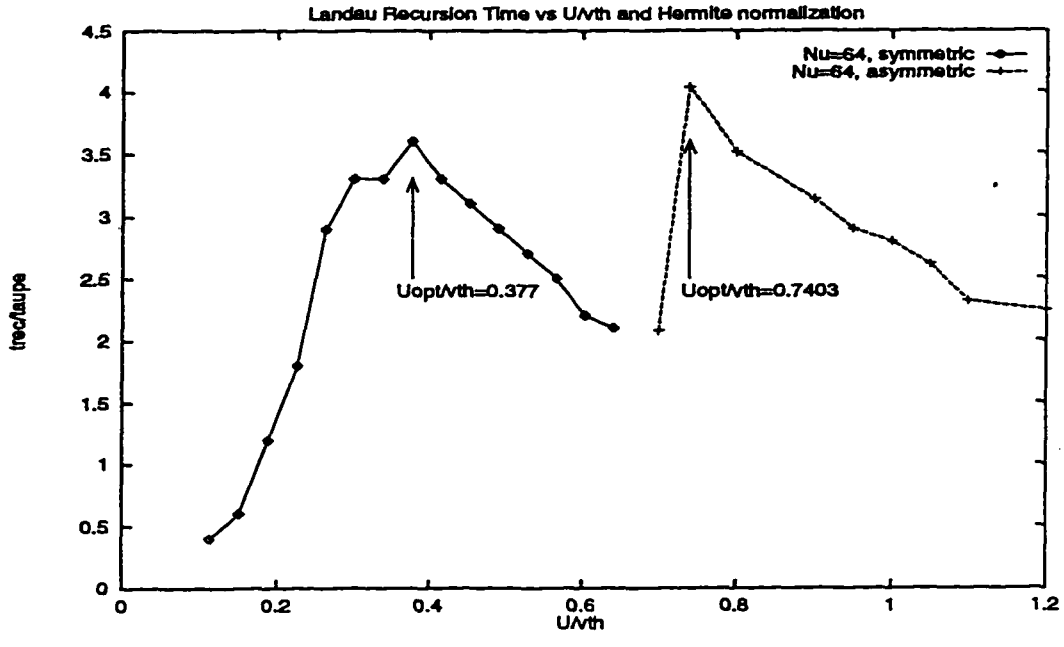

Figure 4.4: A plot of Landau damping recursion time versus the Hermite velocity scale factor U/v_{th}.

4.1.2 Recursion and filtering

We recall from Section 1.4 that using the filtered equations can yield higher accuracy simulations by improving the spectral convergence of the Hermite coefficients. However, for the asymmetric Hermite method, it is possible to show that the equations of evolution are invariant under the operation of the filter. Dynamically then, the velocity scale U and the filter width v_o play very similar roles. The filter changes only the spectral convergence of the Hermite coefficients, reducing the truncation error introduced by Hermite discretizations.

Scaling all of the Fourier-Hermite coefficients $K^{mn}(t) = [U]^{n+1} f^{mn}(t)$ with the velocity scale U raised to the power $n+1$ (the other superscripts, as usual, denote the Fourier-Hermite mode numbers), we find that the advection and acceleration

equations for the asymmetric Hermite method become

$$\frac{\partial K^{mn}(t)}{\partial t} = -\frac{i\sqrt{2}\pi m}{L}\left[\sqrt{n+1}K^{m,n+1}(t) + U^2\Lambda^2\sqrt{n}K^{m,n-1}(t)\right] \quad (4.1.7)$$

$$\frac{\partial K^{n}(x,t)}{\partial t} = \frac{q_e\sqrt{2n}}{m_e}E(x,t)K^{n-1}(x,t) \quad . \quad (4.1.8)$$

where $U^2\Lambda^2 = U^2 - 2v_o^2$. If we hold $U^2\Lambda^2$ constant while increasing the filter width v_o, these equations do not change. With $U^2\Lambda^2$ held constant, filtering only affects the initial conditions $K^{mn}(0)$ for the asymmetric Hermite method (see Equation 2.4.20).

If we find very low errors during unfiltered asymmetric Hermite simulations (e.g. long recursion time for a particular velocity scale value of U_o with $v_o = 0$), then we may achieve nearly the same solution if we select another velocity scale $U(v_o)$ for filtered simulations using the formula

$$U(v_o) = \sqrt{U_o^2 + 2v_o^2}\,. \quad (4.1.9)$$

Confirmation of this $U(v_o)$ scaling is shown in Figure 4.5 with Landau damping E-field mode evolution versus time for various values of the advection filtering function $\Lambda = \sqrt{1 - 2v_o^2/U^2}$ with $U^2\Lambda^2$ constant. In the simulations shown, the Landau damping dynamics are nearly identical. The maximum measured difference between the E-field mode amplitudes at time $t = 3\tau_{pe}$ is less than 0.1%. If the filter width v_o is varied while holding U constant, the E-field evolution shown in Figure 4.5 will not change but the recursion times will decrease.

We see that filtered asymmetric Hermite simulations, other than having reduced truncation errors, are equivalent to the unfiltered simulations with a good choice of velocity scale U_o. Since filtering reduces truncation errors in the initial conditions, we should find the best U_o velocity scale from unfiltered asymmetric Hermite simulations and use Equation 4.1.9 to pick the U scale for filtered asymmetric Hermite

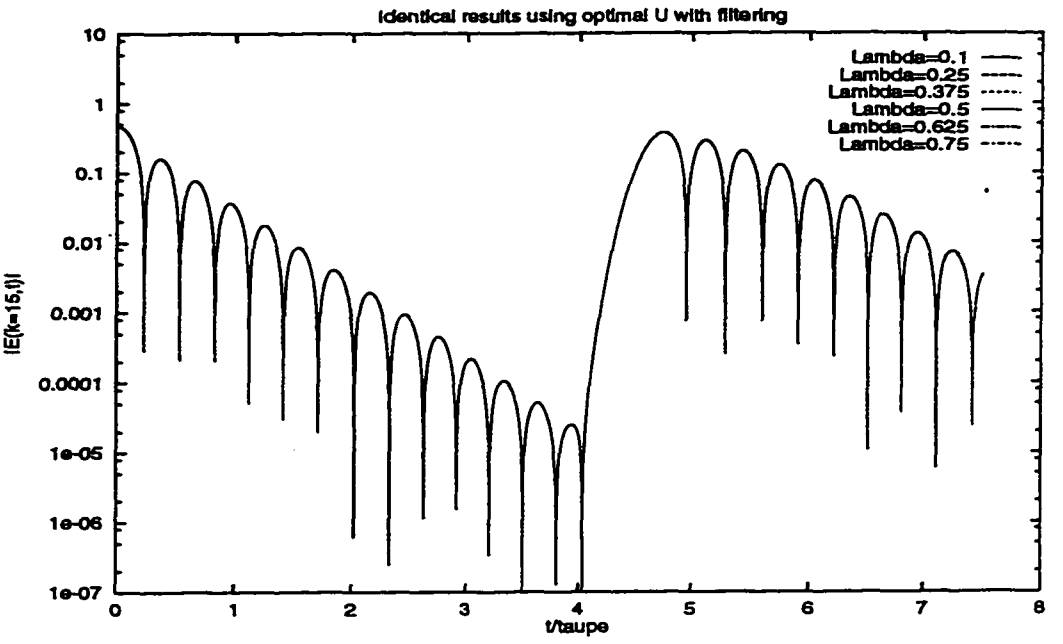

Figure 4.5: For the asymmetric Hermites, variation of the velocity scale U according to Equation 4.1.9 yields identical Landau damping dynamics (maximum difference in mode amplitude is less than 0.1%).

simulations.

We cannot cast the symmetric Hermite equations into a filter-invariant form as we could for Equations 4.1.7 and 4.1.8. However, we may guess an optimal filtering relation for fixed velocity scale U_o by looking at the symmetric Hermite advection equation,

$$\frac{\partial f^{mn}(t)}{\partial t} = -\frac{i\sqrt{2}\pi m}{L}\left[U\Lambda_+^2\sqrt{n+1}f^{m,n+1}(t) + U\Lambda_-^2\sqrt{n}f^{m,n-1}(t)\right] \qquad (4.1.10)$$

where the coefficients $\Lambda_\pm^2$ are defined in Equation 2.3.3. Setting $U\Lambda_-^2 = U_o$ and solving, we find

$$U(v_o) = \frac{U_o}{2} + \sqrt{\left(\frac{U_o}{2}\right)^2 + v_o^2}\ . \qquad (4.1.11)$$

Empirically, we find that this scaling yields improved recursion times for the filtered symmetric Hermite simulations.

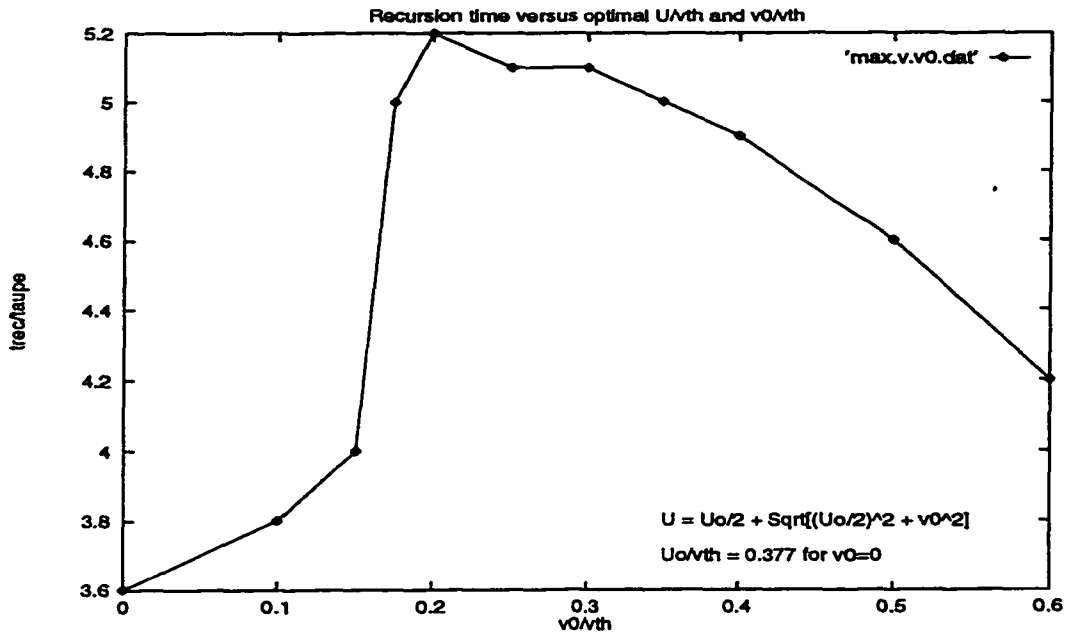

Figure 4.6: For the filtered symmetric Hermite method, variation of the velocity scale U according to Equation 4.1.11 yields improved Landau damping recursion times.

In Figure 4.6, we see that the recursion time is $5.2\tau_{pe}$ for a filter width of approximately $0.2v_{th}$ using $U_o = 0.377v_{th}$ from the optimal unfiltered symmetric Hermite simulations (see Figure 4.4). The maximum recursion time for the asymmetric Hermite simulations using $N_u = 64$ was only $3.9\tau_{pe}$; hence, the careful use of filtering allows the symmetric Hermite simulations a longer recursion time.

In the subsections below we will see how errors in modeling Landau damping with the symmetric and asymmetric Hermite methods can be reduced by proper selection of the scale factor U and filter width v_o.

4.1.3 Variation with U and v_o

Optimal spectral convergence, and therefore overall accuracy of the Hermite methods, is closely tied to proper selection of the velocity scale factor U. Errors

in calculating the eigenvalues of the linearized Vlasov-Poisson system were reduced orders of magnitude by varying the scale U in [Hol.2]. In the non-linear simulations performed here for the Vlasov-Poisson system, the same benefits can be found.

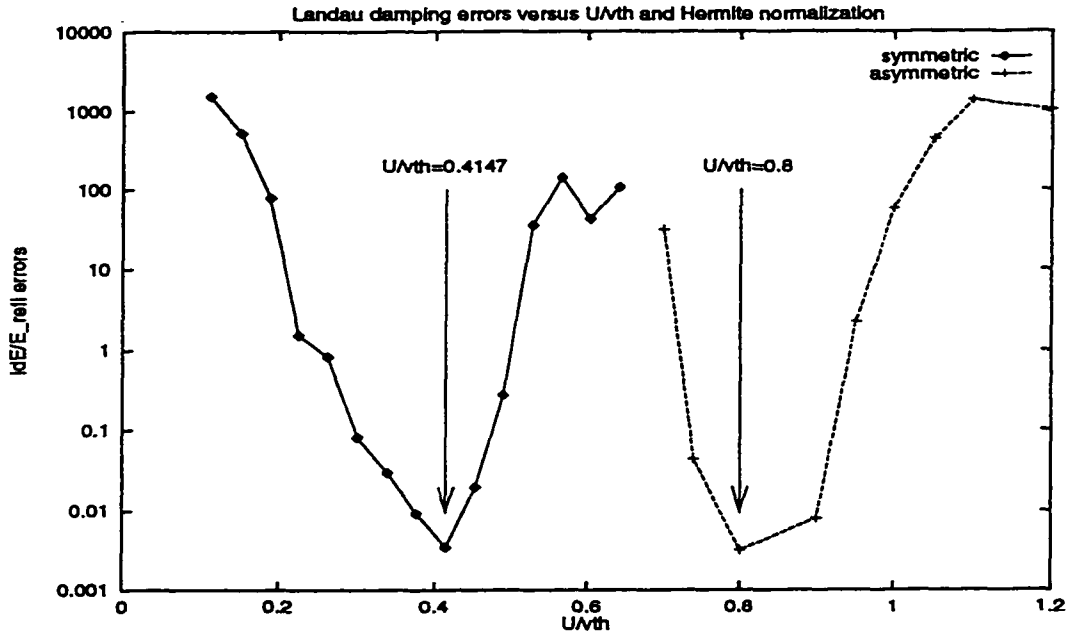

Figure 4.7: A plot of Landau damping E-field errors versus velocity scale factor U, normalized by the thermal width v_{th}.

In Figure 4.7, the errors in predicting the E-field during Landau damping are shown. The standard input deck was used, varying only the velocity scale for the tests. The test cases were compared to a reference case with a large Hermite expansion order of $N_u = 350$ and all other parameters standard. Errors were calculated by taking the difference of the stimulated E-field amplitudes from the tests and reference at a time of $t = 3\tau_{pe}$. In the figure, the symmetric Hermite method has an optimal U scale of $U/v_{th} = 0.4147$ and the asymmetric Hermite method has an optimal value of $U/v_{th} = 0.8$. Away from the optimal U value, the errors can be quite large because that test case has passed its recursion time whereas the high-resolution

reference case has not; hence, we are dividing a large (recurred) E-field value by a small (damped) reference E-field value. Nevertheless, this figure clearly shows the utility of the Hermite velocity scale.

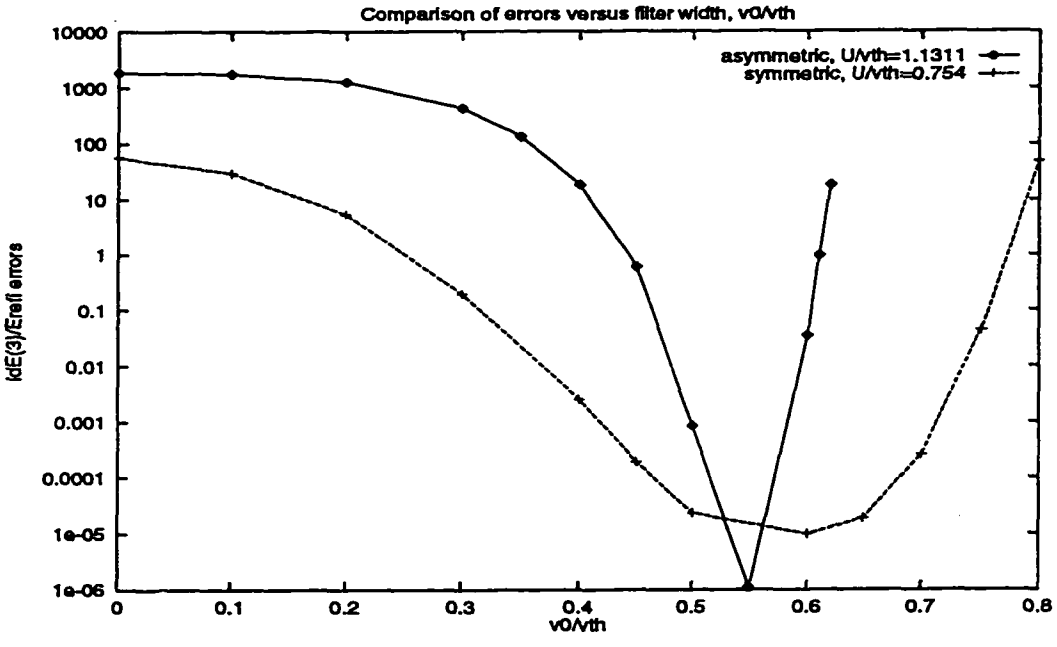

Figure 4.8: A plot of Landau damping E-field errors versus filter factor v_o normalized by the thermal width v_{th} using the two Hermite methods. Different U scales were chosen to make the optimal filter widths for the asymmetric and symmetric schemes nearly equivalent.

Similar optimization of the Landau damping results may also be obtained by variation of the filter width v_o for fixed U. The errors from a series of filtered Hermite simulations are shown in Figure 4.8, compared as before to the high-resolution unfiltered reference simulations. The filtering effectively broadens the thermal width v_{th} of the distribution to $v_{th,\text{eff}} = \sqrt{v_{th}^2 + 2v_o^2}$, thereby changing the optimal velocity scale value of $U/v_{th,\text{eff}}$. Hence, by using Equations 4.1.9 and 4.1.11 and the optimal U scales in Figure 4.7, these optimal filter widths may be predicted. For both cases, the errors are very low for filter widths ranging from $0.4v_{th}$ to $0.7v_{th}$; however, the

symmetric Hermite has a wider range of "good" filter widths because the asymmetric Hermite coefficients diverge for $v_o/v_{th} = 0.65$ in this case with $U/v_{th} = 1.1311$.

4.1.4 Dispersion relations for Landau damping

In this section, we will examine the accuracy of the two Hermite methods by comparing the Landau damping results to linear kinetic theory. The linear damping rate $\omega_i = \gamma$ and oscillation frequency ω_r for a particular perturbation with wavenumber $K = \frac{2\pi k}{L}$ are given by roots of the dispersion function [Nic.1, others],

$$\epsilon(\omega, K) \equiv 1 + \frac{e^2}{K^2 m_e \epsilon_o} \int_{-\infty}^{\infty} \frac{\partial_u f_0(u)}{u + \omega/K} du \tag{4.1.12}$$

where $f_0(u)$ is the equilibrium Maxwellian distribution given in Equation 4.1.1. This integral has a singularity on the real axis at $u = -\omega_r/K$ if $\omega_i = 0$. Analytically continuing into the negative imaginary ω plane using the Plemelj formula we have

$$\epsilon(\omega, K) \equiv \begin{cases} 1 + \frac{e^2}{K^2 m_e \epsilon_o} \left[P \int_{-\infty}^{\infty} \frac{\partial_u f_0(u)}{u+\omega/K} du + i\pi \partial_u f_o(u)\Big|_{u=-\omega_r/K} \right] & \omega_i = 0 \\ 1 + \frac{e^2}{K^2 m_e \epsilon_o} \left[\int_{-\infty}^{\infty} \frac{\partial_u f_0(u)}{u+\omega/K} du + 2\pi i \partial_u f_o(u)\Big|_{u=-\omega_r/K} \right] & \omega_i < 0 \end{cases} \tag{4.1.13}$$

where P denotes the principal value of the integral (evaluated to the left and the right of the singularity).

Using a MATHEMATICA [Wol.1] routine written by Holloway to evaluate the principal value integral and the zeros of $\epsilon(\omega, k)$ for a fixed k, we generated a dispersion relation for electrostatic waves in a Maxwellian plasma. Using the Fourier-Hermite algorithms to simulate Landau damping with various mode numbers k, we then calculated the oscillation frequencies $\omega(k)$ and damping rates $\gamma(k)$ from the peak amplitudes $|E(k,t)|$, generating the dispersion relations using the asymmetric and symmetric (both even and odd expansion order) Hermite methods shown in Figures 4.9, 4.11, and 4.13. The solid and dashed lines represent the results from analytic linear theory. For the asymmetric Hermite simulations, the velocity scale was

$U/v_{th} = 0.740$ and $v_o = 0$. The symmetric Hermite simulations were also unfiltered, using a velocity scale of $U/v_{th} = 0.377$.

Error estimates comparing these simulations to linear theory are shown in Figures 4.10, 4.12, and 4.14. In nearly all of the cases, the errors are less than 1% and are on the order of the systematic $O(\Delta t)$ sampling errors incurred in calculating the frequencies and damping rates from the output field data (about 0.4-0.7%). Because the errors were all low in these unfiltered simulations, no filtered simulations were performed. Variations in temporal resolution Δt showed qualitatively identical answers, although increasing Δt of course increased the sampling error in calculating the frequencies and damping rates. In addition, varying the spatial resolution N_x did not change the accuracy unless the stimulated mode number k could not be resolved (i.e. $N_x < 2k$). In light of this, plots showing the Landau damping simulations with variation in Δt and N_x are not given here.

To see the quality of these Fourier-Hermite results, we performed identical simulations with the Klimas Fourier-Fourier (FF) Vlasov code and generated a Landau damping dispersion relation as before. These results are shown in Figure 4.15 with the errors shown in Figure 4.16. Although the Klimas code produced qualitatively similar dispersion relations as those derived from the Fourier-Hermite simulations, the errors are much larger. In simulating Landau damping, the FH schemes are clearly superior to the Klimas FF scheme.

The large errors shown in Figure 4.16, relative to Hermite simulations with comparable work-loads, are due to the interpolation in transformed u-space (i.e. ν-space, see Appendix B). In attempting to avoid the ν-space interpolation, we encountered a few problems. First, using an integer time-increment value $\Delta\tau$ causes the E-field sampling errors to dominate the dispersion errors (sampling errors are $O(\Delta t)$). We

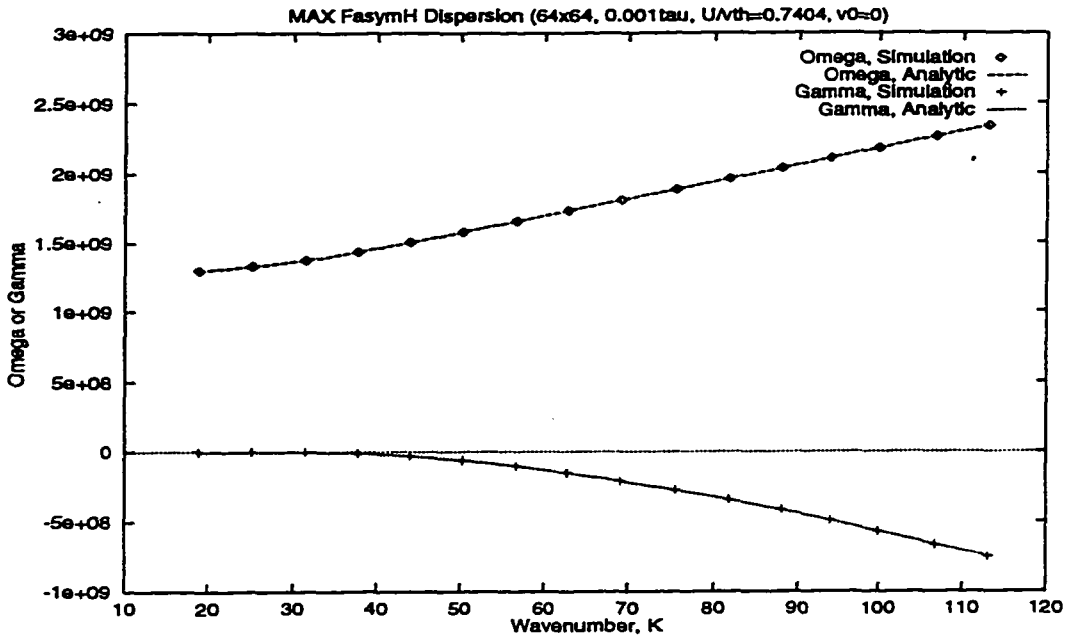

Figure 4.9: Landau damping dispersion relation from using the Fourier-asymmetric Hermite method.

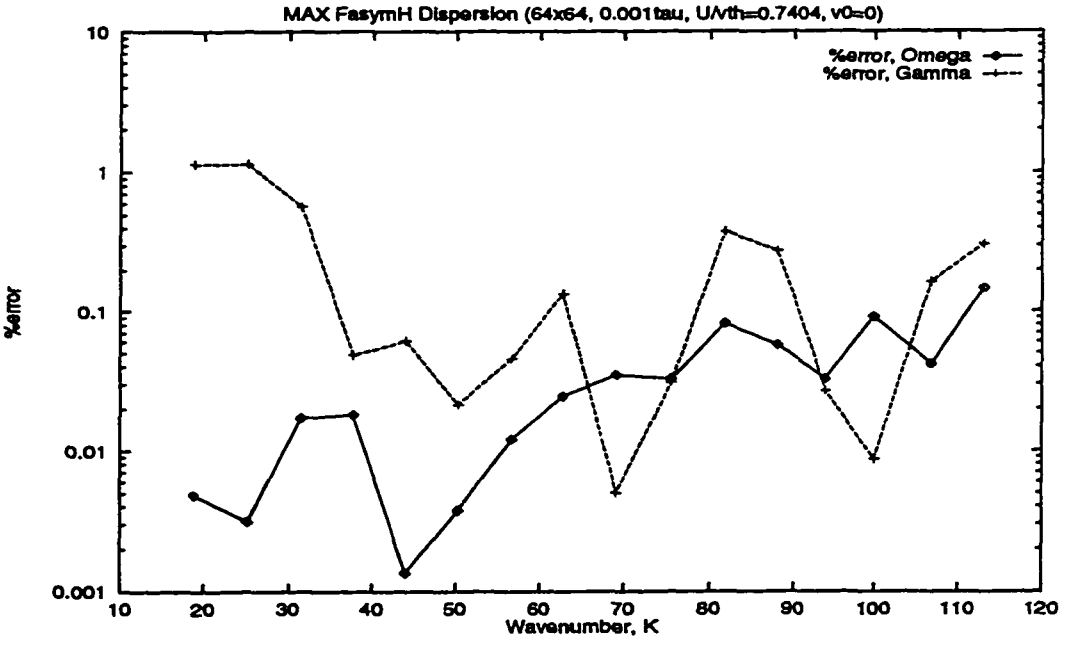

Figure 4.10: Landau damping dispersion errors from using the Fourier-asymmetric Hermite method.

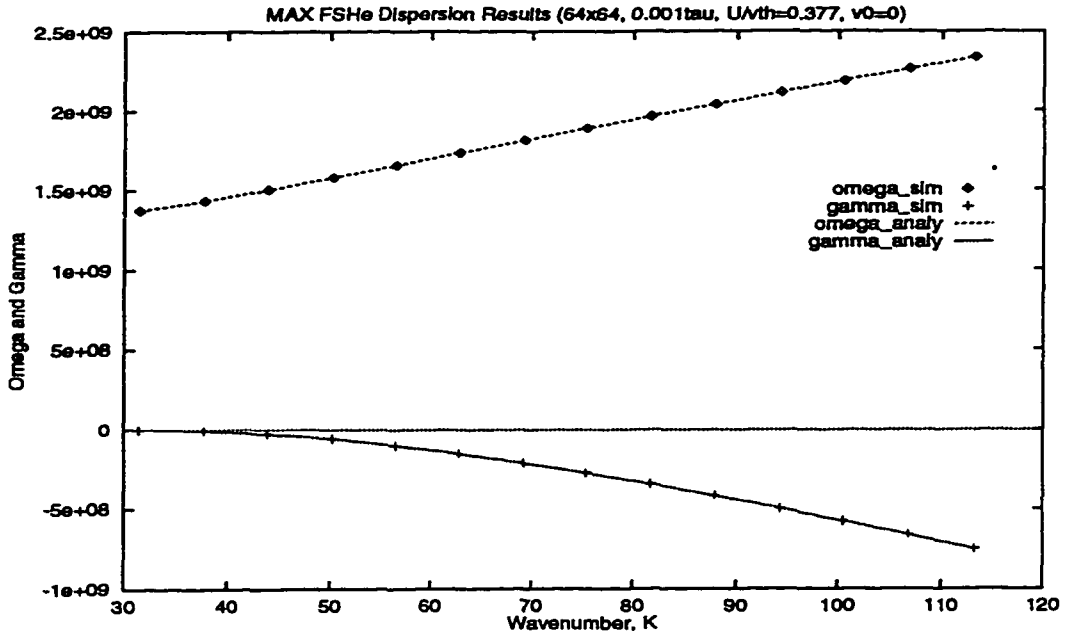

Figure 4.11: Landau damping dispersion relation from using the Fourier-symmetric Hermite (even) method.

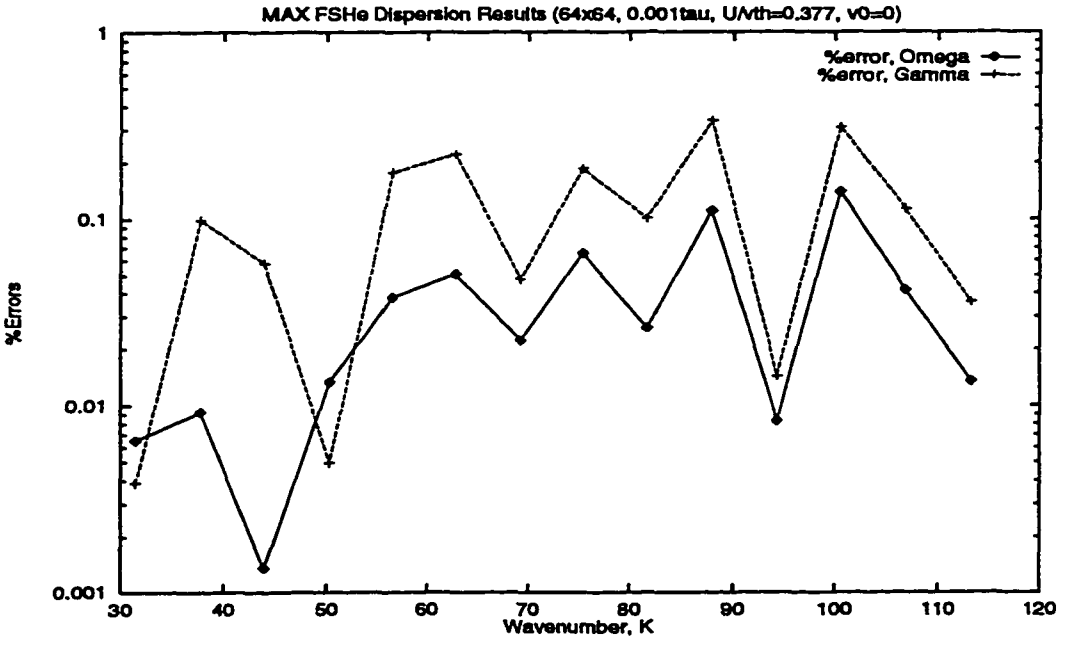

Figure 4.12: Landau damping dispersion errors from using the Fourier-symmetric Hermite (even) method.

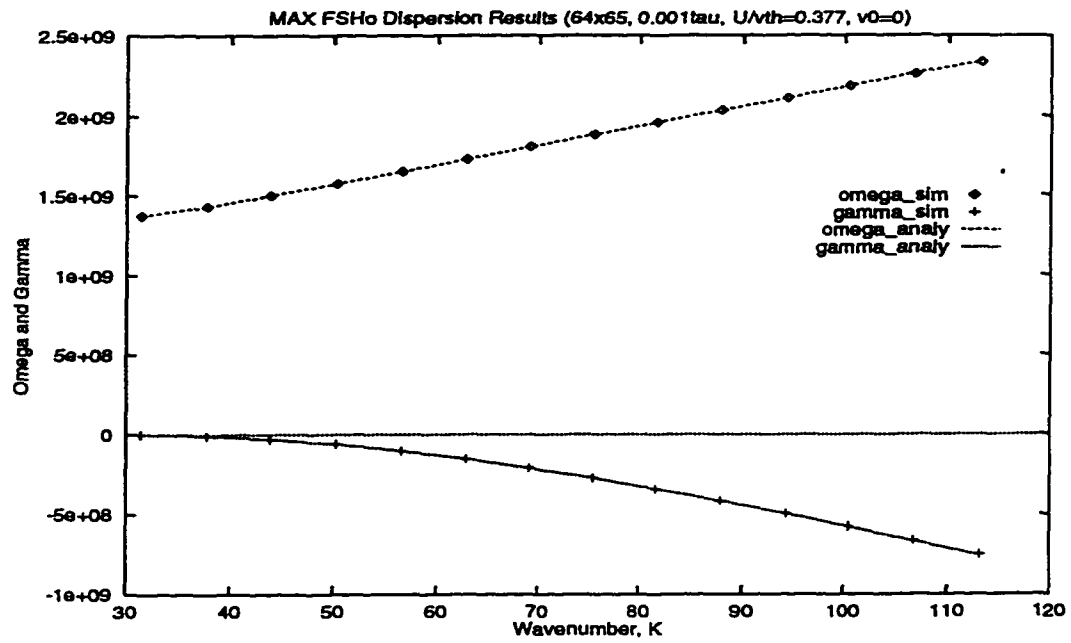

Figure 4.13: Landau damping dispersion relation from using the Fourier-symmetric Hermite (odd) method.

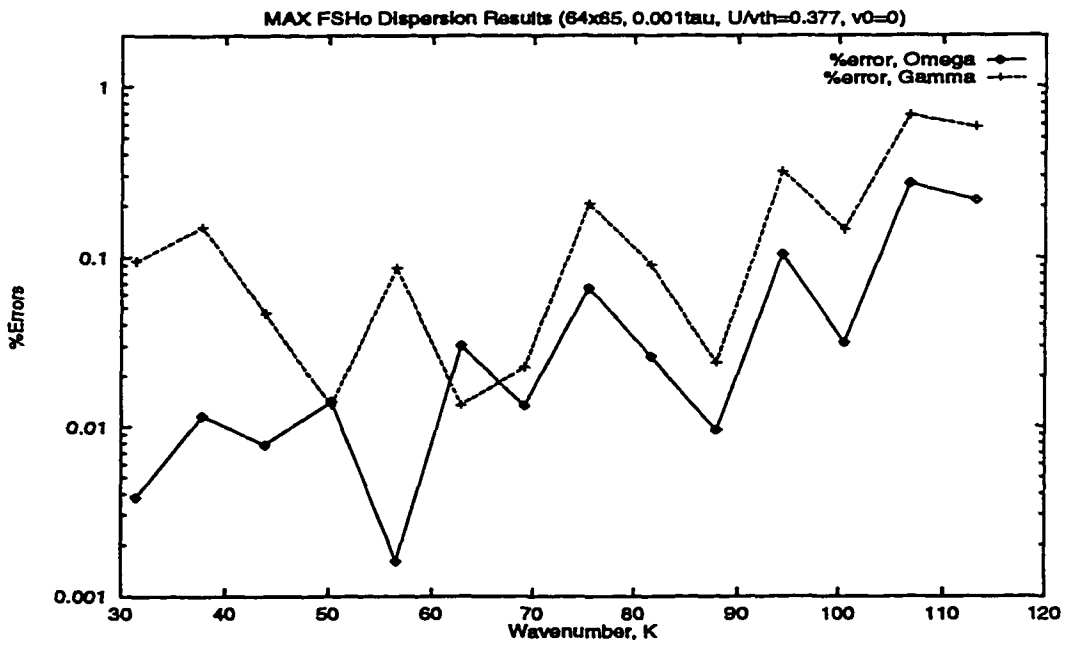

Figure 4.14: Landau damping dispersion errors from using the Fourier-symmetric Hermite (odd) method.

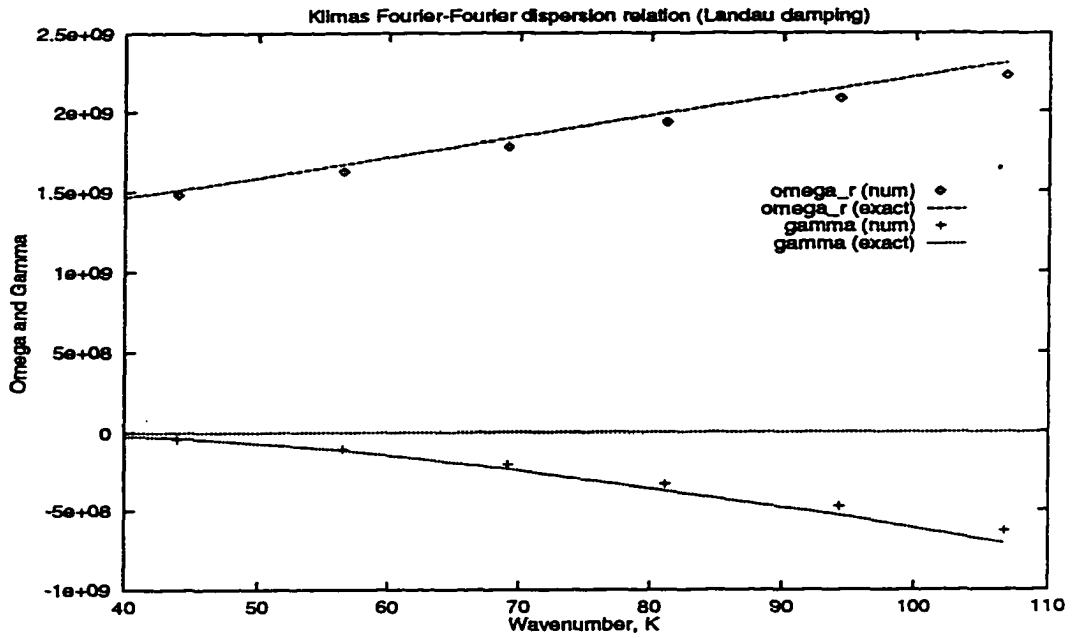

Figure 4.15: Landau damping dispersion relation using the Klimas Fourier-Fourier method.

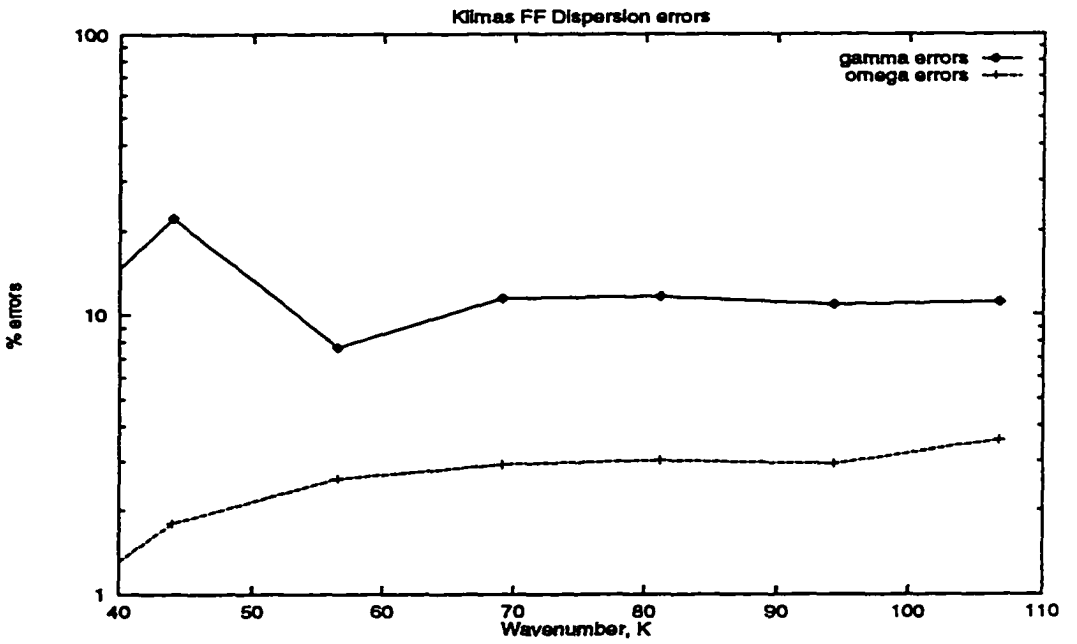

Figure 4.16: Landau damping dispersion errors using the Klimas Fourier-Fourier method.

can reduce the time-step $\Delta t = L\Delta\tau/U_{max}$ by increasing the maximum resolved velocity; however, with FF expansion orders equal to those in the FH simulations (i.e. $N_x = N_u = 64$), the velocity resolution is so poor that the E-field recurs before linear damping can begin! Therefore, to avoid ν interpolation, we are required to increase *both* the maximum resolved velocity U_{max} *and* N_u.

4.1.5 Comparisons to PIC Landau damping simulations

Another test for the Fourier-Hermite methods is a comparison to a PIC code. Using ES1 [Bir.2], a standardized and well-documented PIC code, we generated Landau damping results as shown in Figure 4.17.

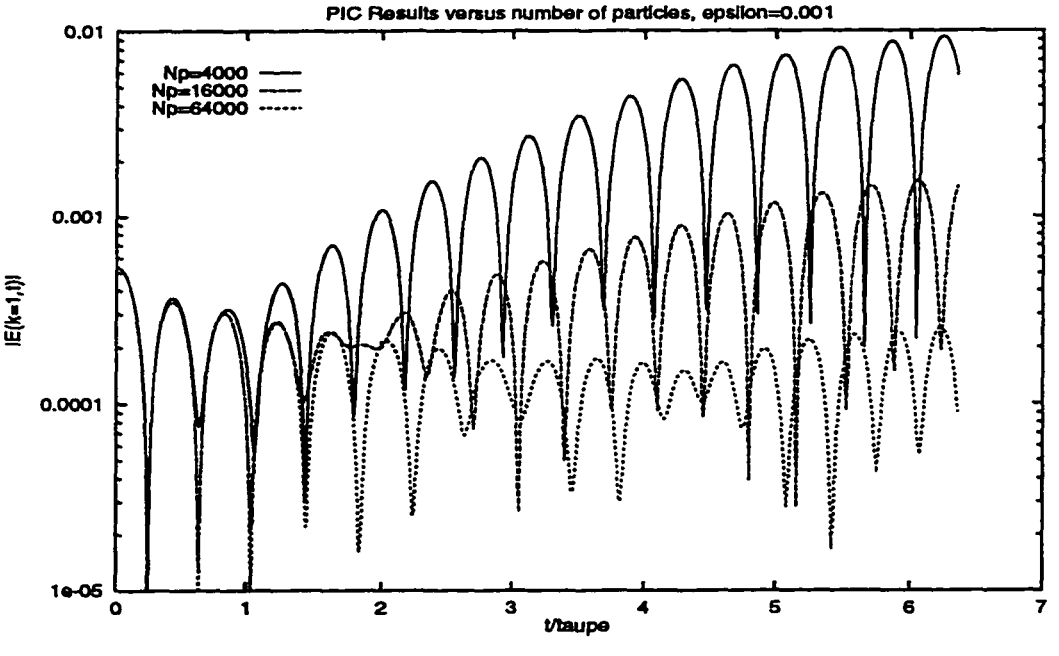

Figure 4.17: Results of PIC (ES1) simulations using the LANDAU.INP standard input versus particle number.

To give the accuracy comparisons a fair basis, we used the Maxwellian beam input "LANDAU.INP" which was provided with the ES1 code; the Fourier-Hermite simulations used similar input parameters. Note that, the input deck for the PIC/FH

Parameter	Symbol	Value used [units]
number of PIC particles	N_p	4000-64000
number density	n	1.0 [sheets/m]
thermal velocity	v_{th}	0.56568542 [m/s]
electron mass	m_e	1.0 [kg]
electron charge	e	-1.0 [C]
permittivity	ϵ_o	1.0 [F/m]
spatial resolution	N_x	64 (FH) or 128 (PIC)
Hermite velocity resolution	N_u	64
temporal resolution	Δt	7.9974 $\times 10^{-4}$ $[\tau_{pe}]$
spatial length	L	2π [m]
velocity scales (asym)	U (asym)	0.53322144 [m/s]
	U (sym)	0.26661072 [m/s]
velocity filter width	v_o (0.2 v_{th})	0.11313708 [m/s]
perturbation amplitude	ϵ	0.1 or 0.001
perturbation mode number	k	1

Table 4.2: Standard values for PIC/FH comparison simulations of Landau damping in an electron plasma with a Maxwellian velocity profile

comparisons, shown in Table 4.2, is much different than the standard Landau damping input previously shown in Table 4.1. The PIC method used a spatial resolution of 128 grid points, whereas the Fourier-Hermite methods used 64 Fourier modes in order to make the run times comparable to the PIC simulations.

Figure 4.17 shows the variation in the PIC E-field evolution versus the number of particles in the system (ranging from 4000 to 64000 macroparticles). There appears to be a recursion of the E-field in the PIC simulations; actually, this is a two-beam instability due to the inability of PIC codes to load an initial Maxwellian velocity profile. In PIC simulations, a Maxwellian velocity distribution is often approximated by initially prescribing several monoenergetic beams of macroparticles; each beamlet, having a specific velocity and density, contributes to the initial velocity profile in a way that approximates a Maxwellian (called a "cold start"). Macroparticle velocities can be then randomized about their mean beamlet velocity in order to reduce this

Method	$\omega + i\gamma$ ($\epsilon = 0.1$)	$\omega + i\gamma$ ($\epsilon = 0.001$)	percent error
linear theory	–	1.2488456 - i 0.0471321	–
PIC, N_p=4000	1.253 - i 0.0549	–	–
PIC, N_p=8000	1.251 - i 0.0569	1.303 - i 0.0102	4.40 + i 78.3%
PIC, N_p=16000	1.250 - i 0.0574	1.267 - i 0.0495	1.50 + i 5.10%
PIC, N_p=32000	1.250 - i 0.0577	1.247 - i 0.0481	0.17 + i 2.00%
Fourier-Hermite	1.252 - i 0.0583	1.249 - i 0.0474	0.03 + i 0.67%

Table 4.3: Comparison of the Landau damping dispersion errors between a PIC code and the Hermite methods. Sampling errors are approximately 0.26% for all cases.

instability by filling in the phase-space holes (called a "warm start"). However, if the total particle number is small, then the tenuous regions between the beamlets are not filled very well, leading to the familiar two-beam instability [Pen.1].

The comparisons of the PIC and the Fourier-Hermite simulations are tabulated in Table 4.1.5. Asymmetric, even symmetric, and odd symmetric Hermite simulations yielded nearly identical results, so only the Fourier-even symmetric Hermite results are shown. For the non-linear ($\epsilon = 0.1$) PIC and FH simulations, the frequencies ω are identical to within the sampling error; the PIC damping rates appear to be converging to the Fourier-Hermite damping rate of $\gamma = -0.0583$ 1/sec. However, without an analytic solution to base comparisons, it is difficult to say which numerical method is more accurate.

For a more conclusive comparison, Table 4.1.5 also shows comparisons to linear theory (the perturbation parameter ϵ was lowered to 0.001 in order to simulate Landau damping in the linear regime). The Fourier-Hermite method using 4096 unknowns yielded results more accurate than a 32000 particle PIC simulation! In fact, the 4000 particle PIC simulation could not exhibit Landau damping with this small perturbation parameter of $\epsilon = 0.001$ due to the two-beam instability; to see

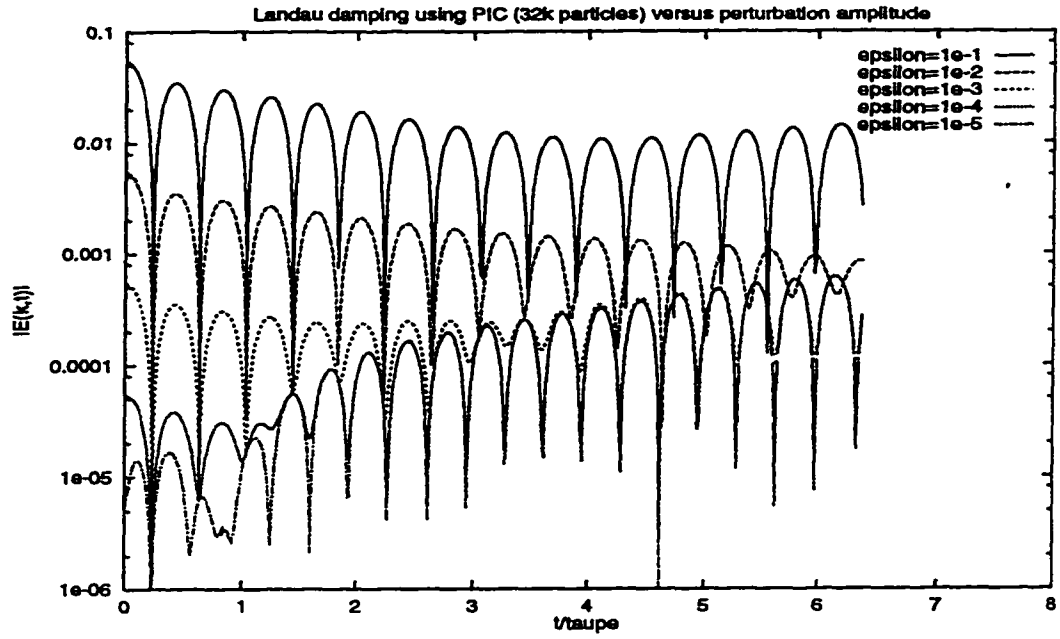

Figure 4.18: For a constant number of particles, long-time Landau damping PIC (ES1) simulations are limited by a two-beam instability. The instability is important for small amplitude perturbations.

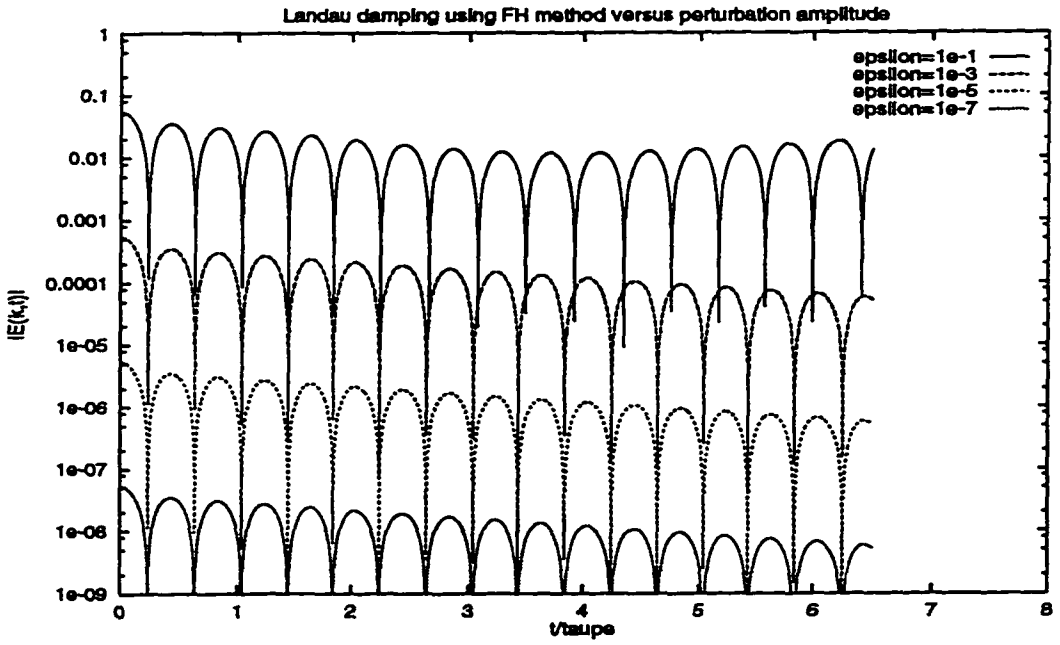

Figure 4.19: FH simulations do not require additional numbers of unknowns in order to simulate Landau damping initiated by small perturbations; FH methods are limited by recursion only (which is not shown here).

long-time Landau damping, the required particle number in PIC simulations must increase as the perturbation parameter ϵ decreases.

The Fourier-Hermite method, which is limited by recursion but not by the two-beam instability, can exhibit long-time Landau damping for a wider range of ϵ (see Figures 4.18 and 4.19). For this reason, a spectral kinetic method is superior to PIC for the study of warm plasma dynamics using any perturbation amplitude ϵ.

4.2 Bump-On-Tail Simulations

In this section, the initial distribution to be used as input is a bump-on-tail (BOT) velocity profile. Simply put, it is a Maxwellian plus a high-energy beam in velocity space, written

$$f(x,u,t) = \frac{g_p(x,0)}{\sqrt{\pi}v_{th,p}} \exp\left[-\frac{u^2}{v_{th,p}^2}\right] + \frac{g_b(x,0)}{\sqrt{\pi}v_{th,b}} \exp\left[-\frac{(u-v_{d,b})^2}{v_{th,b}^2}\right] \qquad (4.2.1)$$

where the main "plasma" distribution is defined by the arbitrary spatial function $g_p(x,0)$, thermal velocity $v_{th,p}$, and the drift velocity $v_{d,b}$. The "bump" distribution is defined by the similarly named but "b" subscripted coefficients. In these simulations, we will again analyze a background-neutralized electron plasma with its physically relevent quantities defined in Table 4.4. A sample BOT profile is shown in Figure 4.20.

The initial spatial dependencies $g_p(x,0)$ and $g_b(x,0)$ will again have cosinusoidal forms defined by

$$g_p(x,0) = \mathrm{n}_p\left[1+\epsilon\cos\left(\frac{2\pi kx}{L}\right)\right] \qquad (4.2.2)$$

$$g_b(x,0) = \frac{\mathrm{n}_b}{\mathrm{n}_p} g_p(x,0) \qquad (4.2.3)$$

where n_p is the number of bulk plasma particles, n_b is the number of beam plasma particles, ϵ is the perturbation amplitude, k is the mode number to stimulated, and

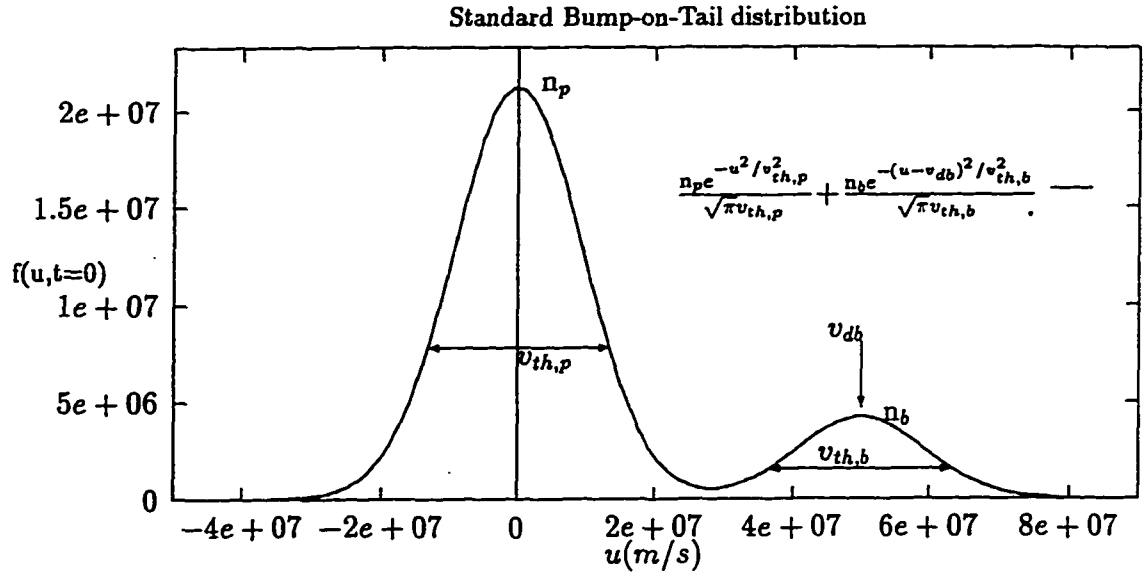

Figure 4.20: A bump-on-tail (BOT) velocity profile.

L is the system length. The perturbations $g_p(x,0)$ and $g_b(x,0)$ were chosen to be similar in order to simplify comparisons to linear theory. Again, for simplicity, only one mode k will be stimulated in any simulation.

The standard input deck is shown in Table 4.4. The number densities, thermal widths, electron mass, and charge were fixed for all simulations using the BOT distribution. For numerical testing, the spatial resolution N_x (the number of x grid points or Fourier modes), the velocity resolution N_u (the number of u grid points or Hermite modes), the time step Δt, the Hermite scale factor U, and the filter width v_o will be varied, allowing comparisons for accuracy against linear theory. The stimulated mode k and the perturbation amplitude ϵ will be varied and used to generate dispersion relations to compare to linearized theory for unstable plasma configurations.

Figure 4.21 shows the initial BOT velocity profile and the profile after saturation at $t = 10\tau_{pe}$ (plots of $f(u,t) = \int f(x,u,t)dx$). Figure 4.22 shows the evolution of

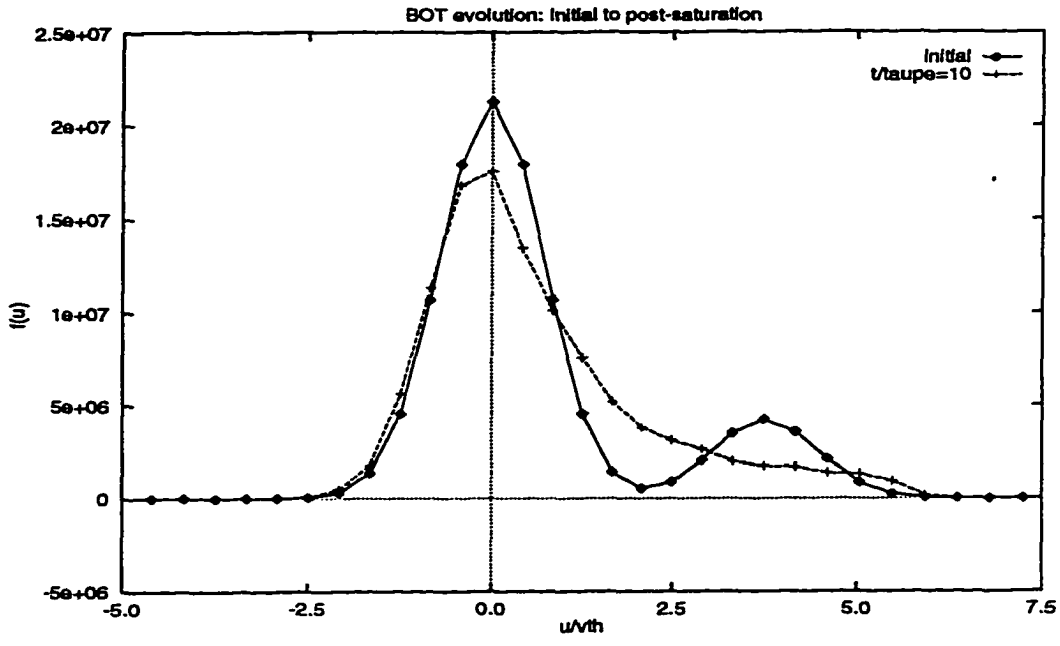

Figure 4.21: Bump-on-tail profiles, initially and after 10 plasma periods of simulation using the Fourier-Hermite methods. The points represent the value of the distribution on the discretized velocity mesh.

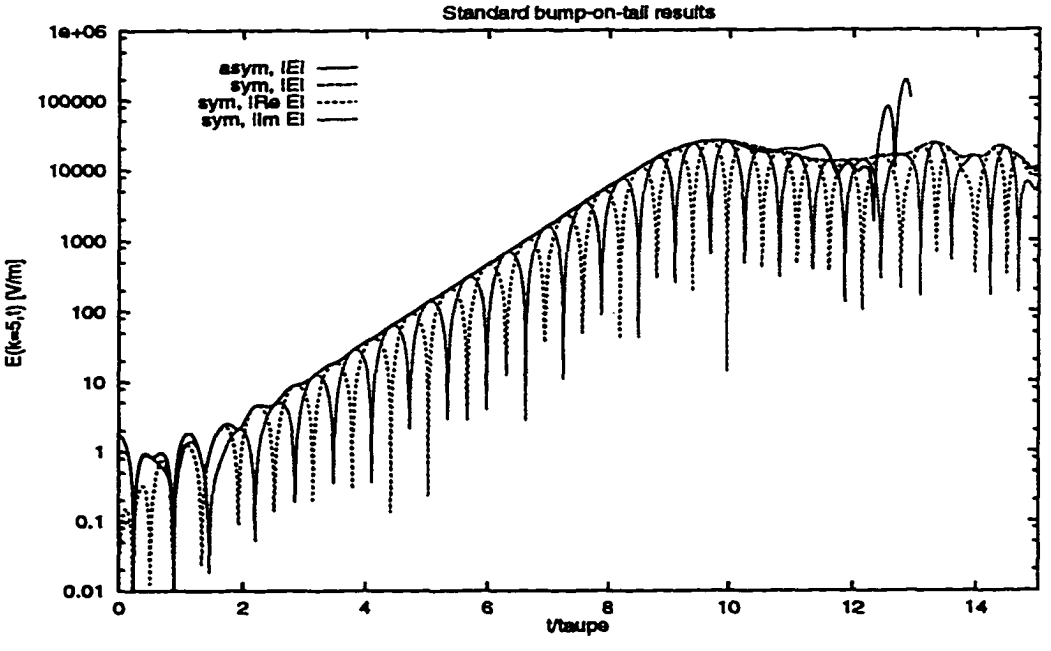

Figure 4.22: Standard results of bump-on-tail simulations using the Fourier-Hermite methods. The dispersion frequency and growth rate will be calculated from various peaks as shown.

Parameter	Symbol	Value used [units]
number densities	n_p	5×10^{14} [sheets/m]
	n_b	1×10^{14} [sheets/m]
thermal velocities	$v_{th,p}$	1.32619×10^{7} [m/s]
	$v_{th,b}$	1.32619×10^{7} [m/s]
drift velocities	$v_{d,p}$	0.0 [m/s]
	$v_{d,b}$	5.0×10^{7} [m/s]
electron mass	m_e	$9.1095e \times 10^{-31}$ [kg]
electron charge	e	-1.60219×10^{-19} [C]
permittivity	ϵ_o	$8.8541878 \times 10^{-12}$ [F/m]
spatial resolution	N_x	64
velocity resolution	N_u	64 or 65
temporal resolution	Δt	0.001 τ_{pe} [s]
spatial length	L	1 [m]
velocity scale	U	2.0×10^{7} [m/s]
velocity filter width	v_o	0.0 [m/s]
perturbation amplitude	ϵ	10^{-5}
perturbation wavenumber	k	5

Table 4.4: Standard values for simulation of growing modes in an electron plasma with a BOT velocity profile

the stimulated $|E(k=5,t)|$ mode during bump-on-tail simulations using the asymmetric and symmetric Hermite methods. The absolute value of real and imaginary components, $|\Re[E(k=5,t)]|$ and $|\Im[E(k=5,t)]|$, are also shown. Growth rates and dispersion frequencies will be obtained from the real-valued peaks of these individual components. In the level-set contours (i.e. plots of $\ln[f(x,u,t)]$ in (x,u) phase-space at various times) shown in Figures 4.23, 4.24, 4.25, and 4.26, we see the formation of trapped and untrapped particle regions as the E-field grows. Until saturation of the E-field at $t \approx 10\tau_{pe}$, the trapped particle regions advect with speed $v = v_{phase} \approx \omega/k$.

It is important to notice the apparent instability at the end of the asymmetric Hermite simulation. This numerical instability was not noticed during any of the Landau damping simulations and is the greatest weakness of the asymmetric Hermite method. After extensive numbers of simulations, varying all possible parameters, we

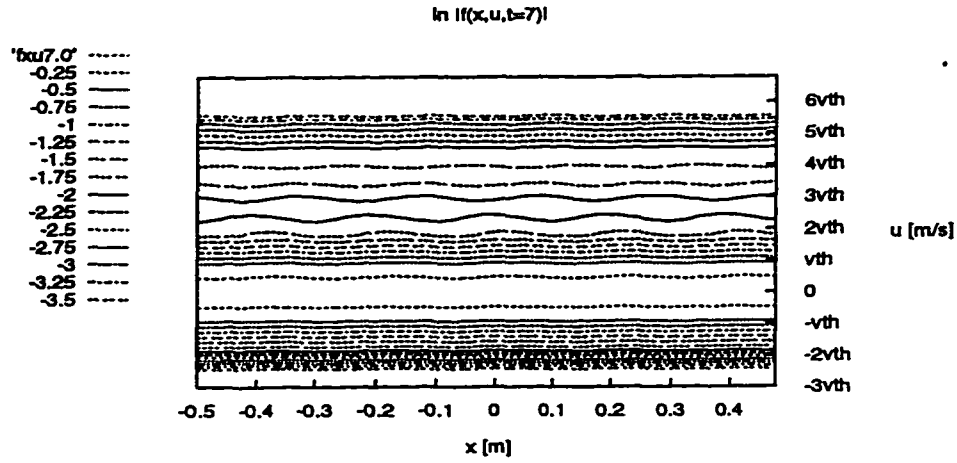

Figure 4.23: Level-set contours of $\ln[f(x,u,t)]$ in (x,u) phase-space at time $t = 7.0\tau_{pe}$. We begin to see particle interactions with the E-field at $u = v_{phase}$.

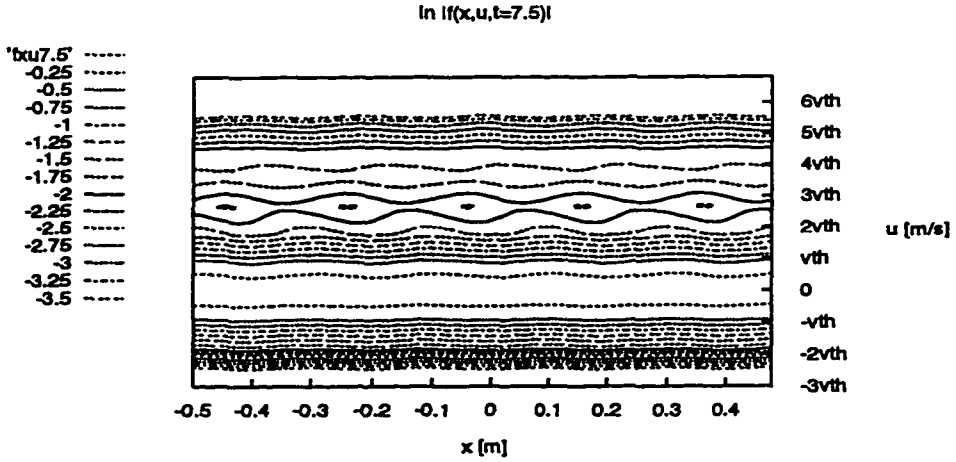

Figure 4.24: Level-set contours of $\ln[f(x,u,t)]$ in (x,u) phase-space at time $t = 7.5\tau_{pe}$. Trapped particle regions for modenumber $m = 5$ are beginning to form.

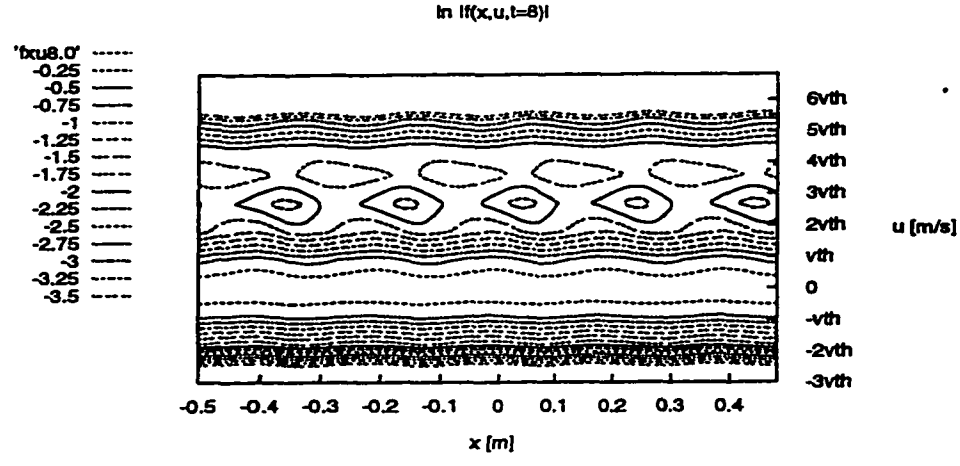

Figure 4.25: Level-set contours of $\ln[f(x,u,t)]$ in (x,u) phase-space at time $t = 8.0\tau_{pe}$. Trapped particle regions are easily identified by the "cat's eye" formations.

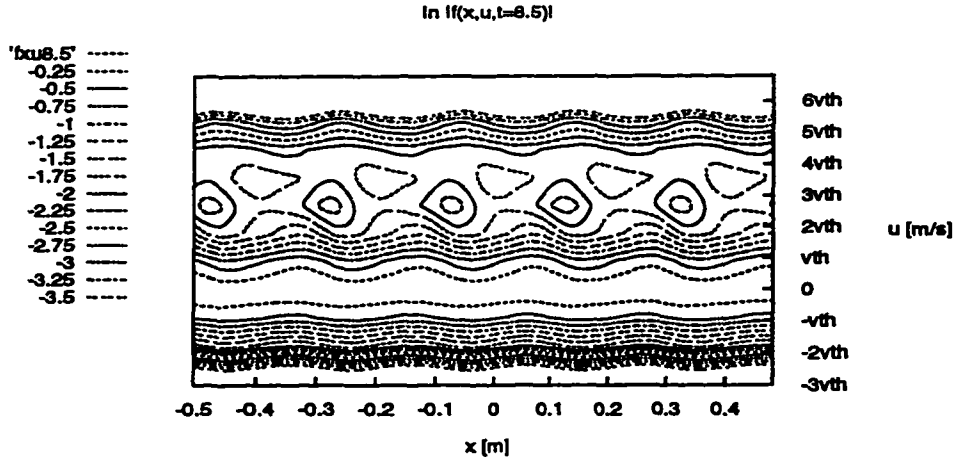

Figure 4.26: Level-set contours of $\ln[f(x,u,t)]$ in (x,u) phase-space at time $t = 8.5\tau_{pe}$. The "cat's eye" formations are well-developed and moving at the phase-speed velocity.

found that this instability could not be avoided with the asymmetric Hermite scheme. The symmetric Hermites, on the other hand, were always stable.

We know that the Vlasov equation, derived from Liouville's equation, conserves all of the quantities $\iint f^p(x,u,t)\,dxdu$ for well-behaved distribution functions f that go to zero at infinity. However, in simulations using the Vlasov equation, the integral $\iint f dxdu$ may be constant even if f itself is wildly oscillatory in space or velocity. Stability of an algorithm is therefore related to the ability to conserve the integral of a positive definite quantity such as $\iint f^2 dxdu$ during simulations. In the next subsection, we derive relations for calculating $\iint f^2 dxdu$ and empirically show that the asymmetric Hermites do not conserve this integral.

4.2.1 Evaluation of $\iint f^2 dxdu$

Conservation of the integral of f^2 is an important numerical stability property for a kinetic algorithm. Weighted-residual methods based on symmetric sets of basis functions naturally conserve this quantity, but, those based on non-self-adjoint basis functions do not [Hol.3]. In this section, we empirically show that the asymmetric Hermite method derived in this dissertation does not conserve the integral of f^2 and is, therefore, unstable. We also show that although the unfiltered symmetric Hermite method conserves this integral, the filtered symmetric Hermite method does not. Fortunately, the integral of $\bar{f}^2$ is bounded for the filtered symmetric Hermite method, implying stability.

The integral of f^2 over all of phase space is written

$$I_2 \equiv \iint [f]^2\,dxdu = \sum_m \sum_n f^{mn}(t) \iint f(x,u,t)\Phi_m(x)\Psi_n(v)\,dxdu \qquad (4.2.4)$$

where $v = u/U$.

For the Fourier-asymmetric Hermite method, we have $\Psi_n = e^{-\lambda^2}\Psi^n$ and $\Phi_m =$

Φ^{-m}, so we find

$$I_2 = LU \sum_{m=-\frac{N_x}{2}}^{\frac{N_x}{2}-1} \sum_{n=0}^{N_u} f^{mn}(t) \left[\sum_{n'=0}^{N_u} f^{-m,n'}(t) C^{n,n'} \right] \tag{4.2.5}$$

where the coefficients $C^{n,n'} = \int_{-\infty}^{\infty} e^{-2\lambda^2} \Psi^n \Psi^{n'} d\lambda$ can be evaluated to be

$$C^{n,n'} = \begin{cases} \frac{(-1)^{\frac{n+n'}{2}}(n+n'-1)!!}{\sqrt{2^{n+n'+1} n!(n')!}} & n+n' \text{ even} \\ 0 & n+n' \text{ odd} \end{cases} . \tag{4.2.6}$$

These may be found by a simple recursion relation using $C^{00} = \sqrt{2}/2$ and $C^{11} = -\sqrt{2}/2$, written

$$C^{n,0} = -\sqrt{\frac{n-1}{n}} C^{n-2,0} , \; n \text{ even} \tag{4.2.7}$$

$$C^{n,1} = -\sqrt{\frac{n}{n-1}} C^{n-2,1} , \; n \text{ odd} \tag{4.2.8}$$

$$C^{n,n'} = -\frac{n+n'-1}{\sqrt{n'(n'-1)}} C^{n,n'-2} . \tag{4.2.9}$$

For the Fourier-symmetric Hermite method, we find a much simpler form for I_2. Recalling that $\Psi_n = \Psi^n$ and $\Phi_m = \Phi^{-m}$, we get the relation

$$I_2 = LU \sum_{m=-\frac{N_x}{2}}^{\frac{N_x}{2}-1} \sum_{n=0}^{N_u} |f^{mn}(t)|^2 . \tag{4.2.10}$$

Using Equations 4.2.5 and 4.2.10 during the simulations, we evaluated the actual change in the integral of f^2 for the asymmetric, the unfiltered symmetric, and the filtered symmetric Hermite methods. The change in I_2 for these three methods is shown in Figure 4.27. The asymmetric Hermite method shows large changes in I_2 with time (over 100% by $t = 1.5\tau_{pc}$), whereas changes in the unfiltered symmetric Hermite method remain at round-off level (about 10^{-15} in double precision) for nearly the entire simulation. At the end of the run, splitting errors begin to dominate the symmetric simulations. Filtered symmetric Hermite simulations do not exhibit perfect conservation of I_2, but the errors remain less than 0.1% for this example.

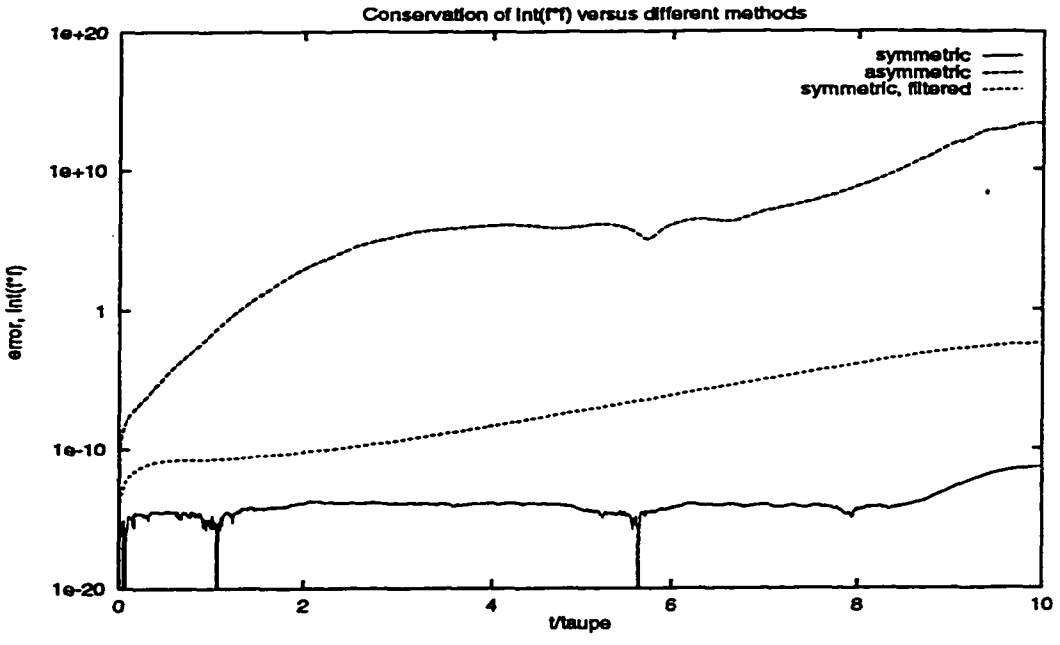

Figure 4.27: Errors in conservation of the integral of f^2 for the asymmetric, symmetric, and filtered symmetric Hermite methods. Symmetric Hermite has the best conservation properties, but eventually the splitting errors begin to dominate at the end of the simulation shown.

In conclusion, the lack of conservation of $\iint f^2 dx du$ during asymmetric Hermite simulations is the cause of the numerical instability seen in Figure 4.22. In order to perform long-time BOT simulations, we found that increasing velocity resolution was the only medicine; however, nothing could cure the disease.

4.2.2 Dispersion relations for bump-on-tail

In this section, we will examine the accuracy of the two Hermite methods by comparing the BOT electrostatic growth rates to the predictions of linear kinetic theory. The linear growth rate $\omega_i = \gamma$ and oscillation frequency ω_r for a particular perturbation with wavenumber $K = \frac{2\pi k}{L}$ are again given by roots of the dispersion function in Equation 4.1.12 where $f_0(u)$ is now the equilibrium BOT distribution given in Equation 4.2.1. The dispersion function does not have a singularity for

$\omega_i > 0$ (growing modes), so we may simply evaluate the integral

$$\epsilon(\omega, K) \equiv 1 + \frac{e^2}{K^2 m_e \epsilon_o} \left[\int_{-\infty}^{\infty} \frac{\partial_u f_0(u)}{u + \omega/K} du \right] , \quad \omega_i > 0 . \tag{4.2.11}$$

Again using the MATHEMATICA [Wol.1] routine written by Holloway to evaluate the zeros of $\epsilon(\omega, k)$ for a fixed k, we generated a dispersion relation for growing electrostatic waves in a BOT plasma. Using the Fourier-Hermite algorithms to evolve the perturbed BOT distribution with various wave numbers k, we then calculated the oscillation frequencies $\omega(k)$ and damping rates $\gamma(k)$ from the peaks in amplitude of $|\Re[E(k,t)]|$ and $|\Im[E(k,t)]|$, generating the dispersion relations for the asymmetric and symmetric (both even and odd expansion orders) Hermite methods shown in Figures 4.28, 4.30, and 4.32. The solid and dashed lines represent the results from linear theory.

Error estimates comparing these simulations to linear theory are shown in Figures 4.29, 4.31, and 4.33. In nearly all of the low mode number cases, the errors are less than 1% and are on the order of the systematic $O(\Delta t)$ sampling errors in calculating the frequencies and damping rates (about 0.4-0.7%). In test cases with high spatial mode numbers (phase velocities corresponding to the low-density region, see Figure 4.21), the dispersion errors can be reduced by decreasing the velocity scale U or increasing the filter width v_o (see Subsection 4.2.3.

Variations in temporal resolution Δt showed qualitatively identical answers, although increasing Δt increased the sampling error in calculating the frequencies and damping rates. In addition, varying the spatial resolution N_x did not change the accuracy unless the stimulated mode number k could not be resolved (i.e. $N_x < 2k$). In Subsection 4.2.3, we will see how these errors can be reduced by increasing the velocity resolution or by using optimal velocity scaling and filtering. For brevity,

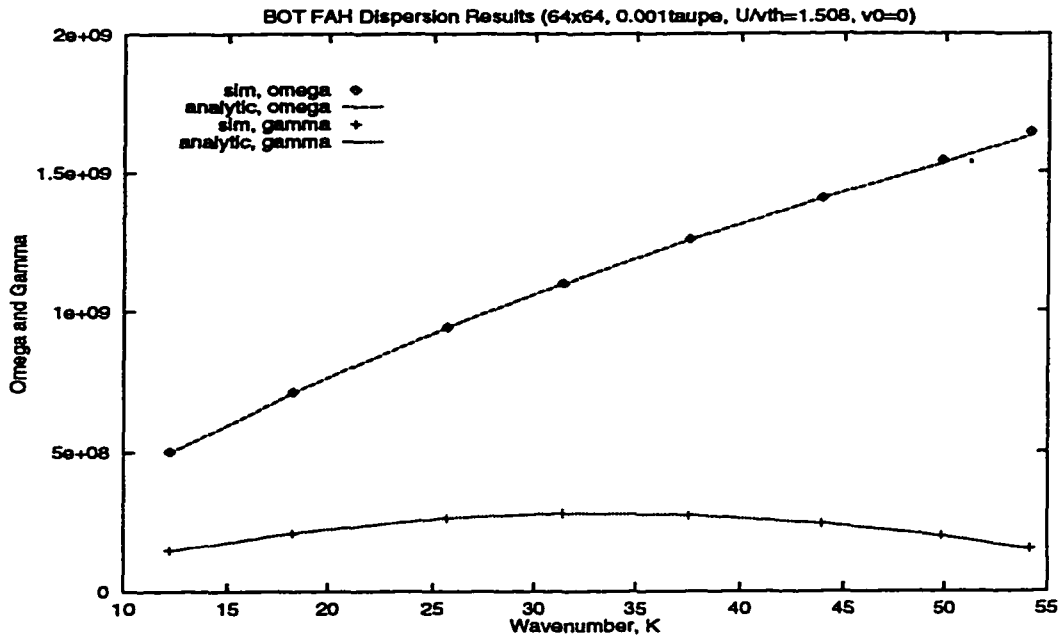

Figure 4.28: Bump-on-tail plasma dispersion relation from using the Fourier-asymmmetric Hermite method.

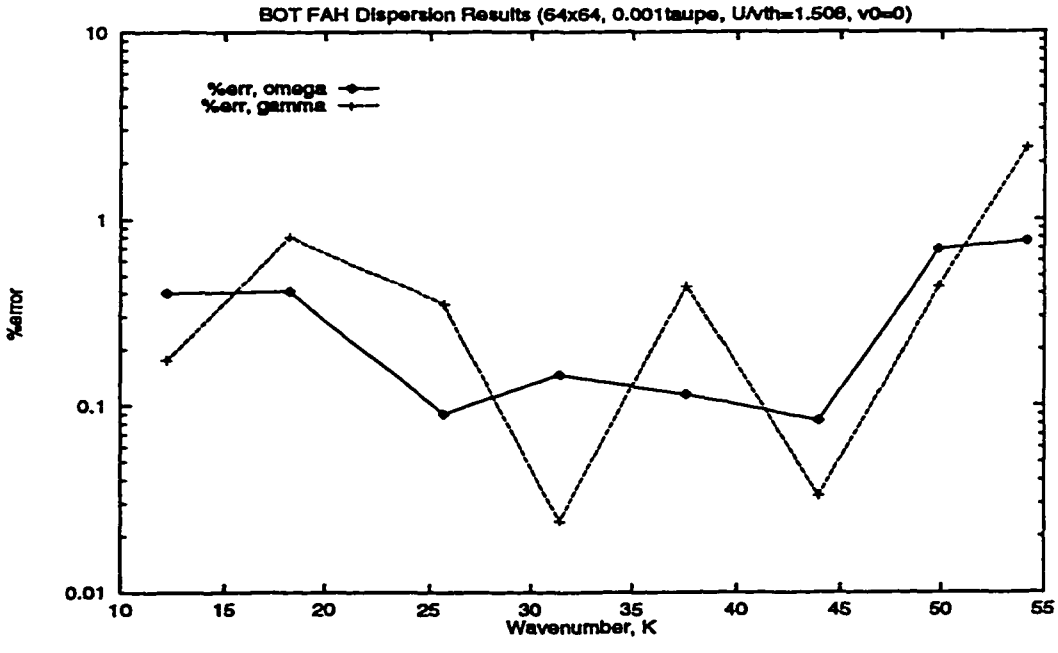

Figure 4.29: Bump-on-tail dispersion errors from using the asymmetric Hermite method.

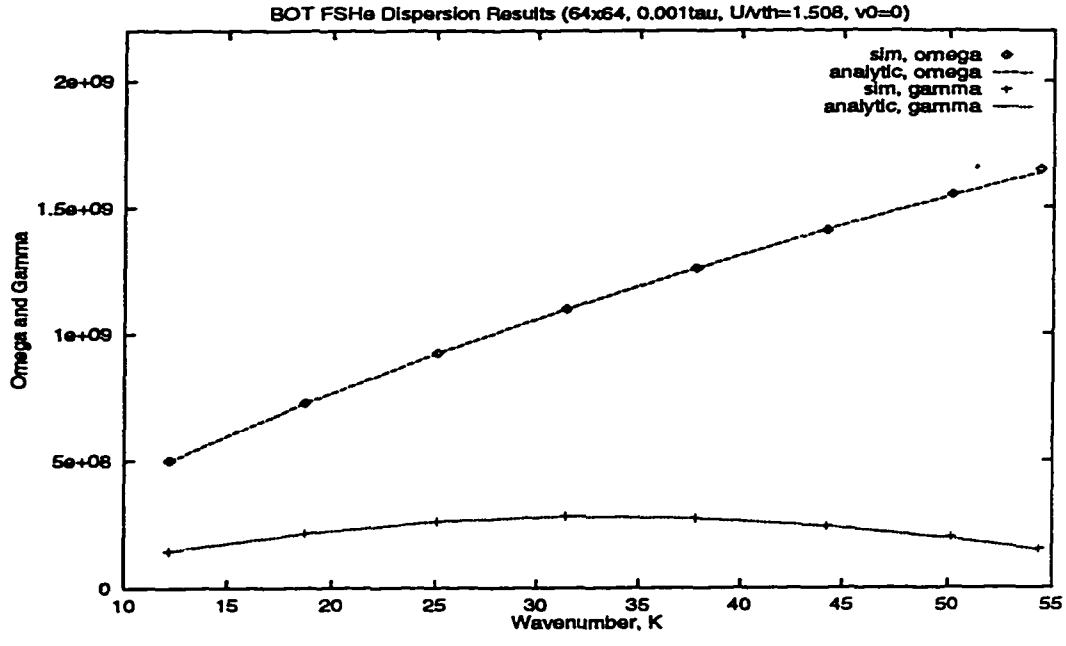

Figure 4.30: Bump-on-tail plasma dispersion relation from using the Fourier-symmetric Hermite (even expansion order) method.

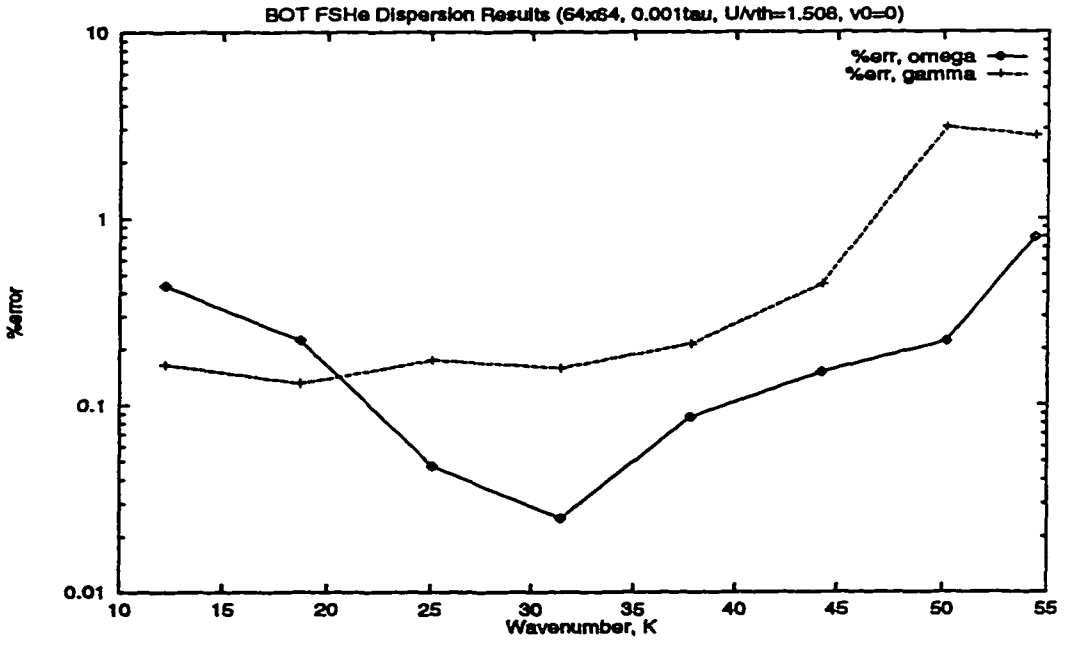

Figure 4.31: Bump-on-tail dispersion errors from using the Fourier-symmetric Hermite (even expansion order) method.

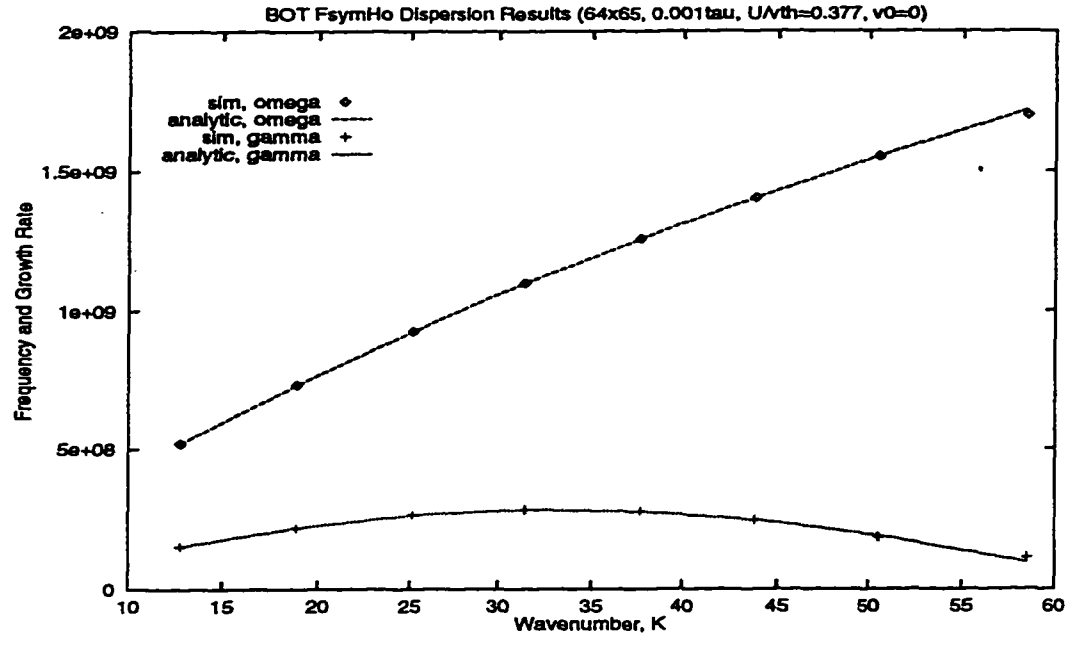

Figure 4.32: Bump-on-tail plasma dispersion relation from using the Fourier-symmetric Hermite (odd expansion order) method.

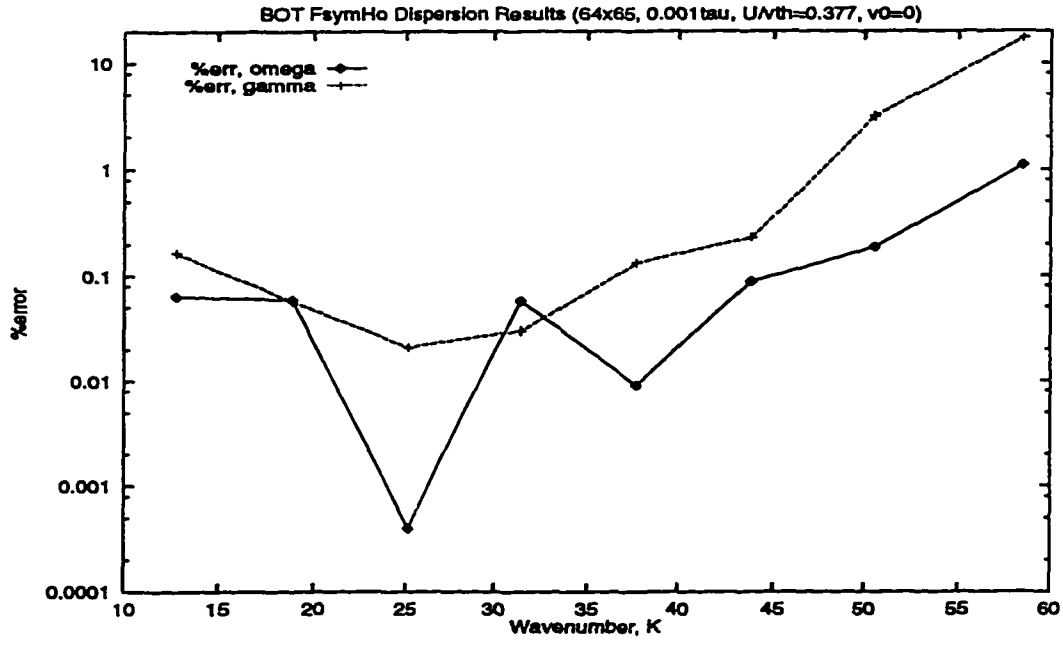

Figure 4.33: Bump-on-tail dispersion errors from using the Fourier-symmetric Hermite (odd expansion order) method.

only results from the even symmetric Hermite simulations will be shown.

4.2.3 Variation with N_u, U, and v_o

In any computer simulation, we expect the errors to decrease with increasing resolution. In Subsection 4.1.1, we saw that the time to recursion was lengthened by increasing the Hermite expansion order N_u. In Figure 4.34, we see that dispersion errors may be decreased by the same resolution increase. These results are from BOT simulations using the symmetric Hermite method with standard input parameters. As N_u increases, the errors fall below the sampling error of 0.32%.

We may also decrease the dispersion errors by choosing an optimal velocity scale U. Figure 4.35 shows error estimates from unfiltered simulations using the even symmetric Hermite method with a variable Hermite velocity scale. We find the best results for $U/v_{th} \approx 0.55$ (the standard normalized velocity scale was $U/v_{th} = 1.51$ in Table 4.4). In these simulations, we stimulated the mode number $k = 9$ since the errors in the standard $k = 5$ simulations were all below the sampling error of 0.32%.

Note once again that an optimal choice of Hermite velocity scale U can reduce errors by several orders of magnitude below errors from simulations with a non-optimal choice.

As a last example, we see in Figure 4.36 that dispersion errors can be reduced by filtering with $v_0/v_{th} = 0.3$ and $U/v_{th} = 1.51$ using the even symmetric Hermites. In these simulations, the standard input parameters were used to again simulate mode number $k = 9$. Optimal filtering can produce the same quality of solutions as an optimal choice of Hermite scale U. However, filtering reduces the truncation errors: hence, it should be used.

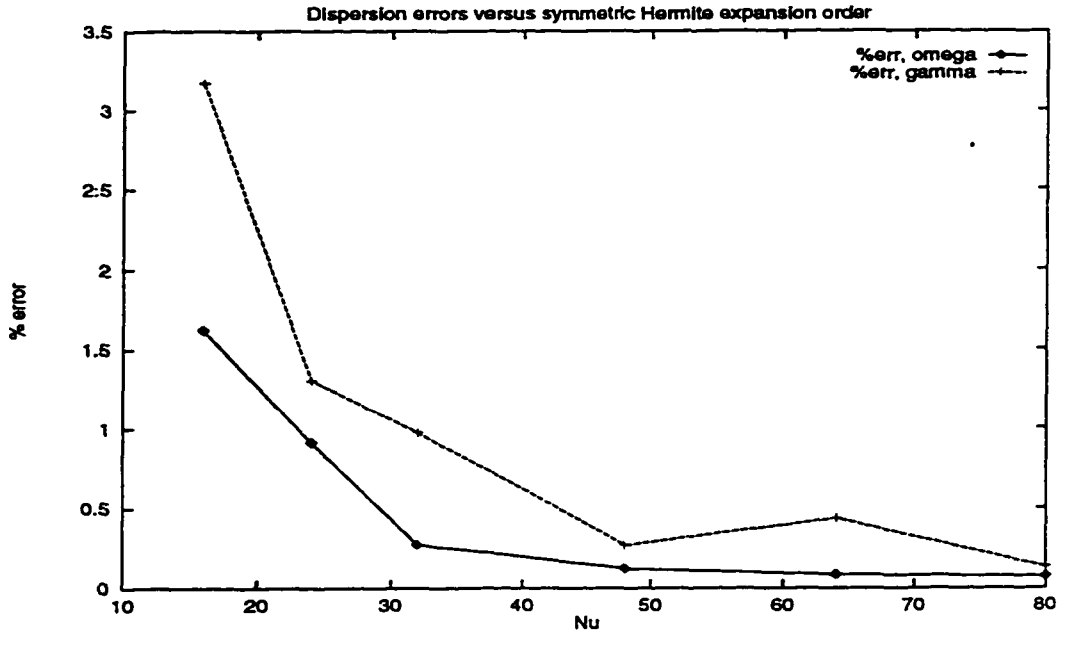

Figure 4.34: Bump-on-tail dispersion errors versus N_u for the symmetric Hermite method. Sampling errors are 0.32%.

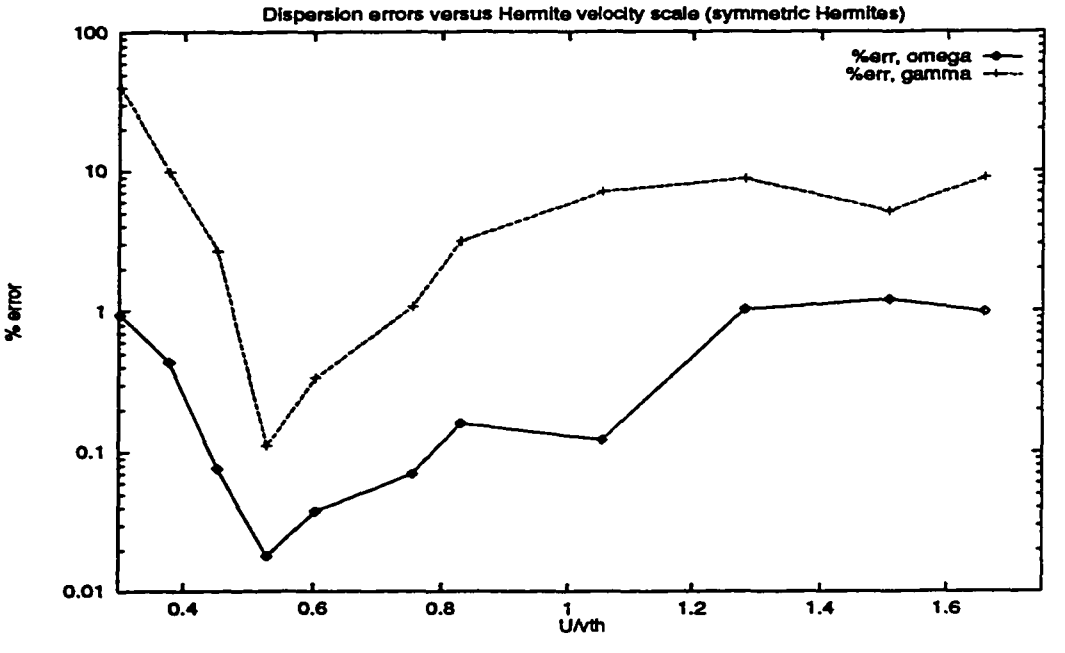

Figure 4.35: Bump-on-tail dispersion errors versus U for the even symmetric Hermite method. Sampling errors are 0.32%.

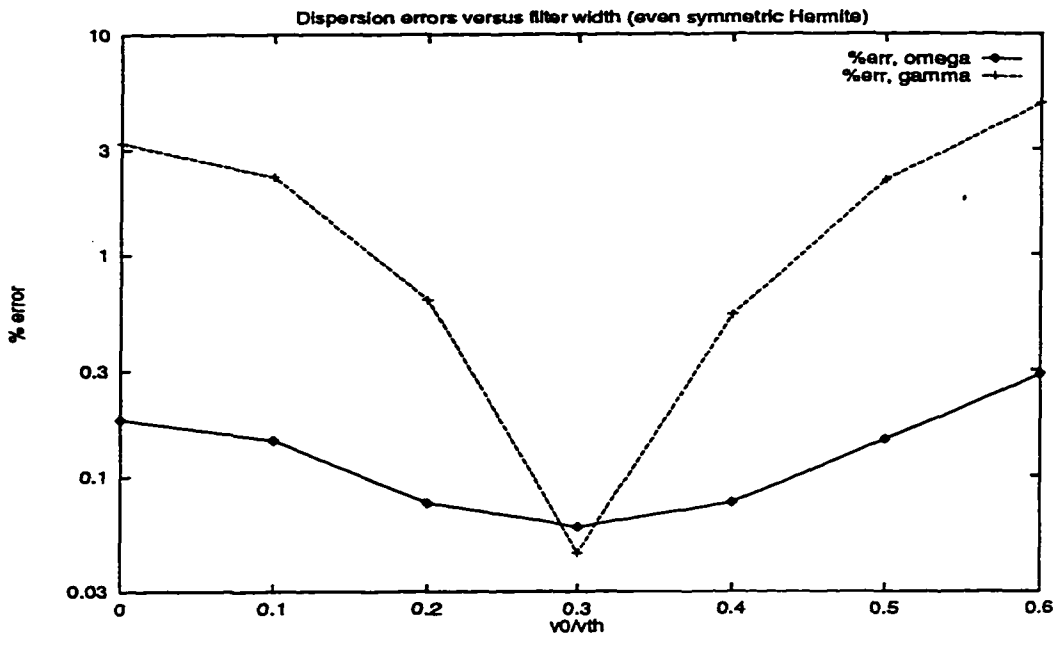

Figure 4.36: Bump-on-tail dispersion errors versus filter width v_0/v_{th} for the even symmetric Hermite method. Sampling errors are 0.49%.

4.2.4 Comparisons to PIC bump-on-tail simulations

In this section, we will compare results of the Fourier-Hermite BOT simulations to those of the 1d-1v ES1 PIC code [Bir.2]. To give the accuracy comparisons a fair basis, we used the bump-on-tail input "WBEAMPLS.INP," which was provided with the ES1 code; the Fourier-Hermite simulations used similar input parameters. The input deck for the PIC/FH comparisons is shown in Table 4.5. The PIC method used a spatial resolution of 128 grid points, whereas the Fourier-Hermite methods used 64 Fourier modes in order to make the run times comparable to the shortest runtime PIC simulations.

A sample of one of the ES1 simulations using 256000 macroparticles is shown in Figure 4.37. A comparison of ES1 results versus number of macroparticles is shown in Figure 4.38, including a comparison to the symmetric Hermite simulations using

Parameter	Symbol	Value used [units]
number of PIC particles	N_p	64000-512000
number densities	n_p	1.0 [#/m]
	n_b	0.01 [#/m]
thermal velocities	$v_{th,p}$	0.28284271 [m/s]
	$v_{th,b}$	7.0710678e-2 [m/s]
electron mass	m_e	1.0 [kg]
electron charge	e	-1.0 [C]
permittivity	ϵ_o	1.0 [F/m]
spatial resolution	N_x	64 (FH) or 128 (PIC)
velocity resolution	N_u	64 or 65
temporal resolution	Δt	7.9974 $\times 10^{-3}$ [τ_{pe}]
spatial length	L	20π [m]
velocity scales	U (asym)	0.4265493 [m/s]
	U (sym)	0.1333054 [m/s]
velocity filter width	v_o (0.2 $v_{th,p}$)	0.056568542 [m/s]
perturbation amplitude	ϵ	1.0×10^{-4}
perturbation wavenumber	k	10

Table 4.5: Standard values for comparison of PIC and FH bump-on-tail simulations.

Method	$\omega + i\gamma$ [1/sec] ($\epsilon = 1e-4$)	percent error
linear theory	0.9295028 + i 0.1084353	–
PIC, N_p=64000	0.9363793 + i 0.0898235	0.740 + i 17.2 %
PIC, N_p=128000	0.9326943 + i 0.0940847	0.343 + i 13.2 %
PIC, N_p=256000	0.9286062 + i 0.1010277	0.097 + i 6.83 %
PIC, N_p=512000	0.9286057 + i 0.1053760	0.097 + i 2.82 %
FH, symmetric (even)	0.9288249 + i 0.1076992	0.073 + i 0.68 %
FH, symmetric (odd)	0.9288249 + i 0.1076980	0.073 + i 0.68 %
FH, asymmetric	0.9289522 + i 0.1114054	0.059 + i 2.74 %

Table 4.6: Comparison of the BOT dispersion errors between a PIC code and the Hermite methods. Frequencies and growth rates were taken from E-field peaks lying between 6 and 10 τ_{pe}. Sampling errors are approximately 0.26% for all cases.

the same input BOT distribution. We see that the PIC results are approaching the symmetric Hermite solution as particle number increases to 512000 particles; however, the Hermite method used only 4096 unknowns.

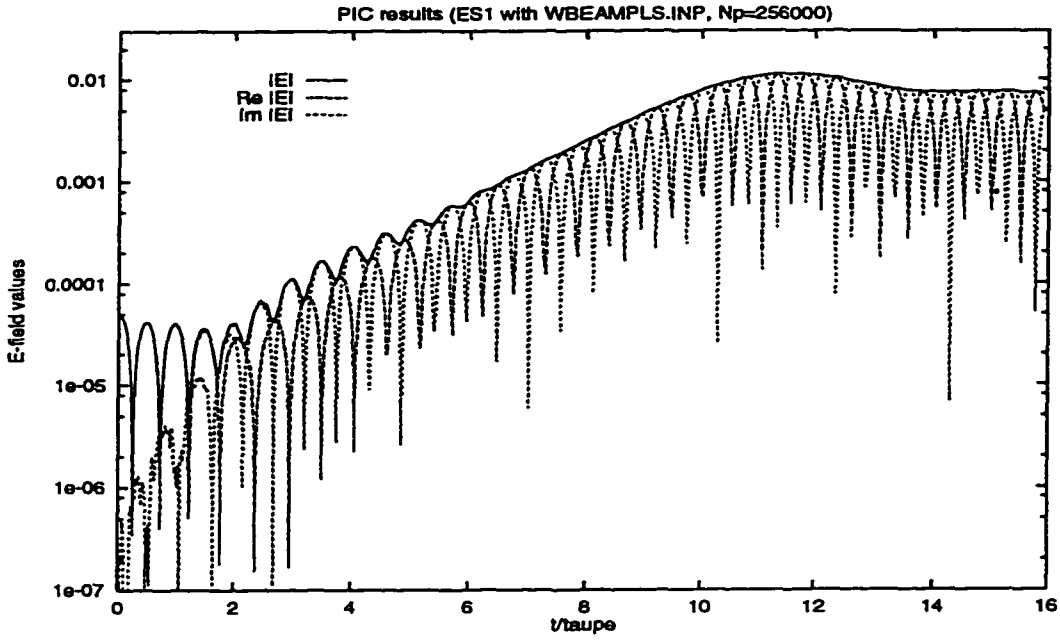

Figure 4.37: Evolution of the E-field during one PIC bump-on-tail simulation (N_p = 256000). Dispersion frequency and growth rate will be calculated from the various E-field peaks, as shown.

We may clearly see that the Hermite methods are superior to the PIC methods for warm plasma simulations by comparing both to linear theory. In Table 4.6, we see that the PIC code can predict the oscillation frequency of the electrostatic mode to within 1% of linear theory and as well as the Hermite methods for about 256000 particles. However, the growth rate γ of the mode is poorly modeled by PIC. Even for over 500000 particles, the error is around 3%. The symmetric Hermite schemes, using about 4000 unknowns, predicted both ω and γ to less than 1% of linear theory. The PIC simulation with 64000 particles had a runtime of 5.47 minutes on a dedicated IBM RS6000. The symmetric Fourier-Hermite simulation had a runtime of 53.5 seconds on the same machine *and* yielded much higher accuracy.

One may ask, "Why use PIC at all?" The methods described and tested in this dissertation have been designed for modeling plasmas and beams with significant

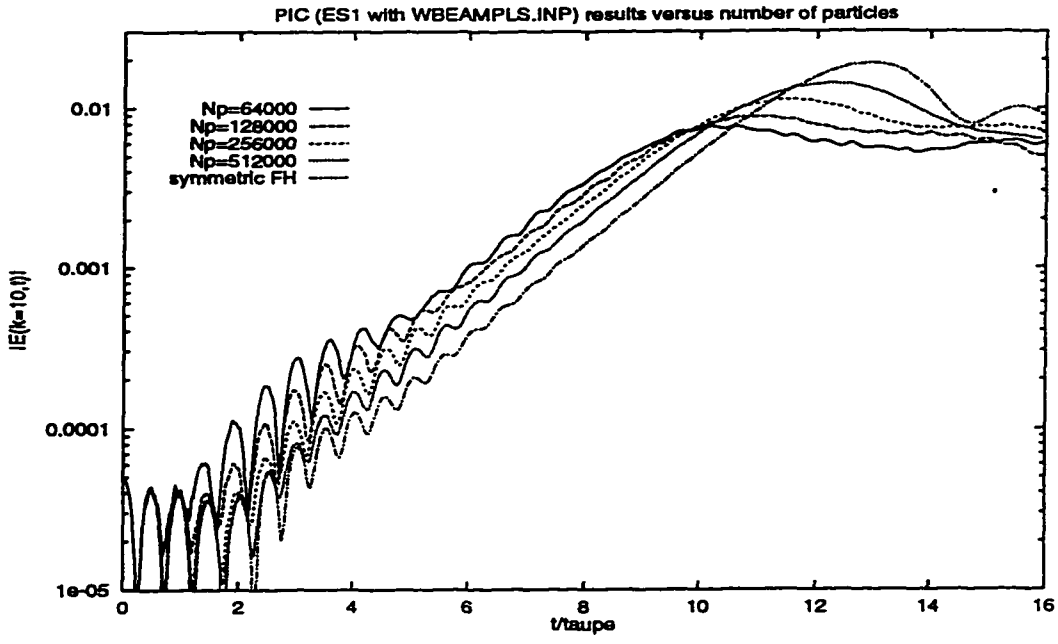

Figure 4.38: Evolution of the E-field during bump-on-tail simulations, comparing PIC (ES1) to FH methods. As the number of particles increases from 64000 to 512000, the PIC simulations appear to become more similar to the even symmetric Hermite simulations (also shown).

velocity spread relative to the drift velocity, i.e. $v_{th} \geq v_d$. Cold charged-particle beams with widely different drift velocities would be difficult to model efficiently using this scheme because spectral methods, in general, represent the *entire* phase-space. Simulations of systems with large regions of empty phase-space would require prohibitively large Fourier and Hermite expansion orders in order to obtain moderate accuracy.

The asymmetric Hermites, while performing better than PIC, did not perform as well as the symmetric Hermites. The maximum energy conservation error for the asymmetric Hermite simulation during the time of frequency and growth rate measurements was approximately $4.3 \times 10^{-5}\%$. The momentum conservation errors were at round-off level during the entire run. Conservation of the integral of f^2 was

very poor; errors in $\iint f^2 dxdu$ rose exponentially throughout the simulation to a maximum of 6000% after saturation of the E-field. During the time of frequency and growth rate measurements, the errors ranged from 20% up to 400%. This large error accounts for the discrepancy with linear predictions (most of this error is contained in the higher-order Hermite coefficients, which are not used in calculating the E-field).

The even and odd order symmetric Hermite simulations yielded equally accurate results. The maximum momentum conservation errors during the frequency and growth rate measurements were $1.3 \times 10^{-3}\%$ (even) and $2.2 \times 10^{-7}\%$ (odd). Energy conservation errors during this same period were $2.4 \times 10^{-5}\%$ (even) and $2.3 \times 10^{-4}\%$ (odd). Errors in the conservation of the integral of f^2 were less than $1.8 \times 10^{-4}\%$ for both methods. All of these errors should not measurably affect the physics in the linear regime, therefore demonstrating that the symmetric Hermite methods are superior to the asymmetric Hermites methods.

CHAPTER V

CONCLUSIONS

Performing collisionless kinetic plasma simulations is an arduous task due to the need for limited resolution. For example, long-time Landau damping is limited by the recursion phenomena. Although recursion can be delayed by using the various techniques as shown in this dissertation, we must realize that recursion is inherent in discretized velocity-space; it is unavoidable without the use of artificial damping. Filamentation in collisionless plasmas is also unavoidable. The Gaussian filtering used in these simulations could only delay its onset. Eventually, high frequency spatial and velocity scales develop and degrade the accuracy of the simulation.

In this dissertation, we combined Gaussian filtering and a velocity-scaled Hermite spectral expansion into one method in order to combat inaccuracies brought on by representing a continuous phase space with a discrete phase-space. Gaussian filtering helped to keep the velocity dependence smooth. The Hermite weighted-residuals method accurately represented the near Maxwellian velocity profiles of the plasma distribution functions. The velocity scale U was the fine tuning knob for the Hermite expansion basis. These features, combined with the split operator Vlasov-Poisson equations, yielded a fast, flexible, and accurate tool for the simulation of 1d-1v collisionless plasma physics.

These simulations using the Fourier-Hermite spectral representation of the filtered Vlasov-Poisson equations, solved with a splitting technique, displayed a number of laudable qualities:

Splitting. The splitting scheme, separating the linear advection and the non-linear acceleration term of the Vlasov equation, allowed an 8-fold decrease in computational time by avoiding the convolution sum in the evaluation of $E\partial_u f$ with only an $O(\Delta t^3)$ inaccuracy. Splitting also gives the computational physicist the freedom to design different time-advancing schemes for the separated advection and acceleration mappings.

Filtering. The Gaussian filter, previously applied in the Fourier-Fourier method by Klimas, preserved the collisionless plasma physics by avoiding the need for artificial damping, thereby providing a numerical technique for delaying filamentation. In addition, the filter enhanced the spectral accuracy of the Hermite methods, reduced the truncation errors, and improved the conservation properties of the symmetric Hermite method.

Hermite functions. The Hermite basis functions, chosen to the represent velocity space in these kinetic simulations, were immediately realized to be ideally suited for the representation of near Maxwellian velocity profiles. Hermite representations in the infinite domain, as opposed to the Fourier-Fourier representations in a finite one, were not accuracy-limited by the dynamics of particles in the high-energy tail and did not destroy momentum conservation by making velocity space periodic. The Fourier-Hermite discretization applied to the filtered Vlasov-Poisson equation was also not limited by large time-step as in the filtered Fourier-Fourier method. In addition, the smooth Hermite-weighted dis-

tribution functions represented low-density regions of phase space much better than PIC methods, and with far less noise. The symmetric Hermite method has now been successfully applied to the modeling of plasma kinetics for the first time.

Velocity scaling. Incorporating the velocity scale U in these Fourier-Hermite simulations allowed an increase in velocity resolution without extra computational cost. This, in turn, increased velocity resolution and enhanced spectral accuracy (which reduced the truncation errors). Optimal choices of the velocity scale yielded longer recursion times and greatly reduced errors compared to linear kinetic theory.

All of these features combine to form a stable numerical method that accurately conserves the physical constants of particle number, momentum, and energy and reliably predicts the linear phenomena of Landau damping and the growth of electrostatic fields in the physically unstable bump-on-tail distribution. This method is the Fourier-symmetric Hermite scheme, as described in this dissertation. We have seen that this method can perform much better than the Klimas Fourier-Fourier scheme and the popular PIC method, both in speed and accuracy.

During the inception and design of any numerical tool, we understand that we cannot create an omnipotent algorithm. Additional features and enhancements are always possible. The algorithms described in this dissertation are limited to the study of one-dimensional (1d-1v) spatially periodic electrostatic phenomena but serve as an excellent test-bed for algorithms that model more complex systems. Perhaps the most obvious "new feature" would be the inclusion of non-periodic physical boundaries by using Chebyshev or Legendre functions in space. This would allow

the modeling of plasma physics in planar diodes, for example. After including non-periodic boundaries, we might extend the code to 1d-2v by adding another velocity dimension. This addition would permit the numerical study of cross-field amplifiers.

Although these two enhancements would allow the modeling of a few experimental devices, the conservative and more prudent approach would be to also consider options that could make this one-dimensional method even better. Even one-dimensional plasma physics is still interesting and is not completely understood. With this goal in mind, let us consider two methods for improving spectral accuracy in long-time simulations: shifted/scaled Hermites and other velocity filters.

In this dissertation, we found significant improvements in simulations that used an optimal Hermite velocity scale U. This optimal velocity scale was chosen knowing the initial value of the thermal velocity v_{th} and was held constant during the simulations. However, the local thermal velocity changes with time and can be written

$$v_{th}(x,t) = \sqrt{\frac{2\,\mathrm{KE}(x,t)}{m}} \equiv \sqrt{\frac{\int u^2 f(x,u,t)\,du}{n(x,t)}}\,. \tag{5.0.1}$$

The design of a new Fourier and Hermite-based algorithm using a variable velocity scale $U(x,t)$ could possibly yield better results. Also, in some systems with a constant external E-field, the total momentum of the distribution changes. We could then design an algorithm with velocity dependence represented by shifted and scaled orthonormal Hermite functions

$$\Psi^n(v) = \Psi^n\left(\frac{u - s(x,t)}{U(x,t)}\right) \tag{5.0.2}$$

where the velocity shift $s(x,t)$ and the velocity scale $U(x,t)$ are allowed to vary with space and time. Using these basis functions, we might find an improvement in the simulations.

Another possible improvement upon the methods derived here is the use of a super-Gaussian velocity filter. Klimas suggested filters of the form [Kli.3]

$$F(u-u') = e^{-\frac{1}{2}\left(\frac{u-u'}{v_o}\right)^{2p}} \quad , p = 1, 2, \ldots \tag{5.0.3}$$

which strongly damp high-order velocity moments but leave the lowest $2p-1$ velocity moments invariant. In this work, we used this filter with $p = 1$; these other filters will not affect the E-field dynamics and may improve the spectral accuracy of the Hermite representation. Without further investigation, we do not know what complications or benefits these different filters may produce.

In addition to shifted/scaled Hermites and super-Gaussian velocity filters, we could also investigate time stability issues. In this dissertation, we used the $O(\Delta t^2)$ splitting scheme with the trusty Runge-Kutta method; we could explore the possibility of using a higher-order splitting scheme, an implicit time-stepping scheme such as Crank-Nicholson, or applying the time-varying E-field splitting scheme discussed briefly in Chapter I.

Before incorporating non-periodic boundaries or adding dimensionality to the code, the issues of velocity resolution and temporal stability should be addressed to insure that we will always have a robust method.

The Fourier-Hermite methods, first derived for kinetic plasma simulations in the 1960's and now outfitted with Gaussian filtering and variable velocity scaling, have been reborn and can now play a significant role in the field of computational plasma physics. With confidence in the symmetric Hermite method and its ability to model plasma physics in the linear regime, we see it is an ideal tool for the study of non-linear plasma dynamics, including saturation of E-fields and the formation of steady-state phase-space structures. Unfortunately, the asymmetric Hermite method, although

computationally cheaper, has an inherent flaw: it can be unstable at long times. The linear predictions found from the asymmetric Hermite simulations were accurate; however, this method has no usefulness as a tool for studying long-time plasma behavior. For the self-consistent study of 1d-1v plasma physics in warm, non-relativistic plasmas, the symmetric Hermite method described in this dissertation is the proper choice.

APPENDICES

APPENDIX A

Filamentation of Collisionless Distributions

A spatially non-uniform collisionless plasma distribution will naturally develop large derivatives in the velocity direction, a process known as filamentation. For example, consider the solution of the field-free Vlasov equation,

$$\frac{\partial f(x,u,t)}{\partial t} = -u\frac{\partial f}{\partial u}\,, \quad f(x,u,0) \Longrightarrow f(x,u,t) = f(x-ut,u,0) \tag{A.1}$$

which is simply free-streaming. The u-derivative of this solution is

$$\frac{\partial f(x,u,t)}{\partial u} = \frac{\partial f(\xi,u,0)}{\partial u}\bigg|_{\xi=x-ut} - t\frac{\partial f(x-vt,u,0)}{\partial x} \tag{A.2}$$

which grows unbounded with increasing time [Kli.2].

For the non-linear problem, we find that filamentation still exists. Using the definition of the *convective derivative*

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + \frac{q}{m}E(x,t)\frac{\partial}{\partial u} \tag{A.3}$$

and applying this operator to a distribution $f(x,u,t)$ puts the Vlasov equation into a compact form, $Df/Dt = 0$.

Taking the ∂_u derivative of the Vlasov equation yields

$$\frac{\partial}{\partial u}\frac{Df(x,u,t)}{Dt} = \frac{\partial^2 f}{\partial u \partial t} + \frac{\partial f}{\partial x} + u\frac{\partial^2 f}{\partial x \partial u} + \frac{q}{m}E(x,t)\frac{\partial^2 f}{\partial u^2} = 0\,. \tag{A.4}$$

Now, if we take the convective derivative of the quantity $\partial_u f$, we find

$$\frac{D}{Dt}\left[\frac{\partial f}{\partial u}\right] = \frac{\partial^2 f}{\partial t \partial u} + u\frac{\partial^2 f}{\partial x \partial u} + \frac{q}{m}E(x,t)\frac{\partial^2 f}{\partial u^2} \tag{A.5}$$

Taking the difference of Equations A.4 and A.5, we find

$$\frac{D}{Dt}\left[\frac{\partial f}{\partial u}\right] = -\frac{\partial f}{\partial x} \tag{A.6}$$

so that long orbits of constant distribution function $f(x,u,t)$ described by the convective derivative, the u-derivative of the distribution increases with time at a rate proportional to the x-derivative of the distribution. Hence, for spatially uniform plasmas, there is no filamentation.

Continuing this idea further, if we take the convective derivative of the quantity $\partial_x f$, we see that

$$\frac{D}{Dt}\left[\frac{\partial f}{\partial x}\right] = \frac{\partial^2 f}{\partial t \partial x} + u\frac{\partial^2 f}{\partial u \partial x} + \frac{q}{m}E(x,t)\frac{\partial^2 f}{\partial u \partial x} \tag{A.7}$$

and taking the u-derivative of the Vlasov equation, we find

$$\frac{\partial}{\partial x}\left[\frac{Df}{Dt}\right] = \frac{\partial^2 f}{\partial x \partial t} + u\frac{\partial^2 f}{\partial x \partial u} + \frac{q}{m}\frac{\partial E(x,t)}{\partial x}\frac{\partial f}{\partial u} + \frac{q}{m}E(x,t)\frac{\partial^2 f}{\partial x \partial u}\,. \tag{A.8}$$

The difference of these yields,

$$\frac{D}{Dt}\left[\frac{\partial f}{\partial x}\right] = -\frac{q}{m}\frac{\partial E(x,t)}{\partial x}\frac{\partial f}{\partial u} \tag{A.9}$$

$$= -\frac{q\rho(x,t)}{\epsilon_o m}\frac{\partial f}{\partial u} \tag{A.10}$$

$$= -\omega_p^2(x,t)\frac{\partial f}{\partial u} \tag{A.11}$$

where $\omega_p^2(x,t) = q^2 \mathrm{n}(x,t)/m\epsilon_o$ for a single-species plasma and $\mathrm{n}(x,t)$ is the local number density.

Taking the 2^{nd} convective derivative of Equation A.6 and inserting Equation A.11, we get a 2^{nd}-order differential equation for the velocity derivative of $f(x,u,t)$

$$\frac{D^2}{Dt^2}\left[\frac{\partial f}{\partial u}\right] = \omega_p^2(x,t)\frac{\partial f}{\partial u}\,. \tag{A.12}$$

Along the orbit of constant distribution function $f(x, u, t)$, this equation has solutions that exponentially decay and grow! Filamentation is, therefore, a serious concern during the simulation of collisionless plasmas.

APPENDIX B

Limitations of the filtered Fourier-Fourier scheme

In the paper by Klimas and Farrell [Kli.3], the filtered FF advection mapping advancing the distribution in x-space is given explicitly by Equation (31) of that paper

$$\bar{K}_m(\nu, \Delta\tau) = e^{-\epsilon m \Delta\tau(\nu - \frac{m}{2}\Delta\tau)} \bar{K}_m(\nu - m\Delta\tau, 0) \tag{B.1}$$

where m is the spatial Fourier mode number of the filtered distribution coefficient $\bar{K}_m(\nu, t)$, ν is the velocity mode number, $\epsilon = (\pi v_o)^2$ is the filtering constant, and $\Delta\tau$ is the normalized time-step. This normalized time-step in terms of real time and system length scales is given by

$$\Delta\tau = \Delta t \frac{L_u}{L_x} \tag{B.2}$$

where Δt is in seconds, L_x is the system spatial length, and L_u is the maximum resolved velocity of the system.

The FF advection mapping above may be performed exactly by selecting $\Delta\tau$ to be an integer so that the difference $\nu - m\Delta\tau$ is integer corresponding to some mode ν'; however, an integer $\Delta\tau$ selection makes Δt, the actual time in seconds, so large that for the fastest particle will cross the entire system in one time step! To compensate for this, a FF method would require $L_u/L_x \gg 1\ s^{-1}$ so that there

would be very, very few "fastest" particles in the system. If L_u is very large, then N_u must be appropriately large as well in order to retain reasonable velocity resolution $\Delta u = L_u/N_u$; an accurate FF simulation would require $L_u \approx 1000 v_{th}$ and $N_u \approx 1000$. To avoid this considerable computation cost, the FF method must use non-integer $\Delta\tau$ and interpolate $\bar{K}_m(\nu - m\Delta\tau, 0)$ in ν space.

Unfortunately, interpolation in ν space gives rise to a numerical instability. We see that for some mode numbers m and ν and time-step $\Delta\tau$, the exponential factor in Equation B.1 can be greater than 1. For this to be true, we require

$$-\epsilon m\Delta\tau(\nu - \frac{1}{2}m\Delta\tau) \;>\; 0 \tag{B.3}$$

for some m, ν, and $\Delta\tau$. There are two cases when $\epsilon > 0$ (filtered simulations):

$$\underline{\text{Case 1 :}} \quad m\nu > 0 \longrightarrow \Delta\tau > \frac{2\nu}{m} \tag{B.4}$$

$$\underline{\text{Case 2 :}} \quad m\nu < 0 \longrightarrow \Delta\tau > 0\,. \tag{B.5}$$

For $\nu = 0$, these cases are equivalent and the exponential term is $\exp\left[\frac{\epsilon}{2}(m\Delta\tau)^2\right]$, which is always greater than 1 for non-zero $\epsilon = (\pi v_o)^2$. An error in the interpolation is amplified for every time-step. Thus, by using non-integer values of time-step $\Delta\tau$ in the filtered FF advection, the errors in interpolation will eventually break down the method. This FF filter instability is shown below in Figures B.1 and Figure B.2. Note, the $\Delta\tau = 1$ simulation is stable, even with filtering.

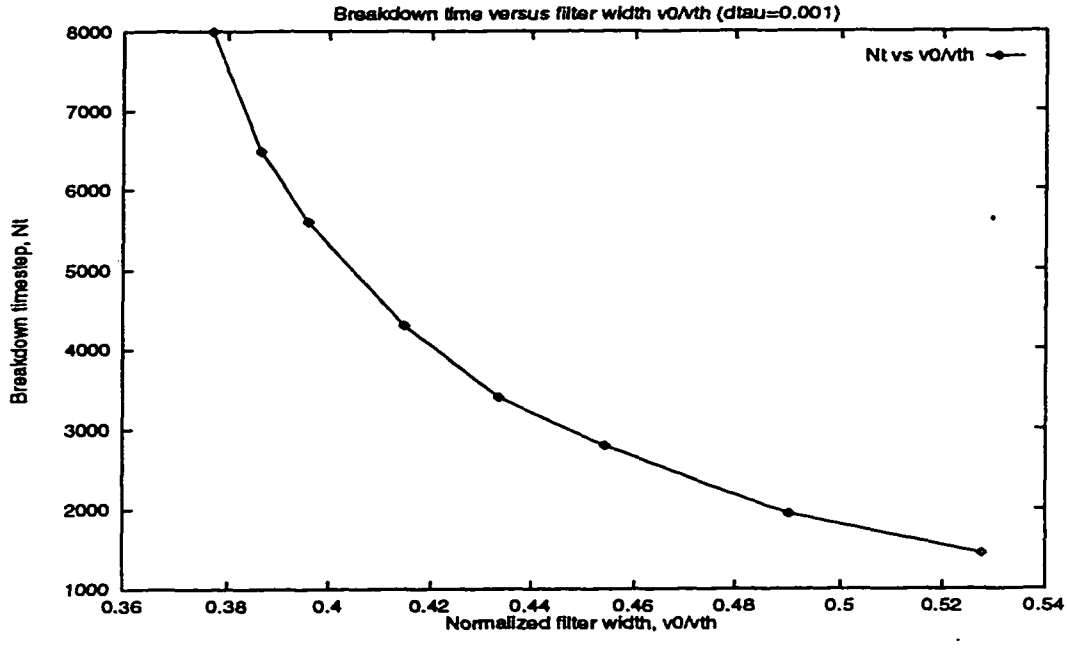

Figure B.1: FF filter instability versus filter width, v_o, for fixed time-step Δt.

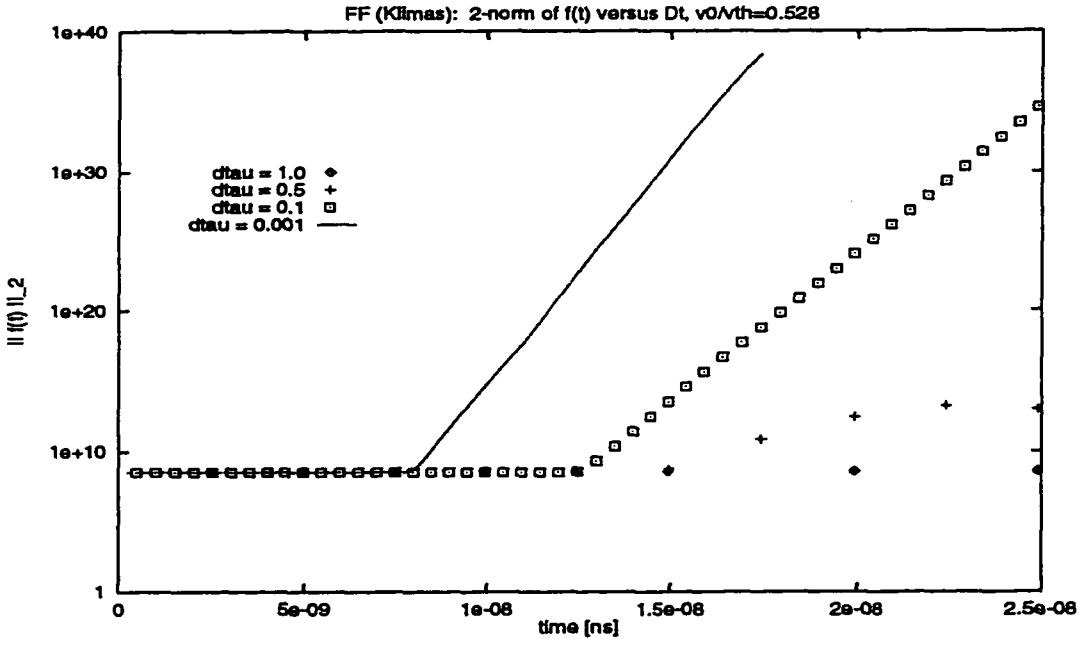

Figure B.2: FF filter instability versus time-step, Δt, for fixed filter width v_o.

APPENDIX C

Vlasov-Poisson Bracketology

The time-rate of change for a functional G of 1d-1v phase-space variables (f, E)

$$G(f,E) = \iint g(f, \partial_x f, \partial_u f, \ldots, E, \partial_x E, \ldots)\, dxdu \tag{C.1}$$

is found from the non-canonical bracket [Mar.1] under the initial constraint of Poisson's equation $\partial_x E(x,0) = \rho_\alpha(x,0)/\epsilon_o$

$$\dot{G} = \{\{G,H\}\} \tag{C.2}$$

$$= \iint \left[f(x,u,t) \left\{ \frac{\delta G}{\delta f}, \frac{\delta H}{\delta f} \right\} + \frac{q_\alpha}{\epsilon_o} \left(\frac{\delta G}{\delta E} \frac{\partial f}{\partial u} \frac{\delta H}{\delta f} - \frac{\delta H}{\delta E} \frac{\partial f}{\partial u} \frac{\delta G}{\delta f} \right) \right] dxdu \tag{C.3}$$

where $\{a,b\} = \partial_x a \partial_u b - \partial_x b \partial_u a$ is the canonical Poisson bracket and the system Hamiltonian is

$$H(f,E) = \underbrace{\iint \frac{1}{2} u^2 f(x,u,t)\, dxdu}_{H_{\text{kinetic}}} + \underbrace{\frac{\epsilon_o}{2m_\alpha} \int E^2(x,t)\, dx}_{H_{\text{fields}}} . \tag{C.4}$$

To find an equation for the time-rate of change of the filtered distribution $\bar{f}(x_o, u_o, t)$, we define the functional

$$G(f,E) = \iint \bar{f}(x,u',t) \delta(x - x_o) \delta(u' - u_o)\, dxdu' \tag{C.5}$$

$$= \iint f(x,u,t) \delta(x - x_o) \left[\int F(u - u') \delta(u' - u_o)\, du' \right] dxdu \tag{C.6}$$

$$= \iint f(x,u,t) \delta(x - x_o) F(u - u_o)\, dxdu \tag{C.7}$$

where $F(u-u_o)$ is the filter function defined (Equation 1.4.10) at u_o. Using the identities for Poisson brackets [Gol.1], we find

$$\frac{\partial \bar{f}(x_o,u_o,t)}{\partial t} = \iint \frac{\delta G}{\delta f}\left\{\frac{\delta H}{\delta f}, f\right\} + \frac{q_\alpha}{\epsilon_o}\left(0 - \frac{\epsilon_o}{m_\alpha}E(x,t)\frac{\partial f}{\partial u}\frac{\delta G}{\delta f}\right) dxdu \quad \text{(C.8)}$$

$$= -\iint \delta(x-x_o)F(u-u_o)\left(u\frac{\partial f}{\partial x} + \frac{q_\alpha}{m_\alpha}E(x,t)\frac{\partial f}{\partial u}\right)dxdu \quad \text{(C.9)}$$

$$= -\left(u\frac{\partial \bar{f}}{\partial x} - v_o^2\frac{\partial^2 \bar{f}}{\partial x \partial u} + \frac{q_\alpha}{m_\alpha}E(x,t)\frac{\partial \bar{f}}{\partial u}\right)\Bigg|_{x_o,u_o} \quad \text{(C.10)}$$

which is the filtered Vlasov equation for the distribution value $\bar{f}(x_o,u_o,t)$. If the filter width v_o is zero, we regain the Vlasov equation for the unfiltered distribution value $f(x_o,u_o,t)$.

To find $\dot{E}(x_o,t)$, we define the functional $G(f,E) = \int E(x,t)\delta(x-x_o)\,dx$ and see that

$$\dot{E}(x_o,t) = \iint 0 + \frac{q_\alpha}{\epsilon_o}\left[\delta(x-x_o)\frac{\partial f}{\partial u}\left(\frac{1}{2}u^2\right) + 0\right]dxdu \quad \text{(C.11)}$$

$$= -\frac{q_\alpha}{\epsilon_o}\int_{-\infty}^{\infty} u f(x_o,u,t)\,du \quad \text{(C.12)}$$

after integration by parts in u and evaluation of the integral over x. Thus, we recover Ampere's Law for the electric field $E(x_o,t)$.

APPENDIX D

Exact Fourier-Asymmetric Hermite Mapping

D.1 Exact Advection Mapping

The X-shifts may be exactly performed by recognizing that the tridiagonal matrix form of the right-hand side of Equation (2.3.2) may be diagonalized using the eigenfunctions $\vec{\Psi}_j = [\Psi_0(\lambda_j), \ldots, \Psi_{N_u}(\lambda_j)]^T$ for eigenvalues λ_j satisfying $\Psi_{N_u+1}(\lambda_j) = 0$ (the Gauss-Hermite quadrature points). To see this, let us write Equation 2.3.2 in matrix form,

$$\frac{d\vec{F}^m}{dt} = C_m \mathbf{A} \vec{F}^m \tag{D.1}$$

where $\vec{F}^m = [f^{m0}, f^{m1}, ..., f^{mn}, ..., f^{mN_u}]^T$, $\mathbf{A}$ represents the $N_u + 1$-dimensional matrix of the X-shift and scalar C_m is defined

$$C_m = -\frac{im2\pi U_\alpha}{L_x}, \quad m = 0, ..., N_x - 1 \, . \tag{D.2}$$

We may decompose the time-dependence of $\vec{F}^m(t)$ with a basis of time-independent eigenvectors $\vec{G}_\lambda$ to be determined from $\mathbf{A}$,

$$\vec{F}^m(t) = \sum_\lambda a^m_\lambda(t) \vec{G}_\lambda \, . \tag{D.3}$$

Using this relation and the eigenvalue equation $\mathbf{A}\vec{G}_\lambda = \lambda \vec{G}_\lambda$ in Equation D.1, we get

$$\sum_\lambda \dot{a}^m_\lambda(t) \vec{G}_\lambda = C_m \sum_\lambda \mathbf{A} \vec{G}_\lambda a^m_\lambda(t) = C_m \sum_\lambda \lambda \vec{G}_\lambda a^m_\lambda(t) \tag{D.4}$$

which is satisfied if for every λ the time-rate of the change of the coefficients is given by $\dot{a}^m_\lambda(t) = C_m \lambda a^m_\lambda(t)$, which has the solution

$$a^m_\lambda(t) = a^m_\lambda(0) \exp[\lambda C_m t] \,. \tag{D.5}$$

Using this, the vectors $\vec{F}^m(t)$ become

$$\vec{F}^m(t) = \sum_\lambda a^m_\lambda(0) \exp[\lambda C_m t] \vec{G}_\lambda \tag{D.6}$$

for $m = 0, ..., N_x - 1$. To find the eigenvalues λ and eigenvectors $\vec{G}_\lambda$ of $\mathbf{A}$, we now use the X-shift (Equation 2.3.2) to see

$$\mathbf{A}\vec{G}_\lambda = \begin{bmatrix} 0 + \sqrt{\frac{1}{2}} G^1_\lambda \\ G^2_\lambda + \sqrt{\frac{1}{2}} G^0_\lambda \\ \vdots \\ \sqrt{\frac{n+1}{2}} G^{n+1}_\lambda + \sqrt{\frac{n}{2}} G^{n-1}_\lambda \\ \vdots \\ \sqrt{\frac{N_u+1}{2}}\, 0 + \sqrt{\frac{N_u}{2}} G^{N_u-1}_\lambda \end{bmatrix} = \lambda \begin{bmatrix} G^0_\lambda \\ G^1_\lambda \\ \vdots \\ G^n_\lambda \\ \vdots \\ G^{N_u}_\lambda \end{bmatrix} \tag{D.7}$$

since $G^{N_u+1}_\lambda = 0$ by truncation of the series. Thus the G^n_λ satisfy the Hermite recursion relation

$$\lambda G^n_\lambda = \sqrt{\frac{n+1}{2}} G^{n+1}_\lambda + \sqrt{\frac{n}{2}} G^{n-1}_\lambda \tag{D.8}$$

except that $\sqrt{\frac{N_u}{2}} G^{N_u-1}_\lambda = G^{N_u}_\lambda$, so we may identify the eigenvectors of $\mathbf{A}$ to be $\vec{G}_{\lambda_j} = [\Psi^0(\lambda_j), \Psi^1(\lambda_j), \ldots, \Psi^{N_u}(\lambda_j)]$ provided that λ_j is a root of Ψ^{N_u+1}. Equation D.6 now becomes

$$\vec{F}^m(t) = \sum_{j=0}^{N_u} a^m_{\lambda_j}(0) \exp[\lambda_j C_m t] \vec{\Psi}(\lambda_j) \tag{D.9}$$

where $\vec{\Psi}(\lambda_j) = [\Psi^0(\lambda_j), \Psi^1(\lambda_j), \ldots, \Psi^{N_u}(\lambda_j)]$.

To solve for $a^m_{\lambda_j}(0)$, we must set $t = 0$ for the n^{th} term to yield

$$f^{mn}(0) = \sum_{j=0}^{N_u} a^m_{\lambda_j}(0) \Psi^n(\lambda_j) \tag{D.10}$$

which is satisfied if we choose,

$$a^m_{\lambda_j}(0) = \sum_{k=0}^{N_u} \bar{w}_j f^{mk}(0)\Psi_k(\lambda_j) \tag{D.11}$$

where $\bar{w}_j$ are the Gauss-Hermite quadrature weights given in Equation 2.4.8. This selection for $a^m_{\lambda_j}(0)$ may be easily confirmed using the Gauss-Hermite quadrature formula given in Equation 2.4.4.

Inserting Equation D.11 for $a^m_{\lambda_j}(0)$ into the eigenfunction expansion of $f^{mn}_\alpha(t_o)$ Equation D.10, we get an expression for the X-shift which exactly advances the distribution from time t_o to t_a,

$$f^{mn}_\alpha(t_a) = \sum_{k=0}^{N_u}\sum_{j=0}^{N_u} \bar{w}_j e^{\frac{1}{2}C_m\lambda_j\Delta t}\Psi^n(\lambda_j)\Psi_k(\lambda_j)f^{mk}_\alpha(t_o) \tag{D.12}$$

where $u_j = \lambda_j U_\alpha$ and C_m is defined in Equation D.2. This expression may be further reduced by realizing that the sum over k is the inverse Hermite transform to u-space,

$$f^{mn}_\alpha(t_a) = \sum_{j=0}^{N_u} \bar{w}_j e^{\frac{1}{2}C_m\lambda_j\Delta t}\Psi^n(\lambda_j)\left[\sum_{k=0}^{N_u}\Psi_k(\lambda_j)f^{mk}_\alpha(t_o)\right] \tag{D.13}$$

$$= \sum_{j=0}^{N_u} \bar{w}_j e^{\frac{1}{2}C_m\lambda_j\Delta t}\Psi^n(\lambda_j)f^m_\alpha(u_j,t_o)\,. \tag{D.14}$$

From this we can identify,

$$f^m_\alpha(u_j,t_a) = \bar{w}_j e^{\frac{1}{2}C_m\lambda_j\Delta t}f^m_\alpha(u_j,t_o) \tag{D.15}$$

This equation is an exact formulation of the mapping $\mathbf{M}_x$, as introduced earlier in Section 1.5. This exact algorithm for solution of the X-shift is costly and was not performed in the simulations shown in this dissertation; two Hermite transforms are required *per* full time step at a cost of $O(N_u^2)$ per coefficient. The derivation, shown above, is provided for the sake of completeness.

D.2 Exact Acceleration Mapping

For the asymmetric Hermite method, the field-acceleration equation (V-shift) may also be exactly integrated. Using Equation 2.3.5 for the evolution of Hermite mode $n = 0$, we find

$$\frac{\partial f_\alpha^n(x,t)}{\partial t} = \frac{\sqrt{2n}\, q_\alpha}{m_\alpha U_\alpha} E(x,t) f_\alpha^{n-1}(x,t) \tag{D.16}$$

$$\frac{\partial f_\alpha^0(x,t)}{\partial t} = \frac{\sqrt{2 \cdot 0}\, q_\alpha}{m_\alpha U_\alpha} E(x,t) \cdot (0) = 0 \tag{D.17}$$

so $f_\alpha^0(x,t_b) = f_\alpha^0(x,t_a)$ during the V-shift. For the $n = 1$ Hermite mode,

$$\frac{\partial f_\alpha^1(x,t)}{\partial t} = \frac{\sqrt{2}\, q_\alpha}{m_\alpha U_\alpha} E(x,t) f_\alpha^0(x,t_a)\,. \tag{D.18}$$

Integrating over the time interval $t : [t_a, t_b]$ and possibly allowing $E(x,t)$ an explicit time-dependence

$$f_\alpha^1(x,t_b) = f_\alpha^1(x,t_a) + \frac{\sqrt{2}\, q_\alpha}{m_\alpha U_\alpha} f_\alpha^0(x,t_a) \int_{t_a}^{t_b} E(x,t)\, dt \tag{D.19}$$

$$= f_\alpha^1(x,t_a) + \frac{\sqrt{2}\, q_\alpha}{m_\alpha U_\alpha} \epsilon(x,t_b) f_\alpha^0(x,t_a) \tag{D.20}$$

where we have pulled the constant $f_\alpha^0(x,t_a)$ from the integration and defined,

$$\epsilon(x,t_b) \equiv \int_{t_a}^{t_b} E(x,t)\, dt\,. \tag{D.21}$$

If we assume a constant E-field through the V-shift, this will just be $\epsilon(x,t_b) = E(x,t_a)\Delta t$ where $\Delta t = t_b - t_a$. However, we can do better than a constant E-field by using the time-derivative of Ampere's Law (Equation 1.4.8)

$$\frac{\partial J(x,t)}{\partial t} = \sum_\alpha \left(\frac{U_\alpha^2 q_\alpha}{\sqrt{2}\epsilon_o} \right) \frac{\partial f_\alpha^1(x,t)}{\partial t} \tag{D.22}$$

$$= \sum_\alpha \left(\frac{U_\alpha q_\alpha^2}{m_\alpha \epsilon_o} \right) f_\alpha^0(x,t_a) E(x,t) \tag{D.23}$$

$$\frac{\partial^2 E(x,t)}{\partial t^2} = -\omega_p^2(x) E(x,t) \tag{D.24}$$

whose solution is

$$E(x,t) = E(x,t_a)\cos\left[\omega_p(x)(t-t_a)\right] - \frac{J(x,t_a)}{\omega_p(x)}\sin\left[\omega_p(x)(t-t_a)\right] \tag{D.25}$$

where we define the local plasma frequency at "time" t_a

$$\omega_p^2(x) = \sum_\alpha \frac{U_\alpha q_\alpha^2 f_\alpha^0(x,t_a)}{\epsilon_o m_\alpha} \tag{D.26}$$

and $E(x,t_a)$ and $J(x,t_a)$ are calculated via Equations 2.3.11 and 2.3.13, respectively. Inserting this into Equation D.21 and performing the integral, we get

$$\varepsilon(x,t_b) = \frac{E(x,t_a)}{\omega_p(x)}\sin\omega_p(x)\Delta t + \frac{J(x,t_a)}{\omega_p^2(x)}(\cos\omega_p(x)\Delta t - 1) \tag{D.27}$$

after using the initial condition $\varepsilon(x,t_a) = 0$ and $\Delta t = t_b - t_a$.

Returning to the full V-shift equation (Equation D.16), we may integrate with respect to time and use the definition of $\varepsilon(x,t)$ above. Then, by induction, starting with $f_\alpha^0(x,t) = f_\alpha^0(x,t_a)$ and $\varepsilon(x,t_a) = 0$, we see that the *exact V-shift equation* for the asymmetric Hermite method is

$$f_\alpha^n(x,t_b) = \sqrt{n!}\sum_{k=0}^{n}\left[\frac{\sqrt{2}q_\alpha\varepsilon(x,t_b)}{m_\alpha U_\alpha}\right]^k \frac{f_\alpha^{n-k}(x,t_a)}{k!\sqrt{(n-k)!}}. \tag{D.28}$$

We may scale the distribution using $g_\alpha^n(x,t) = f_\alpha^n(x,t)/\sqrt{n!}$ to write the exact V-shift as

$$g_\alpha^n(x,t_b) = \sum_{k=0}^{n}\frac{[\beta(x,t_b)]^k}{k!}g_\alpha^{n-k}(x,t_a), \tag{D.29}$$

$$\beta(x,t_b) = \left[\frac{\sqrt{2}q_\alpha\varepsilon(x,t_b)}{m_\alpha U_\alpha}\right] \tag{D.30}$$

which is a more computationally efficient form. However, it still requires a lower-triangular matrix multiply which is $O(n^2)$ operations per $g_\alpha^n(x,t)$ coefficient, as compared to order $O(1)$ operations for the RK4 method.

APPENDIX E

Listing of FORTRAN-77 code VMSFH.F

```
************************************************************************
* Program: Vlasov-Maxwell Solver_Fourier-Hermite (VMS_FH)              *
************************************************************************
* Author:      Joseph Schumer, University of Michigan,                 *
* Initiated:   18May95       Nuclear Engineering                       *
* Revised:     24Nov96                                                 *
************************************************************************
*
* Initialize variables
      include 'parameter.incl'                                          10
      include 'common.incl'
      call setup
*
* Post-process f(m,n,s,t)->f(x,u,s,t), if requested, then end.
      if (ipost.eq.1) then
         write (6,*) ' '
         write (6,*) 'You are in POST-PROCESSOR mode.'
         write (6,*) 'The file FMN.DAT will be used.'
         call postfxu
         goto 999                                                       20
      endif
*
* Write initial data and open proper files
      if (icalex.eq.1) call fexact
      write (6,*) 'Writing initial distributions f(x,u) and E(x).'
      call ecalc
      call writer
      call monitor
*
****************** Main integration loop ************************       30
*
      write (6,*) 'Beginning main iterations.'
      do 10 iter = 1,niter
         time = dfloat(iter)*dtime
         call xshift(dtx)
         call ecalc
```

```
        call vshift(dtv)
        call xshift(dtx)
        call monitor
        call writer                                                    40
 10   continue
      write (6,*) 'End of VMS-FH executable.'
*
 999  end
*
*************************************************************************
      subroutine reader
*************************************************************************
*
      include 'parameter.incl'                                         50
      include 'common.incl'
*
* Load input deck
      open (unit=ifin,file='xvmsfh.dat',status='old',err=1000)
      read (ifin,4)
      read (ifin,*) nx,nu,nxstep,nustep
      read (ifin,*)
      read (ifin,*) xmin,xmax
      read (ifin,*)
      read (ifin,*) iasym,iroot                                        60
      read (ifin,3)
      read (ifin,*) iorder,dtpper,niter
      read (ifin,*)
      read (ifin,*) iof,ioe,iomon,iofu,ioehst
      read (ifin,*)
      read (ifin,*) itest,itrans,ipost
      read (ifin,3)
      read (ifin,*) nspec
      do 5 ispec = 1,nspec
        read (ifin,2)                                                  70
        read (ifin,*) ms(ispec),qs(ispec),uscale(ispec)
        read (ifin,2)
        read (ifin,*) ifdist(ispec),nbeam(ispec),nperb(ispec)
        do 10 ib = 1,nbeam(ispec)
          read (ifin,*)
          read (ifin,*) amp0(ib,ispec),x0(ib,ispec),x1(ib,ispec),u0(ib,ispec),u1(ib,ispec)
          if (uscale(ispec).le.u0(ib,ispec)/rt2*iasym) stop 'U scale too small!'
 10     continue
        do 11 ip = 0,nperb(ispec)-1
          read (ifin,*)                                                80
          read (ifin,*) amp1(ip,ispec),hk(ip,ispec)
 11     continue
 5    continue
      read (ifin,3)
      read (ifin,*) ifilt,v0,collf
*
* Shut-off filter if ifilt=0
      v0 = v0*ifilt
*
      if (iasym.eq.1) then                                             90
```

```
* Check filter factor BETA is not imaginary (limit set by beta > 0.1)
        v0lim = 0.70356236
        do is= 1,nspec
          if (v0.gt.v0lim*uscale(is)) stop 'beta < 0.1 limit!'
        enddo
      endif
*
      read (ifin,3)
      read (ifin,*) iself,icalex
      read (ifin,*)                                                    100
      read (ifin,*) iedist,ampe,omega,hek,itmin,itmax,itorx
      read (ifin,3)
      read (ifin,*) perm0
*
      close (ifin)
*
* Check input deck consistency
*
      if (nx.gt.jmax.or.nu.gt.kmax) stop 'Input grid exceeds design memory.'
      if (nxstep.gt.nx.or.nustep.gt.nu) pause 'Output grid for plots empty...'   110
      if (xmin.ge.xmax) stop 'Input xmax/xmin boundaries inconsistent.'
*
 2    format (/)
 3    format (//)
 4    format (///)
*
      return
 1000 stop 'Bad file name for input deck.'
      end
*                                                                      120
*******************************************************************************
      subroutine fsetup
*******************************************************************************
* Updates:                                                                    *
*       07MAR96 with exact asymm Hermite coefs for Maxwellian input           *
*       11MAR96 with exact filtered asym Hermites for beam input              *
*       05AUG96 with exact initial Fourier coefs for f1~cos(kx)               *
*       15AUG96 with numquad filtering for symmetric Hermites                 *
*******************************************************************************
*                                                                      130
      include 'parameter.incl'
      include 'common.incl'
      real*8 storef(0:kmax,2),f0(0:kmax)
*
* Set-up x-u phase space, external fields, and filter
      lx = xmax - xmin
      dx = lx/dfloat(nxp)
      umin = root(0)
      umax = root(nu)
      lu = umax - umin                                                 140
*
* Calculate the dt in terms of plasma periods
      ompe = c0p0
      do 2 is = 1,nspec
```

```
        amp = c0p0
        do 1 ib = 1,nbeam(is)
          amp0(ib,is) = amp0(ib,is)
          amp = amp + amp0(ib,is)
 1      continue
        ompe = ompe + qs(is)**2*amp/(ms(is)*perm0)                    150
 2    continue
      ompe = dsqrt(ompe)
      taupe = twopi/ompe
      print *,'Plasma frequency = ',ompe
      dtime = taupe*dtpper
      print *,'dt = dtpper*tau_pe = ',dtime
*
*********************************************************************
* PHASE SPACE GRID AND EXTERNAL E-FIELD E(x,t=0)                    *
* Allowed constant, ramped, cosine, sine, or complex exponential.   *  160
*********************************************************************
      do i = 0,nxp-1
        x(i) = xmin + dfloat(i)*dx
        if (iedist.eq.0) eext(i) = ampe*dcmplx(c1p0,c0p0)
        if (iedist.eq.1) eext(i) = ampe*dcmplx(x(i),c0p0)
        if (iedist.eq.2) eext(i) = ampe*dcos(hek*twopi*x(i)/lx)
        if (iedist.eq.3) eext(i) = ampe*dsin(hek*twopi*x(i)/lx)
        if (iedist.eq.4) eext(i) = ampe*cdexp(ei*hek*twopi*x(i)/lx)
      enddo
*                                                                     170
*********************************************************************
* DISTRIBUTION SET-UP MENU:                                         *
*       ifdist = 0  arbitrary f(x,u,t=0)                            *
*       ifdist = 1  u-Maxwellian beam(s)                            *
*       ifdist = 2  u-Maxwellian(s) with arbitrary x-dependence     *
* NOTE: The option of analytic prescription of the Hermite coefs    *
* f(n) for a sum of Maxwellian beams is given by selecting          *
* -1 or -2.  Various x-perturbations are always allowed and         *
* turned on/off by non-zero amp1 for the kth beam.                  *
*********************************************************************  180
*
* Start of species loop
      do 100 is = 1,nspec
*
* Set-up filtered velocity grid
      beta0(is) = dsqrt(c1p0 - c2p0*(v0/uscale(is))**2)
      do j = 0,nu
         u(j,is) = root(j)*uscale(is)
      enddo
*                                                                     190
*********************************************************************
* Arbitrary distribution f(x,u,t=0)
      if (ifdist(is).eq.0) then
*       f(ix,iu,is) = ????
        call hermcon(f,'u+')
        call fftcon(f,'x+')
      endif
*
```

```
**********************************************************************
* VELOCITY DEPENDENCE KEY for u-Maxwellian(s)                        *
*       nbeam(is) = # of beams                                       *
*       k th beam parameters:                                        *
*       amp0(k,is) = number density of plasma species "is" [#/m]     *
*       u1(k,is) = relative speed of beam [m/s]                      *
*       u0(k,is) = thermal velocity/spread [m/s] (non-zero!!)        *
**********************************************************************
*
**********************************************************************
* NUMERICAL QUADRATURE EVALUATION OF f(n):                           *
* Calculate f0(u_j,t=0) = sum of Maxwellians                         *
**********************************************************************
*
* Set-up f(u) (and filter via Gauss-Hermite quadrature)
      if (ifdist(is).gt.0) then
         do 10 j = 0,nu
*
* No filtering
         if (ifilt.eq.0) then
            do ib = 1,nbeam(is)
               f0(j) = f0(j) + amp0(ib,is)/(rtpi*u0(ib,is))
     &                 *dexp(-((u(j,is)-u1(ib,is))/u0(ib,is))**2)
            enddo
         endif
*
* With filtering
         if (ifilt.eq.1) then
            do kp = 0,nu
            do ib = 1,nbeam(is)
               f0(j) = f0(j) + amp0(ib,is)/(pi*u0(ib,is))*worku(kp)
     &      *dexp(-((u(j,is)-u1(ib,is)-rt2*v0*root(kp))/u0(ib,is))**2)
            enddo
            enddo
         endif
*
 10      continue
*
* Hermite transform f(u)->f(n)=f(0,n,is)
         do 12 n = 0,nu
         do 11 j = 0,nu
            storef(j,1) = f0(j)*hup(n,j)
            storef(j,2) = c0p0
 11      continue
         call srtsm(nu,storef,f(0,n,is))
 12      continue
         f(0,nu+1,is) = f0d0
*
* Truncate the quadrature errors less than 1e-15*f(0,0,is)
         f00 = f(0,0,is)
         do n = 1,nu
            if (cdabs(f(0,n,is)/f00).lt.1e-15) f(0,n,is) = f0d0
         enddo
      endif
```

```
*
*********************************************************************
* EVALUATION OF f(n) VIA ANALYTIC INTEGRATION                       *
* Calculate Hermite coefficients f(*,n,t=0), given an initial sum of *
* Maxwellians, then using filtered asymmetric Hermites with an exact *
* integration formula.  No sym Hermite evaluation to date 8/25/96.  *
*********************************************************************
*                                                                     260
      if (ifdist(is).lt.0.and.iasym.eq.0)
     &   stop 'No exact analytic evaluation of sym H coefs yet.'
*
      if (ifdist(is).lt.0.and.iasym.eq.1) then
         gam0is = uscale(is)*uscale(is) - c2p0*v0*v0
         do ib = 1,nbeam(is)
            gam0 = gam0is - u0(ib,is)*u0(ib,is)
            vd = c0p0
            if (u1(ib,is).ne.c0p0) then
               if (gam0.le.c0p0) stop 'Violated Uscale limit in FSETUP'   270
               vd = u1(ib,is)/dsqrt(gam0)
            endif
            ig = gam0/dabs(gam0)
            gam0 = dsqrt(dabs(gam0))
            do 20 n = 0,nu
               f(0,n,is) = f(0,n,is) + amp0(ib,is)/uscale(is)*rtpi4
     &      *(ig)**(n/2)*(gam0/uscale(is))**dfloat(n)*hermite(n,vd)
 20       continue
         enddo
      endif                                                             280
*
*********************************************************************
* SPATIAL DEPENDENCE KEY for u-Maxwellians(s) (ifdist=+/-2)          *
*       k th beam parameters:                                       *
*       x1(k,is) = offset of spatial nonuniformtiy from zero [m]    *
*       x0(k,is) = spatial width about x1 [m] (non-zero!)           *
*********************************************************************
*
      if (iabs(ifdist(is)).eq.2) then
         do 30 n = 0,nu                                                 290
         do 30 i = 0,nx-1
*
* single-hump cosine shape, positive from [-x0,x0]
c           if (x0(1,is).eq.c0p0) stop 'Zero x0 not allowed!'
c           if (dabs(x(i)-x1(1,is)).le.x0(1,is)) then
c              f(i,j,is) = f(0,j,is)*dcos(c0p5*pi*(x(i)-x1(1,is))/x0(1,is))
c           else
c              f(i,j,is) = f0d0
c           endif
*                                                                       300
* positive-definite cosine
            f(i,n,is) = f(0,n,is)*(c1p0 + dcos(twopi*x(i)/lx))
 30       continue
*
* Fast-Fourier Transform f(x,n,t=0)->f(m,n,t=0)
         call fftcon(f,'x+')
```

```
      endif
*
*************************************************************************
* SPATIAL PERTURBATIONS                                                 *
*       nperb(is) = # perturbation wavenumbers excited, species 'is'    *
*       k th perturbation parameters:                                   *
*       ampl(k,is) = 1st-order amplitude ("epsilon")                    *
*       hk(k,is) = wavenumber of perturbation [unitless] (K=2*pi*hk/Lx) *
*************************************************************************
*
* Add perturbation coefficients exactly w/o FFT, then 2/3-truncate
* to avoid aliasing in the V-shift E*df/du term
*
      do in = 0,nu
         do ip = 0,nperb(is)-1
            f(hk(ip,is),in,is) = -ampl(ip,is)*f(0,in,is)
            f(nxp-hk(ip,is),in,is) = dconjg(f(hk(ip,is),in,is))
         enddo
         if (iasym.eq.1) then
            do im = nx/2,nx-1
               f(im,in,is) = f0d0
            enddo
         endif
      enddo
*
 100  continue
      return
      end
*
* SUBROUTINE FEXACT OMITTED FOR BREVITY
*
*************************************************************************
      subroutine hermset
*************************************************************************
* This routine controls the set-up of the Hermite transforms.           *
*************************************************************************
*
      include 'parameter.incl'
      include 'common.incl'
*
*  Find the maximum root of the H_(nu+1) polynomial, then
*  for all of the roots of H_(nu+1) = 0
      if (iroot.eq.0) then
         call hmaxrt(nu+1)
         print *,'Max Hermite root found.'
         call hroot(nu+1)
         print *,'All Hermite roots found.'
      endif
      if (iroot.eq.1) then
         open (unit=irt,file='root.dat',status='old')
         do j = 0,nu
            read (irt,*) ij,root(j),worku(j)
         enddo
         close (irt)
```

```
      endif
*
*  Set-up exponential integration weights
      do 10 k = 0,nu
         if (iasym.eq.0) rexp(k) = dexp(-root(k)*root(k)/c2p0)
         if (iasym.eq.1) rexp(k) = dexp(-root(k)*root(k))/rtpi4
 10   continue
*
* Set-up Gauss-Hermite weights and Hermite polynomials
      do 50 n = 0,nu                                                    370
         do 30 k = 0,nu
            if (n+iroot.eq.0) then
               h = hermite(nu,root(k))
               worku(k) = c1p0/(h*h*(nu+1))
            endif
            hup(n,k) = worku(k)*hermite(n,root(k))/rexp(k)
            hdown(n,k) = hermite(n,root(k))*rexp(k)
 30      continue
 50   continue
*                                                                       380
      print *,'Hermites calculated.'
      return
      end
*
*************************************************************************
      subroutine hmaxrt(n)
*************************************************************************
* Finds the maximum root of the H_n Hermite polynomial by Newtons's     *
* method, starting from a point just greater than the greatest root     *
* of the previous polynomial H_(n-1). (courtesy J. P. Holloway)         *  390
*************************************************************************
*
      include 'parameter.incl'
      include 'common.incl'
*
* Get all maximum roots up to order n
      reps = 1.0d-5
      iflag = 0
      if (n.eq.0) pause 'H_(N+1) = H_0 has no roots!'
      if (n.eq.1) root(0) = c0p0                                        400
      if (n.gt.1) then
         root(0) = c1p0/rt2
         do 10 k = 1,n-1
            rguess = root(k-1) + 0.1
 1          rstart = rguess
            h = hermite(k,rstart)
            dh = hermite(k-1,rstart)
            rguess = rstart - h/(c2p0*k*dh)
            if (dabs(rguess-rstart).gt.reps) goto 1
            root(k) = rguess                                            410
 10      continue
      endif
*
      return
```

```
      end
*
*************************************************************************
      subroutine hroot(n)
*************************************************************************
* This routine provides the n-roots of an n-th order Hermite           *
* polynomial given the maximum roots in root(n) array.                 *
*************************************************************************
*
      include 'parameter.incl'
      include 'common.incl'
      real*8 ub(1:2*kmax+1)
*
      if (n.eq.0) pause 'No roots of H_0!'
      if (n.eq.1) root(0) = c0p0
      if (n.eq.2) root(0) = -root(1)
*
* Use low order roots as guesses for roots of H_n
      if (n.gt.2) then
* Bracket n roots hrom -(root(n)+1),(root(n)+1)
         nfact = 1
         nroot = 0
         mxroot = root(n-1) + c1p0
 1       delu = (c2p0*mxroot)/dfloat(nfact*n)
         t0 = -mxroot
         do 10 k = 1,nfact*n
            h0 = hermite(n,t0)
            h1 = hermite(n,t0 + delu)
            if (h0*h1.lt.c0p0) then
               nroot = nroot + 1
               ub(nroot) = t0
               ub(nroot+kmax) = t0 + delu
            endif
            t0 = t0 + delu
 10      continue
         if (nroot.lt.n) then
            nfact = nfact*2
            nroot = 0
            goto 1
         endif
*
* Use Bisection to finish up (to machine epsilon, set here)
         reps = 0.25E-15
         call bisection(ub,n,reps)
      endif
*
      return
      end
*
*************************************************************************
      subroutine bisection(xb,n,reps)
*************************************************************************
* This routine uses the false position method within the given          *
* interval [ub(j),ub(j+1)] to find the root of Hermite to within reps.  *
```

```
*************************************************************************
* Input parameters:                                                     *
*              n            # roots, order of Hermite function         *
*              xb(j)        l boundaries of n1 roots                  *
*              xb(j+kmax)   r boundaries of n1 roots(j=1...n1)        *
*              reps         H_n(x) = 0 at x +/- reps                  *
* Output parameters:                                                    *
*              xr(1:n)      n roots of H_n(x)                         *
*************************************************************************
* Warning:  Be certain that only one root is bracketed with each given  *
*           interval [xb(j),xb(j+kmax)].                                *
*************************************************************************
*
      include 'parameter.incl'
      include 'common.incl'
      real*8 xb(1:2*kmax+1)
*
* Check for bracketing of root
      do 10 j = 1,n
         hl = hermite(n,xb(j))
         hr = hermite(n,xb(j+kmax))
         if (hl*hr.gt.c0p0) pause 'Unbracketed.'
 10   continue
*
* Find roots by Bisection
      do 20 j = 1,n
            hj = 9999.d0
            xl = xb(j)
            xr = xb(j+kmax)
            hl = hermite(n,xl)
            hr = hermite(n,xr)
         xweight = 999999.0
 25      if (xweight.gt.reps) then
            xj = c0p5*(xl + xr)
            hj = hermite(n,xj)
            if (hl*hj.le.c0p0) then
               xr = xj
               hr = hj
            endif
            if (hj*hr.le.c0p0) then
               xl = xj
               hl = hj
            endif
            if (hl*hr.gt.c0p0) pause 'Unbracketed in BISECTION.'
            xweight = c2p0*dabs(xl - xr)/dabs(xl + xr)
            goto 25
         endif
         root(j-1) = (xl + xr)*c0p5
 20   continue
*
      return
      end
*
*************************************************************************
```

```
      function hermite(n,v)
*****************************************************************************
* Given n,x, finds H_n(v) using recursion relations, where                  *
* H_n(v) is the symmetric Hermite without the exp term.                     *
*****************************************************************************
*
      include 'parameter.incl'
      include 'common.incl'                                                  530
      real*8 arecur,dfact1,dfact2,v,hermite
*
* Calculate H_n(v)/sqrt(2^n*(n!)*rtpi)
      h0 = c1p0
      h1 = rt2*v
      if (n.eq.0) h=h0
      if (n.eq.1) h=h1
      if (n.gt.1) then
         do 50 k=2,n
            h = (c2p0*v*h1 - dsqrt(c2p0*(k-1))*h0)/dsqrt(c2p0*k)            540
            h0 = h1
            h1 = h
 50      continue
      endif
      hermite = h/rtpi4
*
      return
      end
*
*****************************************************************************  550
      subroutine fftcon(fdum,ifft)
*****************************************************************************
*
      include 'parameter.incl'
      include 'common.incl'
      double complex fdum(0:jmax,0:kmax,nsmax)
      character*2 ifft
*
      ierr = 1
*                                                                             560
* FFT_x f(x_j,*)->f(j,*) with zero-padding of coefficients
      if (ifft.eq.'x+') then
         ierr = 0
         do 10 is = 1,nspec
         do 10 j = 0,nu
            call dcfftf(nxp,fdum(0,j,is),workxp)
            do 15 i = 0,nxp-1
               fdum(i,j,is) = fdum(i,j,is)/dfloat(nxp)
               if (i.ge.nx/2.and.i.le.nx-1) fdum(i,j,is) = f0d0
 15         continue                                                         570
 10      continue
      endif
*
* IFFT_x f(j,*)->f(x_j,*) with zero-padded coefficients
      if (ifft.eq.'x-') then
         ierr = 0
```

```
      do 30 is = 1,nspec
      do 30 j = 0,nu
         call dcfftb(nxp,fdum(0,j,is),workxp)
30    continue                                                          580
      endif
*
      if (ierr.eq.1) stop 'FFTCON: Bad controller.'
*
      return
      end
*
*************************************************************************
      subroutine hermcon(fdum,iht)
*************************************************************************  590
*
      include 'parameter.incl'
      include 'common.incl'
      double complex fdum(0:jmax,0:kmax,nsmax),fold(0:jmax,0:kmax,nsmax)
      real*8 storef(0:kmax,2)
      character*2 iht
*
      ierr = 1
*
* Hermite transform from f(x_j,u_k)->f(x_j,k)                           600
*
      if (iht.eq.'u+') then
         ierr = 0
         call mcopy(nx,nu,nspec,fdum,fold,jmax,kmax,nsmax,icpy)
         do 10 is = 1,nspec
         do 20 n = 0,nu
         do 30 j = 0,nxp-1
            fdum(j,n,is) = f0d0
            if (j.lt.nx/2.or.j.gt.nx-1) then
               do 35 k = 0,nu                                           610
                  storef(k,1) = dreal(fold(j,k,is))*hup(n,k)
                  storef(k,2) = dimag(fold(j,k,is))*hup(n,k)
35             continue
               call srtsm(nu,storef,fdum(j,n,is))
            endif
30       continue
20       continue
10       continue
      endif
*                                                                       620
* Inverse Hermite transform from f(x_j,n)->f(x_j,u_k)
*
      if (iht.eq.'u-') then
         ierr = 0
         call mcopy(nx,nu,nspec,fdum,fold,jmax,kmax+1,nsmax,icpy)
         do 50 is = 1,nspec
         do 60 k = 0,nu
         do 70 j = 0,nxp-1
            fdum(j,n,is) = f0d0
            if (j.lt.nx/2.or.j.gt.nx-1) then                            630
```

```
            do 80 n = 0,nu
               storef(n,1) = dreal(fold(j,n,is))*hdown(n,k)
               storef(n,2) = dimag(fold(j,n,is))*hdown(n,k)
 80          continue
             call srtsm(nu,storef,fdum(j,k,is))
          endif
 70     continue
 60     continue
 50     continue
      endif                                                          640
*
      if (ierr.eq.1) stop 'HERMCON: Bad controller.'
*
      return
      end
*
*************************************************************************
      subroutine srtsm(n,f,fsum)
*************************************************************************
* This routine takes the vector f, sorts in increasing magnitude the   *  650
* positive and negative entries, then sums in order to avoid           *
* round-off.                                                           *
*************************************************************************
*
      include 'parameter.incl'
      real*8 f(0:kmax,2),fpos(0:kmax),fneg(0:kmax),fn,fp,f0d0,sneg(2),spos(2)
      double complex fsum
*
* Process real and imaginary separately
      do 100 ic = 1,2                                                660
*
* Store positive and negatives separately
      ip = -1
      in = -1
      c0p0 = 0.0d0
      fn = c0p0
      fp = c0p0
      do 10 k = 0,n
         if (f(k,ic).lt.c0p0) then
            in = in + 1                                              670
            fneg(in) = f(k,ic)
         else
            ip = ip + 1
            fpos(ip) = f(k,ic)
         endif
 10   continue
*
* Sort Neg's in decreasing order (increasing in magnitude)
  1   iflag = 0
      do 20 k = 1,in                                                 680
         if (fneg(k-1).lt.fneg(k)) then
            fn = fneg(k-1)
            fneg(k-1) = fneg(k)
            fneg(k) = fn
```

```
            iflag = 1
          endif
 20     continue
        if (iflag.eq.1) goto 1
*
* Sort Pos's in increasing order (increasing in magnitude)                    690
  2     iflag = 0
        do 30 k = 1,ip
          if (fpos(k-1).gt.fpos(k)) then
            fp = fpos(k-1)
            fpos(k-1) = fpos(k)
            fpos(k) = fp
            iflag = 1
          endif
 30     continue
        if (iflag.eq.1) goto 2                                               700
*
* Sum them up
        sneg(ic) = c0p0
        do 40 k = 0,in
          sneg(ic) = sneg(ic) + fneg(k)
          fneg(k) = c0p0
 40     continue
        spos(ic) = c0p0
        do 50 k = 0,ip
          spos(ic) = spos(ic) + fpos(k)                                      710
          fpos(k) = c0p0
 50     continue
*
 100    continue
*
        fsum = dcmplx((sneg(1) + spos(1)),(sneg(2) + spos(2)))
*
        return
        end
*                                                                            720
*******************************************************************************
        subroutine mcopy(ni,nj,nk,a,acopy,mmx,nmx,kmx,iflag)
*******************************************************************************
*
        include 'parameter.incl'
        double complex a(0:mmx,0:nmx,kmx),acopy(0:mmx,0:nmx,kmx),
       &               adum(0:jmax,0:kmax)
*
        if (iflag.eq.icpy) then
          do 5 k = 1,nk                                                      730
          do 5 j = 0,nj
          do 5 i = 0,ni
            acopy(i,j,k) = a(i,j,k)
 5        continue
        endif
*
* Switch Left HP<->Right HP so that
* FFT ready [x_1,0 | 0,x_2] -> Mode-matched [0,x_2 | x_1,0] or vice-versa
```

```
*
      if (iflag.eq.ispz) then                                          740
        do 13 k = 1,nk
          do 15 j = 0,nj
          do 15 i = 0,ni
            adum(i,j) = a(i,j,k)
 15       continue
          do 12 j = 0,nj
            do 10 i = 0,ni/2-1
              acopy(i,j,k) = adum(i+ni/2,j)
 10         continue
            do 11 i = ni/2,ni-1                                        750
              acopy(i,j,k) = adum(i-ni/2,j)
 11         continue
 12       continue
 13     continue
      endif
*
      return
      end
*
*************************************************************************  760
      subroutine xshift(dt)
*************************************************************************
*
      include 'parameter.incl'
      include 'common.incl'
      double complex df,fold(0:jmax,0:kmax,nsmax),fmid(0:jmax,0:kmax,2)
*
* Advect f(m,n) distribution using RK4 method
*   Accuracy: O(dt^5) error, Stability: Stable along some of i-axis,
*   Op count: 29/2 NxNuNs                                            770
*
      if (iorder.eq.3.or.iorder.eq.4) then
      call mcopy(nxp,nu,nspec,f,fold,jmax,kmax,nsmax,icpy)
*
      do 30 is = 1,nspec
        do 35 in = 0,nu
        do 35 im = 0,nx/2
          df = falpha1(im,in+1,is)*fold(im,in+1,is)
     &       + cterm(im,in,is)*fold(im,in,is)
     &       + falpha2(im, in ,is)*fold(im,in-1,is)                  780
          fmid(im,in,1) = fold(im,in,is) + c0p5*df
          f(im,in,is) = f(im,in,is) + c1o6*df
 35     continue
        do 40 in = 0,nu
        do 40 im = 0,nx/2
          df = falpha1(im,in+1,is)*fmid(im,in+1,1)
     &       + cterm(im,in,is)*fmid(im,in,1)
     &       + falpha2(im, in ,is)*fmid(im,in-1,1)
          fmid(im,in,2) = fold(im,in,is) + c0p5*df
          f(im,in,is) = f(im,in,is) + c1o3*df                          790
 40     continue
        do 45 in = 0,nu
```

```
      do 45 im = 0,nx/2
        df = falpha1(im,in+1,is)*fmid(im,in+1,2)
     &       + cterm(im,in,is)*fmid(im,in,2)
     &       + falpha2(im, in ,is)*fmid(im,in-1,2)
        fmid(im,in,1) = fold(im,in,is) + df
        f(im,in,is) = f(im,in,is) + c1o3*df
 45   continue
      do 50 in = 0,nu                                                    800
      do 50 im = 0,nx/2
        df = falpha1(im,in+1,is)*fmid(im,in+1,1)
     &       + cterm(im,in,is)*fmid(im,in,1)
     &       + falpha2(im, in ,is)*fmid(im,in-1,1)
        f(im,in,is) = f(im,in,is) + c1o6*df
 50   continue
 30  continue
     endif
*
* Conjugate to fill Fourier space; since f is real,                      810
*     f^m = conj[f^(-m)]
* so only half of calculations are required.  Also, zero-pad the 'middle'.
*
     do 55 is = 1,nspec
     do 55 in = 0,nu
        do im = nx/2,nx-1
          f(im,in,is) = f0d0
        enddo
        do 55 im = nx,nxp-1
          f(im,in,is) = dconjg(f(nxp-im,in,is))                          820
 55  continue
*
     return
     end
*
*************************************************************************
     subroutine ecalc
*************************************************************************
*
     include 'parameter.incl'                                            830
     include 'common.incl'
     double complex fsum
*
*************************************************************************
* Calculate E(x,t0) and J(x,t0) from Poisson's Equation                 *
* (asymmetric Hermite is the default)                                   *
*************************************************************************
     if (iself.eq.1) then
*
* Zero the all initial values                                           840
     do 15 j = 0,nxp-1
       e(j) = f0d0
 15  continue
*
*************************************************************************
* Calculate E(m) coefficients from Poisson's Equation                   *
```

```
* (zero-pad the middle modes nx/2,nx-1 to remove aliasing          *
* errors from non-linear E*df/du multiply in V-shift)              *
*******************************************************************
*                                                                      850
      do 20 is = 1,nspec
      do 20 j = 1,nxp-1
         fsum = f(j,0,is)
         if (iasym.eq.0) then
           fsum = f0d0
           do 25 n = 0,nu
              fsum = fsum + cfact(n,0)*f(j,n,is)
 25        continue
         endif
         e(j) = e(j) + efact(j,is)*fsum                               860
 20   continue
*
* FFT back to x-space E(m)->E(x) onto grid of 3*nx/2 values
      call dcfftb(nxp,e,workxp)
*
      endif
*
*******************************************************************
* External E-field thru iteration numbers ITMIN<=ITER<=ITMAX      *
*******************************************************************  870
      if (ampe.ne.c0p0.and.iter.ge.itmin.and.iter.le.itmax) then
         dxp = cc/(ompe*omega)
         time = iter*dtpper*twopi/ompe
*
* Start the pulse at 'ipulse' widths outside system
         ipulse = 10
         do 100 j = 0,nxp-1
            xp = (x(j) - cc*time + c0p5*lx)/dxp - ipulse
            e(j) =  e(j)*iself + ampe*dexp(-xp**2)/(rtpi*dxp)
 100     continue                                                     880
      endif
*
      return
      end
*
*************************************************************************
      subroutine vshift(dt)
*************************************************************************
* Updates:                                                            *
*       12/6/95 for symmetric Hermites                                *  890
*       2/24/96 with zero-padding for removal of aliasing errors      *
*************************************************************************
*
      include 'parameter.incl'
      include 'common.incl'
      double complex df1,df2,df3,df4,fold(0:jmax,0:kmax,nsmax),fmid(0:jmax,0:kmax,3)
*
* Copy and IFFT f(j,n,is)->f(x_j,n,is)
      call fftcon(f,'x-')
      call mcopy(nxp,nu,nspec,f,fold,jmax,kmax,nsmax,icpy)            900
```

```
*
*************************************************************************
* SYMMETRIC HERMITE V-SHIFT using                                       *
* df_n(x,t)/dt = q/(mU)*E(x,t)*[sqrt(n+1)f_n - sqrt(n)f_(n-1)]          *
*************************************************************************
*
      if (iasym.eq.0) then
      if (iorder.ne.4)
     &   stop 'No low-order V-shift for symH.'
*                                                                        910
* Runge-Kutta 4 Method: Error, O(dt^5); Ops, 33NxNuNs
      if (iorder.eq.4) then
      do 10 is = 1,nspec
         do 11 in = 0,nu
         do 11 j = 0,nxp-1
            df = (bfact(in+1,is)*fold(j,in+1,is) - bfact(in,is)*fold(j,in-1,is))*e(j)
            fmid(j,in,1) = fold(j,in,is) - c0p5*df
            f(j,in,is) = f(j,in,is) - c1o6*df
 11      continue
         do 12 in = 0,nu                                                920
         do 12 j = 0,nxp-1
            df = (bfact(in+1,is)*fmid(j,in+1,1) - bfact(in,is)*fmid(j,in-1,1))*e(j)
            fmid(j,in,2) = fold(j,in,is) - c0p5*df
            f(j,in,is) = f(j,in,is) - c1o3*df
 12      continue
         do 13 in = 0,nu
         do 13 j = 0,nxp-1
            df = (bfact(in+1,is)*fmid(j,in+1,2) - bfact(in,is)*fmid(j,in-1,2))*e(j)
            fmid(j,in,1) = fold(j,in,is) - df
            f(j,in,is) = f(j,in,is) - c1o3*df                           930
 13      continue
         do 14 in = 0,nu
         do 14 j = 0,nxp-1
            df = (bfact(in+1,is)*fmid(j,in+1,1) - bfact(in,is)*fmid(j,in-1,1))*e(j)
            f(j,in,is) = f(j,in,is) - c1o6*df
 14      continue
 10   continue
      endif
      endif
*                                                                        940
*************************************************************************
* ASYMMETRIC HERMITE V-SHIFTS using                                     *
* df_n(x,t)/dt = sqrt(2n)*q/(mU)*E(x,t)*f_(n-1)(x,t)                    *
*************************************************************************
*
      if (iasym.eq.1) then
*
* Runge-Kutta 2 Method: Error, O(dt^3); Ops, 7NxNuNs
      if (iorder.eq.2) then
         do 20 is = 1,nspec                                             950
         do 20 in = 0,nu
         do 20 j = 0,nxp-1
            f(j,in,is) = fold(j,in,is) + bfact(in,is)*e(j)
     &   *(fold(j,in-1,is) + c0p5*bfact(in-1,is)*e(j)*fold(j,in-2,is))
```

```
 20      continue
      endif
*
* Crank-Nicholson 2 Method: Error, O(dt^3); Ops: 5NxNuNs
      if (iorder.eq.3) then
         do 21 is = 1,nspec                                                    960
         do 21 in = 0,nu
         do 21 j = 0,nxp-1
            f(j,in,is) = fold(j,in,is) + c0p5*bfact(in,is)*e(j)*(fold(j,in-1,is) + f(j,in-1,is))
 21      continue
      endif
*
* Runge-Kutta 4 Method: Error, O(dt^5); Ops, 21NxNuNs
      if (iorder.eq.4.or.iorder.eq.5) then
         do 25 is = 1,nspec
         do 30 in = 0,nu                                                       970
         do 30  j = 0,nxp-1
            df1 = bfact(in,is)*e(j)*fold(j,in-1,is)
            fmid(j,in,1) = fold(j,in,is) + c0p5*df1
            f(j,in,is) = f(j,in,is) + c1o6*df1
 30      continue
         do 31 in = 0,nu
         do 31  j = 0,nxp-1
            df2 = bfact(in,is)*e(j)*fmid(j,in-1,1)
            fmid(j,in,2) = fold(j,in,is) + c0p5*df2
            f(j,in,is) = f(j,in,is) + c1o3*df2                                 980
 31      continue
         do 32 in = 0,nu
         do 32  j = 0,nxp-1
            df3 = bfact(in,is)*e(j)*fmid(j,in-1,2)
            fmid(j,in,1) = fold(j,in,is) + df3
            f(j,in,is) = f(j,in,is) + c1o3*df3
 32      continue
         do 33 in = 0,nu
         do 33  j = 0,nxp-1
            df4 = bfact(in,is)*e(j)*fmid(j,in-1,1)                             990
            f(j,in,is) = f(j,in,is) + c1o6*df4
 33      continue
 25      continue
      endif
*
      endif
*
* FFT f(x_j,n)->f(m,n) and re-zero pad
      call fftcon(f,'x+')
*                                                                            1000
      return
      end
*
*******************************************************************************
      subroutine writer
*******************************************************************************
*
      include 'parameter.incl'
```

```
      include 'common.incl'
      double complex fdum(0:jmax,0:kmax+1,nsmax)                        1010
*
*********************************************************************
* Open FOUT.DAT and EOUT.DAT                                        *
*********************************************************************
*
      if (iter.eq.0) then
        open (unit=ifout1,file='fmn.dat',status='unknown')
        rewind ifout1
        write (ifout1,109) nx,nu,(niter/iof)
        open (unit=ieout1,file='eout.dat',status='unknown')             1020
        rewind ieout1
        write (ieout1,109) nx,nu,(niter/ioe)
      endif
*
*********************************************************************
* ITRANS=-1 Write ln |f(m,n,t)| -> FOUT.DAT                         *
*********************************************************************
*
      if (mod(iter,iof).eq.0.and.itrans.eq.-1) then
        write (*,*) 'Writing record#',iter/iof,' to FOUT.'              1030
*
* Save [m,n,ln |f(m,n,is)|] to FOUT.DAT
        do 1 in = 0,nu
        do 2 im = 0,nx/2-1
          if (cdabs(f(im,in,1)).eq.f0d0) then
            dfabs = -100.d0
          else
            dfabs = sngl(log(cdabs(f(im,in,1))))
          endif
          write (ifout1,115) im,in,dfabs                                1040
 2      continue
        write (ifout1,120)
 1      continue
      endif
*
*********************************************************************
* ITRANS=0  Write f(m,n,t) -> FOUT.DAT                              *
*********************************************************************
*
      if (mod(iter,iof).eq.0.and.itrans.eq.0) then                      1050
        write (*,*) 'Writing record#',iter/iof,' to FOUT.'
*
* Save [m,n,f(m,n,is)] to FOUT.DAT
        do 10 in = 0,nu
        do 11 im = 0,nx-1
          write (ifout1,115) im,in,
     &      (sngl(dreal(f(im,in,is))),sngl(dimag(f(im,in,is))),
     &       sngl(dreal(fex(im,in,is))),sngl(dimag(fex(im,in,is))),
     &       is=1,nspec)
 11     continue                                                        1060
        write (ifout1,120)
 10     continue
```

```
      endif
*
*********************************************************************
* ITRANS=1  Transform and write f(x,u,t) -> FOUT.DAT                 *
*********************************************************************
*
* Copy and inverse transform fdum(m,n)->fdum(x,u), then save
*                                                                    1070
      if (mod(iter,iof).eq.0.and.itrans.eq.1) then
        write (*,*) 'Writing record#',iter/iof,' to FOUT.'
        call mcopy(nx,nu,nspec,f,fdum,jmax,kmax+1,nsmax,icpy)
        call fftcon(fdum,'x-')
        call hermcon(fdum,'u-')
        call fftcon(fex,'x-')
        call hermcon(fex,'u-')
*
        do 20 j = 0,nu,nustep
        do 21 i = 0,nx-1,nxstep                                        1080
          write (ifout1,112) sngl(x(i)),(sngl(u(j,is)),sngl(dreal(fdum(i,j,is))),
     &          sngl(dreal(fex(i,j,is))),is=1,nspec)
 21     continue
        write (ifout1,120)
 20     continue
      endif
*
*********************************************************************
* Transform and write E(x,t) -> EOUT.DAT                             *
*********************************************************************  1090
*
* Save 0:[t,E(x_o,t)] or 1:[x,E(x_i)] to EOUT.DAT
*
      if (mod(iter,ioe).eq.0) then
        write (*,*) 'Writing record#',iter/ioe,' to EOUT.'
        if (itorx.eq.0) then
*
          emode = f0d0
          do is = 1,nspec
            k = hk(0,is)                                               1100
            emode = emode + efact(k,is)*f(k,0,is)
          enddo
          t = iter*dtpper
          write (ieout1,111) sngl(t),sngl(dreal(emode)),sngl(dimag(emode))
        endif
        if (itorx.eq.1) then
          write (ieout1,110) (sngl(x(i)),sngl(dreal(e(i))),i=0,nx-1)
          write (ieout1,120)
        endif
      endif                                                            1110
*
*********************************************************************
* On last iteration, close-up shop.                                  *
*********************************************************************
*
      if (iter.eq.niter) then
```

```
        write (*,*) 'Finished writing output to FOUT, EOUT.'
        write (*,*) 'SPLIT information:'
        close (ifout1)
        if (itrans.eq.1) numf1 = (nx/nxstep + 1)*((nu+1)/nustep + 1)
        if (itrans.eq.0) numf1 = (nx + 1)*(nu + 2)
        if (itrans.eq.-1) numf1 = (nx/2 + 1)*(nu + 2)
        write (*,130) numf1,' fout.dat fout'
        close (ieout1)
        numel = nx + 1
        write (*,130) numel,' eout.dat eout'
      endif
*
*****************************************************************
* Format specifications                                         *
*****************************************************************
*
 109  format ('#',3(i4,2x))
 110  format (2(e15.8,2x))
 111  format (3(e15.8,2x))
 112  format (e15.8,2x,6(e15.8,2x))
 113  format (2(e15.8,2x))
 114  format (e10.4,2x,e15.8)
 115  format (i4,i4,6(e15.8,2x))
 120  format (1x)
 130  format (1x,'Use split -',i4,a16)
*
      return
      end
*
*************************************************************************
      subroutine monitor
*************************************************************************
* This unit controls the output of various quantities of interest:      *
* Into the file MONOUT.DAT, every iomon time-steps:                     *
*       t [ns], t/Tau, n [#/m], p [kg m/s], ke [J], pe [J], H [J],      *
*       dH/H0, dp/p0, d<f,f>/<f0,f0>                                    *
* Into the file FUOUT.DAT, every iofu time-steps:                       *
*       n  u(n)  f(u(n))  fdum(u(n))  f(k1,n)  f(k2,n)                  *
* Into the file FMN.OUT, every iofu time-steps:                         *
*       m  f(m,N)  f(m,N-1)  f(m,N-2)  f(m,N-3)  f(m,N-4)               *
* Into the file EHIST.DAT, every ioehst time-steps:                     *
*       t/Tau  E(k1)  E(k2)  E(k3) . . .                                *
*************************************************************************
*
      include 'parameter.incl'
      include 'common.incl'
      double complex fdum(0:jmax,0:kmax+1,nsmax),em(0:jmax),fu(0:kmax,nsmax)
      real*8 storef(0:kmax,2)
*
*****************************************************************
* Open output files                                             *
*****************************************************************
*
      if (iter.eq.0) then
```

```
      open (unit=imout,file='monout.dat',status='unknown')
      rewind imout
      write (imout,97)
 97   format ('#',4x,'time',7x,'t/tau',3x,'particles',3x,'momentum',3x,'k energy',
     &       3x,'f energy',3x,'t energy',6x,'herror',6x,'perror',6x,'<f,f> error')
      open (unit=ifout2,file='fuout.dat',status='unknown')
      rewind ifout2
      write (ifout2,96) nu+1,(niter/iofu),nspec
 96   format ('#',3(i4,2x))
      open (unit=18,file='fmN.dat',status='unknown')                          1180
      open (unit=ieout2,file='ehist.dat',status='unknown')
      rewind ieout2
      write (ieout2,98) hk(0,1),lx
 98   format ('#',2(f8.4,f11.7))
      write (ieout2,99) (hk(k,1),k=0,nperb(1)-1)
 99   format ('# time',13x,20('k=',f3.0,11x))
    endif
*
*********************************************************************
* Conservation and L_2 monitoring -> MONOUT.DAT                     *   1190
*********************************************************************
*
    time = dfloat(iter)*dtime
    tovtau = time/taupe
    if (mod(iter,iomon)*mod(iter,ioehst)*mod(iter,iofu).eq.0) then
      write (6,101) iter,time,tovtau
 101  format (1x,'Iter = ',i6,5x,'t= ',d10.4,' sec',5x,'t/tau= ',d10.4)
    endif
*
    if (iter.eq.0.or.mod(iter,iomon).eq.0) then                           1200
*
*  Calculate number of particles (#)
    fparts = f0d0
    do 5 is = 1,nspec
    if (iasym.eq.0) then
      do n = 0,nu
        fparts = fparts + lx*uscale(is)*f(0,n,is)*cfact(n,0)
      enddo
    endif
    if (iasym.eq.1) fparts = fparts + lx*uscale(is)*f(0,0,is)             1210
 5  continue
*
* Calculate particle momentum (kg m/s)
    fmom = f0d0
    do 10 is = 1,nspec
    if (iasym.eq.0) then
      do n = 0,nu
        fmom = fmom + lx*ms(is)*uscale(is)**2*f(0,n,is)*cfact(n,1)
      enddo
    endif                                                                 1220
    if (iasym.eq.1) fmom = fmom + lx*ms(is)*uscale(is)**2*f(0,1,is)/rt2
 10 continue
*
* Calculate the kinetic energy [J] and thermal velocity from the temperature
```

```
      fener = f0d0
      deblen = c0p0
      do 15 is = 1,nspec
         if (iasym.eq.0) then
            fsum = f0d0
            do 12 n = 0,nu                                              1230
               fsum = fsum + f(0,n,is)*(cfact(n,2) + cfact(n,3))
 12         continue
            fener = fener + c0p5*ms(is)*(lx*uscale(is)**3*fsum - v0**2*fparts)
         endif
         if (iasym.eq.1) then
            fener = fener + lx*ms(is)*uscale(is)**3/4.0d0
     &              *(beta0(is)**2*f(0,0,is) + rt2*f(0,2,is))
         endif
         deblen = deblen + dreal(fener)/ms(is)
 15   continue                                                          1240
      deblen = dsqrt(lx*deblen/(c2p0*dreal(fparts)))
*
* Calculate the electric field energy [J]
      eener = f0d0
      do 20 is = 1,nspec
      do 20 j = 1,nxp-1
         em(j) = f0d0
         fsum = f(j,0,is)
         if (iasym.eq.0) then
            fsum = f0d0                                                 1250
            do n = 0,nu
               fsum = fsum + cfact(n,0)*f(j,n,is)
            enddo
         endif
         em(j) = em(j) + efact(j,is)*fsum
         eener = eener + c2p0*em(j)*dconjg(em(j))
 20   continue
      eener = c0p5*perm0*eener
*
* Total energy = KE + PE                                                1260
      fpe = fener + eener
*
* Calculate the Debye length from the thermal velocity and ompe
      if (iter.eq.0) then
         deblen = deblen/ompe
         print *,'approximate Debye length [m] = ',deblen
         pparam = fparts*deblen
         print *,'Plasma parameter [#/Debye sheet] = ',pparam
      endif
*                                                                       1270
*************************************************************************
* Store initial conditions for comparison and check stopping           *
* criterion for "instability"                                          *
*************************************************************************
*
      if (iter.eq.0) then
         part0 = dreal(fparts)
         mom0 = dreal(fmom)
```

```
         tener0 = dreal(fpe)
      endif                                                              1280
      if (icalex.eq.0) then
         parerr = dabs(dreal(fparts) - part0)/part0
         herr = dabs(dreal((fpe - tener0)/tener0))
         if (dabs(herr).ge.0.1d0) stop 'Exceeded dH/H0 > 10% conservation limit!'
         if (mom0.eq.c0p0) then
            perr = dreal(fmom)
         else
            perr = dreal((fmom - mom0)/mom0)
         endif
      endif                                                              1290
*
* Calculate ||fex(x,u,t)-f(x,u,t)||_2
      fdiff = f0d0
      fnorm = f0d0
      if (icalex.eq.1) then
         call fexact
         do 32 is = 1,nspec
         do 31 k = 0,nu
            do 30 j = 0,nxp-1
               fdum(j,k,is) = fex(j,k,is) - f(j,k,is)                    1300
               fnorm = fnorm + fex(j,k,is)*dconjg(fex(j,k,is))
               fdiff = fdiff + fdum(j,k,is)*dconjg(fdum(j,k,is))
 30         continue
 31      continue
 32      continue
         if (fnorm.eq.f0d0) fnorm = dcmplx(c1p0,c0p0)
         fdiff = cdsqrt(fdiff/fnorm)
      endif
*
************************************************************************ 1310
* Calculate the Integral f*f                                           *
************************************************************************
*
* For symmetric Hermites, the sum of squared coefficients...
      if (iasym.eq.0) then
      do 33 is = 1,nspec
      do 33 in = 0,nu
         ff = f0d0
         do im = 0,nxp-1
            ff = ff + f(im,in,is)*dconjg(f(im,in,is))                  1320
         enddo
         storef(in,1) = lx*uscale(is)*dreal(ff)
         storef(in,2) = c0p0
 33   continue
      call srtsm(nu,storef,ff)
      endif
*
* For asymmetric Hermites, a bit more complicated...
      if (iasym.eq.1) then
         ff = f0d0                                                       1330
         do 34 is = 1,nspec
         do 34 im = 0,nxp-1
```

```
            do n1 = 0,nu
               fsn = f0d0
               do n2 = 0,nu
                  fsn = fsn + coef(n1,n2)*dconjg(f(im,n2,is))
               enddo
               storef(n1,1) = dreal(fsn*f(im,n1,is))
               storef(n1,2) = dimag(fsn*f(im,n1,is))
            enddo                                                    1340
            call srtsm(nu,storef,fsn)
            ff = ff + lx*uscale(is)*fsn
 34       continue
      endif
*
* Calculate the error in <f,f>
      if (iter.eq.0) ff0 = ff
      ff = ff/ff0 - c1p0
*
*************************************************************************  1350
* Write conservation diagnostics to file                                 *
*************************************************************************
*
      if (icalex.eq.1) herr = fdiff
      write (imout,100) time,tovtau,sngl(real(fparts)),sngl(dreal(fmom)),
     &  sngl(dreal(fener)),sngl(dreal(eener)),sngl(dreal(fpe)),sngl(herr),sngl(perr),
     &  sngl(cdabs(ff)),sngl(parerr)
 100  format (7(e11.4), 4(1x,e13.6))
*
      endif                                                          1360
*
*************************************************************************
* Calculate f(u),f1(u),f(k,n) -> FUOUT.DAT                             *
*************************************************************************
*
      if (iter.eq.0.or.mod(iter,iofu).eq.0) then
*
* Write f(m,N-1),f(m,N) modes to fmN.out
        do 35 im = nx,nxp-1
          write (18,113) (im-nx),sngl(cdabs(f(im,nu,1))),sngl(cdabs(f(im,nu-1,1))),   1370
     &   sngl(cdabs(f(im,nu-2,1))),sngl(cdabs(f(im,nu-3,1))),sngl(cdabs(f(im,nu-4,1)))
 35     continue
        do 36 im = 0,nx/2-1
          write (18,113) im,sngl(cdabs(f(im,nu,1))),sngl(cdabs(f(im,nu-1,1))),
     &   sngl(cdabs(f(im,nu-2,1))),sngl(cdabs(f(im,nu-3,1))),sngl(cdabs(f(im,nu-4,1)))
 36     continue
*
        write (18,120)
        do 40 is = 1,nspec
        do 40 k = 0,nu                                               1380
          do 41 n = 0,nu
            storef(n,1) = dreal(f(0,n,is))*hdown(n,k)
            storef(n,2) = dimag(f(0,n,is))*hdown(n,k)
 41       continue
          call srtsm(nu,storef,fu(k,is))
 40     continue
```

```
      do 45 j = 0,nu
         write (ifout2,113) j,(sngl(u(j,is)),sngl(dreal(fu(j,is))),
     &                  sngl(cdabs(f(hk(0,is),j,is))),is=1,nspec)
 113     format (i3,3x,18(e15.8,2x))                                    1390
 45    continue
       write (ifout2,120)
     endif
*
*********************************************************************
* Calculate E_k amplitudes -> EHIST.DAT                             *
*********************************************************************
*
     if (iter.eq.0.or.mod(iter,iochst).eq.0) then
       do 25 j = 1,3*nperb(1),3                                       1400
         emhist(j-1) = cdsqrt(em(hk(j-1,1))*dconjg(em(hk(j-1,1))))
         emhist(j) = dabs(dreal(em(hk(j-1,1))))
         emhist(j+1) = dabs(dimag(em(hk(j-1,1))))
 25    continue
       write (ieout2,102) tovtau,(sngl(dreal(emhist(im))),im=0,3*nperb(1)-1)
 102   format (e13.6,21(2x,e14.7))
     endif
*
*********************************************************************
* Close-up shop at end of run.                                      *  1410
*********************************************************************
*
     if (iter.eq.niter) then
       write (*,*) 'Finished writing output to MONOUT, FUOUT, EHIST.'
       close (imout)
       close (ieout2)
       close (ifout2)
       numf2 = (nu + 1) + 1
       write (*,130) numf2,' fuout.dat fuout  '
       write (*,120)                                                  1420
 120   format (1x)
 130   format (1x,'Use split -',i4,a16)
     endif
*
     return
     end

-------------------------------------------------------------------
-------------------------------------------------------------------
                                                                    1430
*********************************************************************
*            PARAMETER.INCL for VMS_FH routines                     *
*********************************************************************
*
     parameter (ifin=11,ifout1=12,ifout2=13,ieout1=14,ieout2=15,
    &           imout=16,irt=17,icpy=0,ispx=1,jmax=192,kmax=512+1,
    &           nbmax=2,npmax=6,nsmax=1,ntmax=10,nwk=4*jmax+15)
*
*********************************************************************
*            COMMON.INCL for VMS_FH routines                        *  1440
```

```
*******************************************************************
*
* Implicit statements
      implicit real*8 (a-d,g-h,l,m,o-z)
      implicit integer (i,j,k,n)
      implicit double complex (e-f)
*
* Common arrays and constants
      common /grids/ xmin,xmax,umin,umax,lx,lu,nx,nxp,nxstep,nu,nustep
      common /phspc/ x(0:jmax),u(0:kmax,nsmax)                                  1450
      common /ffts/ workx(nwk),workxp(nwk)
      common /hts/ root(0:kmax),worku(0:kmax),rnorm(-1:1),rexp(0:kmax),
     &         hup(0:kmax,0:kmax),hdown(0:kmax,0:kmax),iasym,iroot
      common /time/ dtime,dtpper,ompe,taupe,niter,iter
      common /filter/ ufilt(0:kmax,0:kmax,nsmax),v0,beta0(nsmax),collf,ifilt
      common /species/ qs(nsmax),ms(nsmax),uscale(nsmax),nspec
      common /fnumer/ f(0:jmax,0:kmax,nsmax),iorder
      common /fanaly/ fex(0:jmax,0:kmax,nsmax),icalex
      common /distr1/ amp0(nbmax,nsmax),amp1(0:npmax,nsmax),x0(nbmax,nsmax),
     &         x1(nbmax,nsmax),u0(nbmax,nsmax),u1(nbmax,nsmax),hk(0:npmax,0:nsmax),   1460
     &         ifdist(nsmax),nbeam(nsmax),nperb(nsmax)
      common /iostuff/ iof,ioe,iomon,iofu,ioehst,itest,itrans,ipost
      common /cnsts/ f0d0,ei,perm0,pi,twopi,rtpi,rtpi4,rt2,c0p0,c0p5,c1p0,c2p0,
     &         c3p0,c6p0,c1o6,c1o3
      common /cutoff/ cut0,cutm,iclose
      common /dgnstc/ ff0,part0,mom0,tener0
      common /shifts/ falpha1(0:jmax,0:kmax+1,nsmax),falpha2(0:jmax,0:kmax+1,nsmax),
     &         cfilt1(nsmax),filt2(nsmax),cterm(0:jmax,0:kmax,nsmax),
     &         bfact(-1:kmax+1,nsmax)
      common /eextern/ eext(0:jmax),ampe,hek,omega,itmin,itmax,itorx,iedist     1470
      common /efields/ e(0:jmax),ejcurr(0:jmax),emhist(0:3*npmax),efact(0:jmax,nsmax),
     &         iself
      common /symint/ cfact(0:kmax,0:3)
      common /asymff/ coef(0:kmax,0:kmax)
      common /facts/  afact(0:kmax),iinit
```

---VMSFH Input Deck 9/09/96----Bump-on-tail distribution----- 1480

PHASE SPACE (grids and dimensions)

nx	nu	nxp	nup
64	64	2	8

xmin(m)	xmax(m)
-0.5	0.5

iasym (HERMITES: 0=sym, 1=asym)	iroot (1=load root.dat)
1	1

TIMING & OUTPUT SWITCHES 1490

iorder	dt/tau	niter		
4	1.0e-3	25000		
iof	ioe	iomon	iofu	ioehst
1000000	1000000	10	125	10

itest	itrans	ipost
0	0	0

PLASMA PARAMETERS

nspecies

1 1500

SPECIES1	mass(kg)	charge(C)	uscale(m/s)
	9.10953d−31	−1.60219e−19	2.0e7

Beam & Perturbation Parameters

ifdist	nbeam	nperb
−1	2	1

ibeam1	amp0	x0(m)	x1(m)	u0(m/s)	u1(m/s)
	5.0e14	0.0	0.0	1.32619e7	0.0e7
ibeam2	amp0	x0(m)	x1(m)	u0(m/s)	u1(m/s)
	1.0e14	0.0	0.0	1.32619e7	5.0e7
iperb1	amp1	hk(1,ispecies)			
	1.0e−5	5			

1510

FILTERING

ifilt	v0(m/s)	collf
0	0.0e7	0.0e3

INTERNAL & EXTERNAL E−FIELD / EXACT SOLUTION MONITOR

iselfconsist	icalex (0=off, 1=on)
1	0

1520

iedist	ampe	omega	hk(0,0)	itmin	itmax	time/x data
0	0.0e4	0.0d9	0.0d0	0	100000000	0

PHYSICAL CONSTANTS

permittivity of free space (F/m)

8.854187818d−12

BIBLIOGRAPHY

BIBLIOGRAPHY

[Abr.1] Abramowitz, Milton and Stegun, Irene A., **Handbook of Mathematical Functions.** Dover Publications, Inc. New York, 10th printing, December 1972, pp. 773-802.

[Arm.1] Armstrong, Thomas P., "Numerical Studies of the Nonlinear Vlasov Equation," Physics of Fluids, vol. 10, June 1967, pp. 1269-1280.

[Arm.2] Armstrong, Thomas and Montgomery, David, "Asymptotic state of the two-stream instability," Plasma of Physics, Vol 1, No. 4, 1967, pp. 425-433.

[Arm.3] Armstrong, T. P. and Montgomery, D., "Numerical Study of Weakly Unstable Electron Plasma Oscillations," Physics of Fluids, vol. 12, 1969, pp. 2094-2098.

[Ber.1] Bernstein, I. B., Greene, J. M., and Kruskal, M. D., "Exact Nonlinear Plasma Oscillations," Physical Review, vol. 108, November 1957, pp. 546-550.

[Ber.2] Berk, Herbert L. and Roberts, Keith V., "The Water-Bag Model," Methods In Computational Physics, Volume 9 (Plasma Physics), Academic Press, Inc., January 1970, pp. 88-133.

[Bir.1] Birdsall, Charles K., Langdon, A. Bruce, and Okuda, H., "Finite-sized particle physics applied to plasma simulation," Methods In Computational Physics, Volume 9 (Plasma Physics), Academic Press, Inc., January 1970, pp. 241-257.

[Bir.2] Birdsall, C. K. and Langdon, A. B., **Plasma Physics Via Computer Simulation.** IOP Publishing Ltd, 1991.

[Boy.1] Boyd, John P., "Asymptotic coefficients of Hermite Function Series," Journal of Computational Physics, vol. 54, 1984, pp. 382-410.

[Bye.1] Byers, Jack A. and Killeen, John, "Finite-difference methods for collisionless plasma models," Methods In Computational Physics, Volume 9 (Plasma Physics), Academic Press, Inc., January 1970, pp. 259-305.

[Can.1] Canuto, C., Hussaini, M. Y., Quarteroni, A., Zang, T. A., **Spectral Methods in Fluid Dynamics.** Berlin: Springer-Verlag, 1988.

[Che.1] Chen, Francis F., **Plasma Physics and Controlled Nuclear Fusion, Volume 1: Plasma Physics.** Plenum Press, 2nd edition, 1984, pp. 225-285.

[Che.1] Cheng, C. Z. and Knorr, Georg, "The Integration of the Vlasov Equation in Configuration Space," Journal of Computational Physics, vol. 22, 1976, pp. 330-351.

[Dav.1] Davidson, R. C., **Methods in Nonlinear Plasma Theory.** New York: Academic Press, 1972.

[Daw.1] Dawson, J. M., "One-dimensional plasma model," Physics of Fluids, vol. 5, April 1962, pp. 445-459.

[Daw.2] Dawson, John M., "The electrostatic sheet model for a plasma and its implications to finite-sized particles," Methods In Computational Physics, Volume 9 (Plasma Physics), Academic Press, Inc., January 1970, pp. 1-27.

[Der.1] Derrick, William R., **Complex Analysis and Applications.** 2nd edition, Brooks/Cole Publishing Company, 1983, pp. 152-184.

[Dem.1] Demeio, Lucio and Holloway, James P. "Numerical simulations of BGK modes," Journal of Plasma Physics, vol. 46, 1991, pp. 63-84.

[Den.1] Denavit, J. "Simulations of the single-mode, bump-on-tail instability," Physics of Fluids, vol. 28, No. 9, September 1985, pp. 2773-2777.

[Dur.1] Durran, Dale R. "The Third-Order Adams-Bashforth Method: An Attractive Alternative to Leapfrog Time Differencing," Monthly Weather Review, vol. 119, March 1991, pp. 702-720.

[Eng.1] Engelmann, F., Fiex, M., Mindardi, E., and Oxenius, J., "Nonlinear Effects from Vlasov's Equation," Physics of Fluids, vol. 6, 1963, pp. 267-275.

[Fri.1] Frieman, E. and Rutherford, P. "Kinetic Theory of a Weakly Unstable Plasma," Annals of Physics, vol 28, 1964, pp. 134-177.

[For.1] Forest, Etienne and Ruth, Ronald D.,"Fourth-Order Symplectic Integration," Physica D, vol. 43, 1990, pp. 105-117.

[Gar.1] Gardner, L. R. T. and Gardner, G. A. "A locally one dimensional space time finite element method," **Numerical Methods for Nonlinear Problems**, Proceedings of the International Conference held at Dubrovnik, Yugoslavia, 1986, pp. 813-824.

[Ghi.1] Ghizzo A., Izrar, B., Bertrand, P., Fijalkow, E., Feix, M. R.,and Shoucri, M., "Stability of Bernstein-Greene-Kruskal plasma equilibria. Numerical experiments over a long time," Physics of Fluids, vol. 31, No. 1, 1988, pp. 72-82.

[Ghi.2] Ghizzo A., Reveille, T., B., Bertrand, Johnston, T. W., Lebas, J., and Shoucri, M., "An Eulerian-Hilbert Code for the Numerical Simulation of the Interaction of High-Frequency Electromagnetic Waves with Plasma," Journal of Computational Physics, vol 118, 1995, pp. 356-365.

[Gra.1] Grant, Frederick C., and Feix, Marc R.,"Fourier-Hermite Solutions of the Vlasov Equation in the Linearized Limit," Physics of Fluids, vol. 10, No. 4, April 1967, pp. 696-702.

[Gra.2] Gradshteyn, I. S. and Ryzhik, I. M., **Table of Integrals, Series, and Products.** Academic Press, Inc., 5^{th} edition, 1994, pp. 842-1059.

[Gol.1] Goldstein, Herbert, **Classical Mechanics.** 2nd edition, Addison-Wesley Publishing Company, 1980.

[Har.1] Harding, Rollin C., "Response of a One-Dimensional Vlasov Plasma to External Electric Fields," Physics of Fluids, Vol 11, October 1968, pp. 2233-2240

[Hol.1] Holloway, J. P. and Dorning, J. J. "Undamped Longitudinal Plasma Waves," Physics Letters A, vol. 138, 1989, pp. 279-284.

[Hol.2] Holloway, J. P., "Spectral Velocity Discretizations for the Vlasov-Maxwell Equations," Transport Theory and Statistical Physics, vol. 25, 1996, pp. 1-32.

[Hol.3] Holloway, J. P. "On Numerical Methods for Hamiltonian PDEs and a Collocation Method for the Vlasov-Maxwell Equations," Journal of Computational Physics, vol. 129, 1996, pp. 121-133.

[Joy.1] Joyce, Glenn, Knorr, Georg, and Meier, Homer K., "Numerical Integration Methods of the Vlasov Equation," Journal of Computational Physics, vol. 8, 1971, pp. 53-63.

[Kil.1] Killeen, John and Marx, Kenneth D. "The solution of the Fokker-Planck equation for a mirror-confined plasma," Methods In Computational Physics, Volume 9 (Plasma Physics), Academic Press, Inc., January 1970, pp. 422-489.

[Kli.1] Klimas, A. J., "Vlasov-Maxwell and Vlasov-Poisson equations as models of a one-dimensional electron plasma," Physics of Plasmas, vol. 26, No. 2, February 1983, pp. 478-479.

[Kli.2] Klimas, A. J., "A Method for Overcoming Velocity Space Filamentation Problem in Collisionless Plasma Model Solutions," Journal of Computational Physics, vol. 68, No. 1, January 1987, pp. 202-226.

[Kli.3] Klimas, A. J. and Farrell, W. M., "A Splitting Algorithm for Vlasov Simulation with Filamentation Filtration," Journal of Computational Physics, vol. 110, No. 1, January 1994, pp. 150-163.

[Kno.1] Knorr, Georg, "Plasma Simulation with Few Particles," Journal of Computational Physics, vol. 13, 1973, pp. 165-180.

[Kno.2] Knorr, Georg and Shoucri, Magdi, "Plasma Simulation as eigenvalue problem," Journal of Computational Physics, vol. 14, 1974, pp. 1-7.

[Mar.1] Marsden, Jerrold E. and Weinstein, Alan, "The Hamiltonian Structure of the Maxwell-Vlasov Equations," Physics 4D, North-Holland Publishing Company, 1982, pp. 394-406.

[Mor.1] Morrison, P. J., "The Maxwell-Vlasov Equations As a Continuous Hamiltonian System," Physics Letters, vol. 80A, Number 5.6, December 1980, pp. 383-386.

[Mor.2] Morrison, P. J., "Variational Principle and Stability of Nonmonotonic Vlasov-Poisson Equilibria," Z. Naturforsch, vol. 42a, 1987, pp. 1115-1123.

[Mor.3] Morse, R. L., "Multidimensional plasma simulation by the particle-in-cell method," Methods In Computational Physics, Volume 9 (Plasma Physics), Academic Press, Inc., January 1970, pp. 213-239.

[Nic.1] Nicholson, Dwight R., **Introduction to Plasma Theory.** Krieger Publishing Company, 1992, pp. 71-126.

[Nrc.1] National Research Council, **Plasma Science–From Fundamental Research to Technological Applications.** National Academy Press, Washington D.C. 1995, pp. 77-93.

[Nun.1] Nunn, D. "A novel technique for the numerical simulation of hot collision-free plasma; Vlasov hybrid simulation," Journal of Computational Physics, vol. 108, No. 1, September 1993, pp. 180-196.

[Pen.1] Penrose, O., "Electrostatic Instabilities of a Uniform Non-Maxwellian Plasma," Physics of Fluids, vol. 3, No. 2, March 1969, pp. 258-265.

[Pre.1] Press, William H., Flannery, Brian P., Teukolsky, Saul A., Vetterling, William T., **Numerical Recipes: The Art of Scientific Computing (FORTRAN).** Cambridge Press University, 1989.

[Rob.1] Roberts, Keith V. and Potter, D. E., "Magnetohydrodynamic calculations," Methods In Computational Physics, Volume 9 (Plasma Physics), Academic Press, Inc., January 1970, pp. 340-417.

[Sho.1] Shoucri, Magdi and Knorr, Georg, "Numerical Integration of the Vlasov Equation," Journal of Computational Physics, vol 14, 1974, pp. 84-92.

[Sho.2] Shoucri, Magdi, and Gagne, Real R. J., "Numerical Solution of a Two-Dimensional Vlasov Equation," Journal of Computational Physics, vol. 25, No. 2, October 1977, pp. 94-103.

[Ska.1] Skarka, V., Mahajan, S. M. and Fijalkow, E. "Explicit analytical solution of the nonlinear Vlasov-Poisson system," Physics of Plasmas, vol. 1, No. 3, March 1994, pp. 529-540.

[Sti.1] Stix, Thomas Howard, **Waves in Plasmas.** American Institute of Physics, 1992, pp. 151-216.

[Tan.1] Tang, Tao, "The Hermite spectral method for Gaussian-type functions," SIAM Journal of Scientific Computing, vol. 14, 1993, pp. 594-606.

[Tip.1] Tipler, Paul A. **Modern Physics.** Worth Publishers, Inc., 1978, pp. 63-74.

[Wol.1] Wolfram, Stephen, **Mathematica.** Addison-Wesley Publishing Company, 2^{nd} edition, 1991.