NUMERICAL EXPERIMENTS USING AN ELECTROSTATIC, RELATIVISTIC PLASMA SIMULATION CODE

by

RONALD D. PARACHONIAK

B.Sc. Engineering Physics, University of Saskatchewan, 1977

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE in THE FACULTY OF GRADUATE STUDIES (DEPARTMENT of PHYSICS)

We accept this thesis as conforming to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

September 1986

©Ronald D. Parachoniak, 1986
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of  

PHYSICS

The University of British Columbia
1956 Main Mall
Vancouver, Canada
V6T 1Y3

Date  Oct. 1, 1986
ABSTRACT

A 1-1/2 dimensional electrostatic, relativistic, plasma simulation model was constructed, implemented on the UBC FPS array processor, and tested by solving three problems in plasma physics. The model will handle a variety of different initial conditions including cold plasmas, cold beams, and drifting maxwellians. Relativistic particle velocities are properly modelled and up to two species of particles may be specified. The numerical experiments run with the model included cold plasma oscillations, cold and warm two-stream instabilities, and Landau damping. Results of the simulations were compared to theory and excellent agreement was obtained in all the cases studied. The model may now be used with confidence to research various (relativistic) electrostatic problems. It could also be modified to make it fully electromagnetic, in which case it would be useful for simulating many additional phenomena, including laser-plasma interactions.
# TABLE OF CONTENTS

ABSTRACT .................................................................................. ii  

TABLE OF CONTENTS ................................................................. iii  

LIST OF TABLES ........................................................................... v  

LIST OF FIGURES .......................................................................... vi  

ACKNOWLEDGEMENTS ................................................................ viii  

INTRODUCTION .............................................................................. 1  

CHAPTER 1 – PARTICLE SIMULATION THEORY  
  
1.1 Introduction ........................................................................... 6  
1.2 Particle models ...................................................................... 7  
1.3 The basic computational cycle .................................................. 8  
1.4 Particle Collisions .................................................................. 9  
1.5 Particle and force weighting ...................................................... 11  
1.6 Effects of the spatial grid .......................................................... 14  
1.7 Time integration of the particles ............................................... 16  
1.8 Integration of the field equations ............................................... 19  
1.9 Particle position and velocity loading ...................................... 22  
1.10 Computer variables ............................................................... 25  

CHAPTER 2 – PROGRAM ESR1  
  
2.1 Introduction ........................................................................... 27  
2.2 Overall structure of ESR1 ........................................................ 28  
2.3 Subroutine CREATR ............................................................... 29  
2.4 Subroutine SETRHO ............................................................. 32  
2.5 Subroutine FIELDS ............................................................... 32  
2.6 Subroutine SETV ................................................................. 33  
2.7 Subroutine ACCEL ............................................................... 34  
2.8 Subroutine MOVE ............................................................... 35  
2.9 Miscellaneous Subroutines ...................................................... 35  
2.10 Program ES1PLT ................................................................... 36
CHAPTER 3 - THE PLASMA DISPERSION RELATION

3.1 Introduction ......................................................... 37
3.2 Derivation of the dispersion relation ................................ 37
3.3 Dispersion relation — cold beams .................................. 40
3.4 Dispersion relation — non-relativistic Maxwellian .............. 41
3.5 Dispersion relation — relativistic Maxwellian ..................... 42

CHAPTER 4 - COLD PLASMA OSCILLATIONS

4.1 Introduction ......................................................... 45
4.2 Theory ................................................................. 47
4.3 Trial runs and results ............................................... 48

CHAPTER 5 - TWO STREAM INSTABILITIES

5.1 Introduction ......................................................... 54
5.2 Theory ................................................................. 55
5.3 Trial runs and results ............................................... 58
  5.3.1 Non-relativistic two stream instability (cold beams) .......... 58
  5.3.2 Non-relativistic two stream instability (warm beams) ......... 61
  5.3.3 Relativistic two stream instability (cold beams) ................ 62

CHAPTER 6 - LANDAU DAMPING

6.1 Introduction ......................................................... 76
6.2 Trial runs and results ............................................... 77
  6.2.1 Non-relativistic Landau damping ................................ 77
  6.2.2 Relativistic Landau damping .................................... 80

CHAPTER 7 - RESULTS and CONCLUSIONS ............................ 86

REFERENCES ............................................................. 88

APPENDIX A - Two-stream instability dispersion relation

A.1 Introduction ......................................................... 91

APPENDIX B - ESR1 users guide and program listing

B.1 Introduction ......................................................... 97
B.2 Setup Procedure .................................................... 97
B.3 Running ESR1 and ES1PLT ........................................... 99
LIST OF TABLES

I-1 Bit-reversed fractions .................................................. 24

III-1 Dispersion relation — Non-relativistic Maxwellian .................. 43

III-2 Dispersion relation — Relativistic Maxwellian ..................... 44

IV-1 Input Parameters. (Run no. 105422) .................................. 49

V-1 Input Parameters (Run no. 121112) .................................... 58

V-2 Two-stream Instability Results .......................................... 64

VI-1 Input Parameters. (Run no. 101708) .................................. 77

VI-2 Input Parameters. (Run no. 124958) .................................. 79

VI-3 Input Parameters (Run no. 164028) .................................. 80
**LIST OF FIGURES**

4-1(a,b) DISPERSION RELATION .......................................................... 51
4-2 ELECTROSTATIC ENERGY—SELECTED MODES. ................................. 52
4-3 PARTICLE CROSSING EFFECTS. ..................................................... 53
A-1 DISPERSION RELATION ............................................................... 57
5-2 ELECTROSTATIC ENERGY—MODE 1. ............................................... 65
5-3 ELECTROSTATIC ENERGY—MODES 1 and 2. .................................. 66
5-4 EVOLUTION in PHASE SPACE. ....................................................... 66
5-5 DISTRIBUTION FUNCTION. ............................................................ 68
5-6 ELECTRON KINETIC ENERGY and MOMENTUM. ............................... 69
5-7 ELECTROSTATIC ENERGY—MODE 1. ............................................... 70
5-8 ELECTROSTATIC ENERGY—MODES 1 and 2. .................................. 71
5-9 CHARGE DENSITY. .......................................................... 72
5-10 ELECTROSTATIC POTENTIAL. .................................................... 73
5-11 EVOLUTION in PHASE SPACE. .................................................... 74
5-12 DISTRIBUTION FUNCTION. ........................................................ 75
6-1(a,b) DISTRIBUTION FUNCTION and MODE 1 ENERGY ...................... 82
6-2 ELECTROSTATIC ENERGY—MODE 1. ............................................... 83
6-3(a,b) DISTRIBUTION FUNCTION & DENSITY PERTURBATION .............. 84
6-4(a,b) ELECTROSTATIC ENERGY—MODE 1. ..................................... 85
A-1 DISPERSION RELATION .............................................................. 92
A-2a DISPERSION RELATION ............................................................. 93
A-2b DISPERSION RELATION (Cont.) ................................................ 94
A-3a PROGRAM ROOOTER ............................................................... 95
A-3b PROGRAM ROOOTER (Cont.) ...................................................... 96
B-1 INPUT FILES—ESR1 ................................................................. 100
B-2 INPUT FILES—ES1PLT ............................................................ 101
B-3 ESR1.INSERT Listing ............................................................. 103
B-4 ESR1.CLICHE Listing ............................................................ 104
B-5 ESR1 and ES1PLT— Listing ..................................................... 105
ACKNOWLEDGMENTS

Special thanks to my supervisor Dr. A.J. Barnard for his advice and help in this work. I would also like to thank Sandy Calder for many useful discussions, and the National Sciences and Engineering Research Council for financial support.
INTRODUCTION

Computer modelling is now established as a very important method in all branches of science. In many cases it is the only way to arrive at solutions for complex problems. Oftentimes experiments may be difficult or even impossible to do and analytical treatments may not be feasible. Also, there are cases where the correct theory is not known. Here, computer modelling may be helpful in differentiating among the possibilities. Computer simulations of plasmas have proven fruitful in all these areas.

The goal of this thesis work was to construct and test a 1-1/2 dimensional electrostatic, relativistic, plasma simulation code. This goal was met and the result of the work is the program ESR1 (ElectroStatic Relativistic 1-1/2 dimensional). A 1-1/2 dimensional code was selected mainly for economical reasons—working in a higher dimension would have increased running costs prohibitively. Fortunately, it turns out that working in only 1-1/2 dimensions is not excessively restrictive; there are many phenomena that may be adequately modelled. In addition to the problems examined in this thesis, examples include beam-plasma interactions, plasma expansion into a vacuum, and so-called explosive instabilities. Although many of the available topics have been well covered in the literature, a
number of them have not been treated relativistically. Thus having a 1-1/2 D relativistic code that is tested and inexpensive (compared to higher dimensional models) is very useful. Also, the program could be extended to include electromagnetic fields and then used to investigate a wide variety of phenomena related to laser-plasma interactions such as stimulated Raman scattering, two-plasmon decay, stimulated Brillouin scattering, etc. Finally, since the program is well documented and easy to use, it could be very useful as a learning tool for students taking graduate level courses in plasma physics. The extensive graphical output routines contained in the program would help to illuminate many basic plasma phenomena such as waves, oscillations, and instabilities.

ESR1 was constructed by taking an existing program, (ES1, originally written by A.B. Langdon in 1972), as a basic shell, extensively modifying it to include relativistic effects, and adding graphical output. The process of transforming ES1 into ESR1 and then implementing it at UBC required three main steps.

The first step was to change the code in order to make the program relativistic. This mainly entailed modifications to the particle loading and particle acceleration routines. The particle loading was changed to allow either a relativistic or a non-relativistic Maxwellian distribution to be used (depending on the temperature specified). An option for using a quiet start and the ability to add, relativistically, a drift velocity to all particles were also included. In a quiet start, the particles are loaded in a uniform (but not ordered) way in order to reduce the initial noise level. In the particle acceleration subroutine, algorithms devised by Boris[1] and Buneman[2] were incorporated in order to handle properly the relativistic effects.
A number of other changes were also made; not all of them directly related to making
the program relativistic. Examples include:

- the system of units was changed from a modified MKS system to cgs-Heaviside/Lorentz.
- all energies were normalized to the rest mass energy of species one (usually elec-
trons) and momenta are reported in units of the speed of light.
- the initial density (position) perturbation was changed to be in terms of interpar-
ticle spacings instead of total system length.
- routines were added to calculate frequencies and damping rates automatically.
- restart capability was incorporated.

The second major step in effecting the change from ESI to ESR1 was to add the
graphics routines. ESI was written with the graphics routines interwoven with the main
part of the code and, unfortunately, the graphics routines used were not available at UBC.
This meant that almost totally new graphics routines had to be written, debugged, and
integrated into the new program.

The third major step involved setting up ESR1 to run on the UBC Floating Point
Systems array processor (the "AP"). The AP is an attached arithmetic processor which,
because of its speed and low operating charges, allows one to run large programs such as
ESR1 relatively inexpensively at any time of the day. Before the AP was available, one
normally would only run a program of this size overnight on extra-low priority. While
the AP is very good for running production programs, it is prohibitively expensive to
use for program development and debugging (the compiler is very slow and expensive to
run). Thus, ESR1 was first run and debugged on the UBC Amdahl computer and then
transferred to the AP. Transferring the program was a fairly major undertaking, principally because of the data transfer that takes place between the Amdahl and the AP when doing a run. This data transfer is necessary because the AP has no facility for I/O; all the printed output as well as the plotting routines must be done by the Amdahl. Most of the data transfer is done via unformatted files which have to be translated before they can be used by the plotting program. This translation is necessary because the AP has a 64 bit word while the Amdahl uses a 32 bit word.

After the program was running on the AP, it still had to be tested—i.e. verified against problems that had known solutions. This testing had to cover both the relativistic and non-relativistic regimes. Three tests were chosen. The first, “Cold Plasma Oscillations”, provided a good opportunity to check out the overall program operation. Next came trial runs on the “Two Stream Instability” and “Landau Damping”. Both these tests were run with relativistic and non-relativistic parameters and so they provided an opportunity to check out the relativistic modifications to the program. As well, such items as energy variation, (a measure of overall energy conservation), and quiet start technique were checked. In all three of these cases ESR1 provided results that agreed very well with the theoretical results. Thus the program may now be used with confidence to venture into new areas of research.

The remainder of this report is organized along the following lines.

In chapter 1, a brief account of the theory behind particle simulation is given. This chapter draws heavily on the work of Hockney and Eastwood[3] and Birdsall and Langdon[4]. Various aspects are covered including the effects of the mesh and timestep sizes,
solution of the field equations using fast fourier transform techniques, and initialization of
the run.

Chapter 2 gives a detailed description of the code ESR1. Subroutine functions are
outlined and detailed explanations of the code "inner workings" are given when believed
necessary.

In chapter 3, we digress slightly by reviewing the dispersion relation for a plasma
and solving it for various distribution functions. These results are used in the next three
chapters as benchmarks to which the simulation results are compared.

Chapters 4, 5, and 6 give the results of the computer trial runs performed using ESR1.
In each case a brief theoretical discussion of the problem is given followed by the simulation
parameters used and the results obtained.

Finally, in chapter 7 we summarize the results obtained and draw some conclusions.

There are two appendices. Appendix A contains a listing of the program used to solve
the dispersion relation numerically for the two-stream instability and the results obtained
from the program. Appendix B contains a users guide to ESR1 and a complete program
listing.
1.1 INTRODUCTION

There are three main classes of plasma simulation models — fluid codes, Vlasov codes and particle codes.

Fluid codes use the equation of continuity, the force equation (Newton's second law of motion), and Maxwell's equations to describe the plasma. Essentially, the plasma is considered to be two interpenetrating fluids with no account being taken of the differing particle velocities. The advantage of the method is its simplicity, which is a result of only equations with three space dimensions and time being used. The disadvantage is that when using the fluid equations, no velocity-dependent effects such as Landau damping may be considered.

Vlasov codes integrate the distribution function, \( f(x, v, t) \), forward in time according to the Vlasov equation. Although Vlasov codes have been successfully used in 1-D simulations, they have not yet proven to be as generally useful as particle simulations, especially in two or more dimensions.
Particle codes represent the plasma by a large number of computational particles ("superparticles"), which move in their self-consistent electromagnetic fields according to the laws of classical mechanics. The name superparticle is used because each computer particle represents very many real plasma particles. Particle models may be electrostatic, magnetostatic, or fully electromagnetic. They come in $1, 1\frac{1}{2}, 1\frac{3}{2}, 2, 2\frac{1}{2},$ and 3 dimensions. For example, $1\frac{1}{2}$ dimensional implies a one-dimensional model which allows the charge slabs to have a $y$-velocity. The particle motions are specified by $x, v_x,$ and $v_y$ — there is no spatial variation in the $y$-direction. In this way, a magnetic field in the $z$-direction may be properly included in the simulation. Similarly, a $1\frac{2}{2}$ dimensional model would imply the specification of $x, v_x, v_y,$ and $v_z$; and so on for the higher dimensions.

In this chapter we concentrate on the theory behind the construction and use of a $1\frac{1}{2}$ dimensional electrostatic, relativistic, particle model.

1.2 PARTICLE MODELS

There are three principal types of particle simulation models.

(i) Particle-Particle models use Coulomb's law directly to calculate forces between particles.

(ii) Particle-Mesh models calculate the fields on a spatial grid and then use these grid quantities to calculate the forces acting on the particles.

(iii) Particle-Particle-Particle-Mesh (Hybrid) models are combinations of the above two ideas.
Particle-particle models have limited use due to the large number of calculations involved per timestep. They are generally used either for small systems with long range forces or for large systems where the interaction forces become negligible after a few interparticle distances.

Hybrid models are generally only needed for large correlated systems that have long range forces.

Particle-mesh models are the most widely used ones today. Although they smooth out forces and fields that have scale lengths less than a grid cell, they are computationally much faster than particle-particle methods. Also, in many problems, such as collisionless plasmas, the smoothing effect of the grid turns out to be more of an advantage than a disadvantage. ESR1 is a particle mesh model.

1.3 THE BASIC COMPUTATIONAL CYCLE

Starting at time zero the memory is loaded with the particle positions and velocities. Using these positions and velocities the charge density (and current density in electromagnetic codes) is calculated on the grid. The assignment of these quantities is done using a form of weighting which is a function of the particle positions. Knowing the charge and current densities, one uses Maxwell's equations to determine the electric and magnetic fields at each grid point. These fields are then used to calculate the forces on the particles using the Newton-Lorentz equation. Once the forces are known, the particle velocities and then the positions are advanced in time. The cycle then repeats with the assignment of new charge and current densities and so on.
1.4 PARTICLE COLLISIONS

The effects of particle collisions are as important in simulation models as they are in real plasmas. A typical charged particle in a plasma is interacting electrostatically with many other charged particles. However, since the field of the particle is greatly reduced at distances greater than a Debye length[5], the particle is limited to roughly $N_D$ simultaneous coulomb collisions. $N_D$ is the number particles in a Debye sphere, i.e. a sphere with its radius equal to a Debye length, $\lambda_D$. The Debye length is defined by

$$\lambda_D = \frac{v_{th}}{\omega_p}, \quad (1-1)$$

with $\omega_p$, the plasma frequency, given by

$$\omega_p^2 = \frac{4\pi n_0 e^2}{m}, \quad (1-2)$$

in cgs units, and the thermal velocity, $v_{th}$, given by

$$v_{th} = \sqrt{\frac{kT}{m}}. \quad (1-3)$$

Note that some authors define $v_{th}$ to be a factor of $\sqrt{2}$ larger than that given here.

In our computer model, each superparticle represents a very large number of electrons or ions and so the forces between them are larger than in a real plasma. Since Coulomb's force is proportional to $r^{-2}$, $r^{-1}$, and $r^0$ in 3-D, 2-D, and 1-D respectively, we see that it becomes very large, (in fact singular in 2-D and 3-D), when two particles get near to each other. These large, short range forces are what give rise to collisional effects. Since the superparticles have larger forces acting between them than in a real plasma, they will also
have larger collisional forces. We need to somehow reduce these forces if we are to model a real system properly. Additionally, in many plasmas of interest the temperatures are so high that collisional effects may be neglected altogether; these are known as collisionless plasmas. When modelling these systems it is especially important to reduce the collision rate. However, while we would like to reduce, or even eliminate, the large, short range forces which give rise to collisions, we must keep the long range part of the force since that is what allows many particles to interact simultaneously (giving rise to the collective behaviour of the plasma).

It turns out that if we have two charge clouds, circular in 2-D and spherical in 3-D, which are free to pass through each other without rotating or changing in shape, we get just the right type of force—it is the Coulomb force at large distances but it goes to zero for short distances. This suggests that if we can somehow treat the point particles as charge "clouds", we may be able to get the desired force. A natural consequence of defining the charge density on a grid is that the point particles acquire an effective size and shape. The actual size and shape depends on the weighting scheme used to assign the charge to the grid. Thus, by using different weighting schemes one may alter the particle size and shape and hence modify the force law to reduce the collision rate. Studies of the collision rates for finite-sized particle systems[2,6] show that in 2-D the rate is independent of the number of particles in a Debye sphere and is reduced by approximately an order of magnitude compared to the point particle case. In 3-D, the reduction in collision rate does depend on the number of particles in a Debye sphere but the reduction is still generally larger than in the 2-D case.
It should be noted that since the force is independent of $r$ in 1-D, it is not necessary, from the standpoint of reducing collisions, to use a spatial grid. However, a grid is generally used since the number of numerical operations required is lowered significantly in most problems. Also, using a grid smooths the fields since no fluctuations smaller than a grid spacing are resolved. The limit on how coarse you may make the grid is generally set by the Debye length. The grid needs to be fine enough to resolve a Debye length in order to calculate properly the charge density (and hence the electric field).

1.5 PARTICLE and FORCE WEIGHTING

As mentioned in the previous section, whenever one uses a grid for the charge density, point particles acquire effective sizes and shapes; we will denote the effective shape by $S(x)$.

In the nearest grid point (NGP) or zero order weighting, the complete particle charge (or other grid quantity) is assigned to the nearest grid point. As a particle moves into the $j$th cell, through the cell boundaries at $x = x_j \pm \Delta x/2$, the grid density due to that particle jumps up. Likewise, when the particle moves out of the cell, the grid density jumps down. The net result is that the particles appear to be finite sized with a rectangular shape of width $\Delta x$. Unfortunately, the abrupt change in density at the particle edges produces sharp discontinuities in the field quantities as particles pass through the grid. This makes the NGP method very "noisy" in both space and time.

First order weighting schemes are an improvement over the NGP scheme. They are called Cloud-in-cell (CIC) or Particle-in-cell (PIC) weighting. In CIC weighting, the par-
ticles are considered to be uniformly charged clouds of a definite size and shape which can pass freely through each other. In PIC weighting, the particles are considered to be point particles but their total charge is interpolated to the grid. Although CIC and PIC weighting are not identical, in 1-D the difference is more one of viewpoint—they are computationally equivalent[7]. Therefore no further distinction between the two will be made here. In the particle-in-cell scheme, assignment of the point charge at position $x$ is by linear interpolation to the nearest two grid points. The net effect is to produce a particle with a triangular shape of width $2 \Delta x$. Since there are no sharp discontinuities, first order weighting smooths density and field fluctuations relative to zero order weighting. First order schemes do require more computation time per particle than NGP weighting but this is usually more than offset by the fact that a coarser grid and fewer particles can be used for a given noise level. Particle-in-cell weighting is used for both charge assignment and force interpolation in ESR1.

Higher order weighting schemes involving the use of quadratic or cubic splines have also been employed by others [3,7]. These methods smooth out the particle shape even more than CIC/PIC weighting and so are a further improvement as far as noise suppression is concerned. The drawback to their use is the further increase in computation time required.

Following Langdon[7], we will look at the effect of the grid on the charge density for a point particle at position $x$. Without a grid, the charge density is given by $q \delta(x' - x)$. With the inclusion of a grid, the particle acquires an effective shape, $S(x)$, (for example triangular of width $2 \Delta x$ in the case of first order weighting). Its charge density is then given by $qS(x' - x)$, where $q$ is the total charge given by $q \int dx' S(x' - x)$. 
The grid charge density \( \rho_j \), is obtained from the charges \( q_i \) located at \( x_i \) from:

\[
\rho_j = \sum_i q_i S(x_j - x_i),
\]

(\( i \) subscripts refer to particle quantities and \( j \) subscripts refer to grid quantities). The force on the particle may be found from:

\[
F_i = q_i \Delta x \sum_j E_j S(x_j - x_i). 
\]

Here \( E_j \) is the electric field on the grid; we will defer its calculation to section 1.8.

As also will be shown in section 1.8, one method of solving Poisson's equation involves using the Fourier transform of \( \rho_j \). In those cases, whenever we transform the charge density we must include the Fourier transform, \( S(k) \), of the charge shape \( S(x) \):

\[
S(k) = \int dx S(x)e^{(-i k \cdot x)}. 
\]

Normally, one uses the same weighting in both the density and force calculations in order to prevent particles from accelerating themselves (self-force) and to ensure momentum conservation. Using different weighting functions can also lead to a gravitation-like instability\[8\]. Hockney and Eastwood\[3\] show that in the absence of roundoff errors, total momentum is conserved if:

(i) identical charge assignment and force interpolation functions are used, and,

(ii) correctly space-centered difference approximations to derivatives are used.
1.6 EFFECTS of the SPATIAL GRID

Langdon[7] and Lindman[9] have analyzed grid effects in detail. They found that the effect of the spatial grid is twofold; it smoothes the interaction force and it couples plasma perturbations to perturbations at different wavelengths (aliases). The coupling strength depends on the smoothness of the weighting (interpolation) method used.

Let us look at this in more detail. The force on a particle at position $x$ is:

$$F(x) = \int dx' F(x', x) n(x'), \quad (1-7)$$

where $n(x')$ is the particle density. Transforming this we get,

$$F(k) = \sum_{p=-\infty}^{+\infty} F_p(k - \frac{1}{2} pk_g)n(k_p), \quad (1-8)$$

where $k_p \equiv k - pk_g$ and the grid wavenumber, $k_g$, is given by $k_g = 2\pi/\Delta x$.

The $p \neq 0$ terms couple the density perturbations and forces at wavenumbers that are integral multiples of the grid wavenumber, $k_g$. These wavenumbers are known as aliases.

Similarly, the charge density in k-space may be shown to be:

$$\rho(k) = q \sum_{p=-\infty}^{+\infty} S(k_p)n(k_p). \quad (1-9)$$

Again one sees that aliases become coupled through the grid.
1.7 TIME INTEGRATION of the PARTICLES

Assuming we already know the electric and magnetic fields on the grid, and hence at the particle positions, we need a discrete (finite difference) form for the Newton-Lorentz equation of motion. The most commonly used method, and the one used in ESR1, is the leap-frog method.

In this method, the continuous equations are converted to the finite difference forms as follows:

\[ m \frac{dv}{dt} = F \quad \rightarrow \quad m \frac{v_{\text{new}} - v_{\text{old}}}{\Delta t} = F_{\text{old}}, \quad (1 - 10) \]

and

\[ \frac{dx}{dt} = v \quad \rightarrow \quad \frac{x_{\text{new}} - x_{\text{old}}}{\Delta t} = v_{\text{new}}. \quad (1 - 11) \]

Using equation (1-10), \( v_{\text{new}} \) is calculated and then equation (1-11) is used to solve for \( x_{\text{new}} \).

The leap-frog method used here is time centered and has an error that is proportional to \((\Delta t)^2\). To see the time centering, consider equation (1-11). \( x_{\text{new}} \) is defined at time \( t + \Delta t \) and \( x_{\text{old}} \) is at time \( t \); the difference is centered about time \( t + \Delta t/2 \) — which is where \( v_{\text{new}} \) is defined. Let us now examine the error in going from the continuous equations of motion (the LHS of equations (1-10) and (1-11)) to the finite difference forms (the RHS of the same equations).

We will work in one dimension for simplicity. Let \( \ddot{x} \) be the solution to the continuous equation of motion,

\[ \frac{d^2\ddot{x}}{dt^2} = F/m. \quad (1 - 12) \]
Combining the finite difference forms of equations (1-10) and (1-11) we get

\[ \frac{x_{t+\Delta t} - 2x_t + x_{t-\Delta t}}{\Delta t^2} = \frac{F(x_t)}{m}. \]  

(1 - 13)

Now substitute \( \dot{x} \) into equation (1-13). The error, \( \epsilon_t \), is defined as the difference between the LHS and the RHS:

\[ \epsilon_t = \frac{F(\dot{x}_t)}{m} - \frac{\ddot{x}_{t+\Delta t} - 2\ddot{x}_t + \ddot{x}_{t-\Delta t}}{\Delta t^2}. \]  

(1 - 14)

We now do a Taylor series expansion of \( \ddot{x}_{t+\Delta t} \) and \( \ddot{x}_{t-\Delta t} \) around \( \ddot{x}_t \) to get:

\[ \epsilon_t = \frac{F(\dot{x}_t)}{m} - \frac{d^2 \ddot{x}}{dt^2} \bigg|_{time=t} + \frac{(\Delta t)^2}{12} \frac{d^4 \dot{x}}{dt^4} \bigg|_{time=t} + \ldots \text{(higher order terms)}. \]  

(1 - 15)

Thus we see that \( \epsilon_t \propto (\Delta t)^2 \).

Using the Newton-Lorentz equation for the force on the particles, \( F = q(E + v \times B) \), equation (1-10) becomes

\[ \frac{v_{t+\Delta t/2} - v_{t-\Delta t/2}}{\Delta t} = \frac{q}{m} \left( E + \frac{v_{t+\Delta t/2} + v_{t-\Delta t/2}}{2} \times B \right). \]  

(1 - 16)

The method used in ESR1 to solve this equation was devised by Boris\cite{10}. Using this method, separation of the parallel and perpendicular components of \( v \) is not required. Briefly, if one substitutes

\[ v_{t-\Delta t/2} = v^- - \frac{qE \Delta t}{2m}, \]  

(1 - 17)

and

\[ v_{t+\Delta t/2} = v^+ + \frac{qE \Delta t}{2m}, \]  

(1 - 18)
into (1-16), then $E$ cancels out, leaving

$$\frac{v^+ - v^-}{\Delta t} = \frac{q}{2m} (v^+ + v^-) \times B, \quad (1-19)$$

which is just a rotation of $v^+$ and $v^-$. So the steps in the algorithm are:

(i) calculate $v^-$ from equation (1-17) knowing $v_{t-\Delta t/2}$ and $E$.

(ii) use equation (1-19) to obtain $v^+$.

(iii) use equation (1-18) to obtain $v_{t+\Delta t/2}$.

To handle relativistic particles, $v$ is replaced by $u$ where $u = \gamma v \, [1]$. Equation (1-16) becomes

$$\frac{u_{t+\Delta t/2} - u_{t-\Delta t/2}}{\Delta t} = \frac{q}{m} \left( E_t + \frac{1}{c} \left( \frac{u_{t+\Delta t/2} + u_{t-\Delta t/2}}{2\gamma_t} \right) \times B_t \right), \quad (1-20)$$

where $m$ is the rest mass and $\gamma^2 = 1 + (u^-/c)^2 = 1 + (u^+/c)^2$. Using these definitions, equations (1-17),(1-18), and (1-19) become

$$u_{t-\Delta t/2} = u^- - \frac{QE_t \Delta t}{2m}, \quad (1-21)$$

$$u_{t+\Delta t/2} = u^+ + \frac{QE_t \Delta t}{2m}, \quad (1-22)$$

and,

$$\frac{u^+ - u^-}{\Delta t} = \frac{q}{2m\gamma_t c} (u^+ + u^-) \times B_t. \quad (1-23)$$

Equation (1-23) expresses a rotation of $u^+$ and $u^-$ through the angle

$$\theta = -2 \arctan \left( \frac{qB \Delta t}{2\gamma mc} \right), \quad (1-24)$$
about an axis that is parallel to $B$.

In ESR1, where we have only $u_x, u_y$, and $B_z$, an algorithm conceived by Buneman[2] is used to solve equation (1-23):

$$u'_x = u_x + u_y t,$$  \hspace{1cm} (1 - 25)

$$u'_y = u_y - u_x s,$$  \hspace{1cm} (1 - 26)

and,

$$u'_z = u'_x + u'_y t,$$  \hspace{1cm} (1 - 27)

where

$$t = -\tan (\theta/2) = \frac{qB_z \Delta t}{2\gamma \gamma_i mc},$$  \hspace{1cm} (1 - 28)

and

$$s = -\sin \theta = \frac{2t}{(1 + t^2)}.$$  \hspace{1cm} (1 - 29)

One final note. In order to be truly relativistic, the finite sized particles should undergo a Lorentz contraction—making the form factor a function of velocity. If the particles are highly relativistic, then the particle size may try to become less than a grid spacing (which of course it can't because of the charge interpolation scheme). Dawson, [6], notes that to his knowledge such corrections have never been employed and that for even strongly relativistic situations, reasonably accurate results are obtained without the correction. ESR1 does not contain the correction.
1.8 INTEGRATION of the FIELD EQUATIONS

Starting with the charge and current densities defined on the grid, one uses Maxwell's equations to solve for the fields, E and B. In electrostatic codes like ESR1, we have

\[ \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} = 0, \tag{1-30} \]

so that working in one dimension we get

\[ \mathbf{E} = -\nabla \phi \quad \rightarrow \quad E_x = -\frac{\partial \phi}{\partial x}, \tag{1-31} \]

\[ \nabla \cdot \mathbf{E} = \rho \quad \rightarrow \quad \frac{\partial E_x}{\partial x} = \rho, \tag{1-32} \]

and

\[ \nabla^2 \phi = -\rho \quad \rightarrow \quad \frac{\partial^2 \phi}{\partial x^2} = -\rho. \tag{1-33} \]

The finite difference forms of equations (1-31), and (1-33) are

\[ E_j = \frac{\phi_{j-1} - \phi_{j+1}}{2\Delta x}, \tag{1-34} \]

and

\[ \frac{\phi_{j-1} - 2\phi_j + \phi_{j+1}}{(\Delta x)^2} = -\rho_j, \tag{1-35} \]

where the subscript \( j \) refers to the grid location.

One method of solution for the above equations is to use a direct finite-differencing technique such as Gaussian elimination [3]. In ESR1, we take advantage of the system periodicity to solve equations (1-33) and (1-34) by using Fourier transform methods. Discrete Fourier series are used for all the grid quantities—\( \rho, \phi, \) and \( \mathbf{E} \)—and this means that
(spatial) spectral information is available on them. This proves helpful in relating the simulation results to theory and it also allows smoothing of field quantities.

We define the discrete Fourier transform pair for \( \rho \) as \((x_j = j\Delta x)\):

\[
\rho(k) = \Delta x \sum_{j=0}^{N_y-1} \rho(x_j)e^{-ikx_j},
\]

(1 - 36)

and

\[
\rho(x_j) = \frac{1}{L} \sum_{n=-N_y/2}^{N_y/2-1} \rho(k)e^{ikx_j}.
\]

(1 - 37)

( finite system with \( k = 2\pi n/L, L = N_y\Delta x, \) and \( N_y = \) no. of grid points). The transforms for \( \phi \) and \( E \) are similarly defined.

With these definitions the gradient equation, (1-34), and Poisson's equation, (1-35), become

\[
\frac{\phi_{j-1} - \phi_{j+1}}{2\Delta x} = E_j \quad \longrightarrow \quad -\kappa(k)\phi(k) = E(k),
\]

(1 - 38)

and

\[
\frac{\phi_{j-1} - 2\phi_j + \phi_{j+1}}{(\Delta x)^2} = -\rho_j \quad \longrightarrow \quad K^2(k)\phi(k) = \rho(k),
\]

(1 - 39)

with

\[
\kappa(k) = k\text{ dif}(k\Delta x),
\]

(1 - 40)

\[
K^2(k) = k^2\text{ dif}^2\left(\frac{k\Delta x}{2}\right),
\]

(1 - 41)

and

\[
d\text{if} \theta = \frac{\sin \theta}{\theta}.
\]

(1 - 42)
The term $K^2$ is due to the finite grid size [4]. It is easy to see that $K^2 \rightarrow k^2$ as the grid becomes finer.

With $N_g$ being the number of grid points, the procedure for solving for electric field, $E$, is

(i) Fourier transform the $N_g$ values of $\rho(x)$ to get $\rho(k)$.

(ii) use equation (1-39) to obtain $\phi(k)$ from $\rho(k)$.

(iii) inverse transform $\phi(k)$ and use equation (1-34) to obtain $E$.

There are $N_g$ values of $k$, starting from $k_{\text{min}} = 2\pi/L$ up to $k_{\text{max}} = \pi/\Delta x$ and their negatives, or, in terms of wavelength, $\lambda_{\text{min}} = 2\Delta x$ and $\lambda_{\text{max}} = L$.

For the force we define

$$F(k) = \int_{-\infty}^{+\infty} dx \, F(x) \, e^{-ikx}. \tag{1-43}$$

Using the force interpolated from the grid, equation (1-5), we get

$$F(k) = \int_{-\infty}^{+\infty} dx \, \left[ q \Delta x \sum_j E_j S(x_j - x) \right] e^{ik(x_j - x)} e^{-ikx}. \tag{1-44}$$

Now, reverse the order of the integral and the sum to get

$$F(k) = q \left[ \Delta x \sum_j E_j e^{-ikx_j} \right] \left[ \int_{-\infty}^{+\infty} dx \, S(x_j - x) e^{ik(x_j - x)} \right], \tag{1-45}$$

which, since $S(-k)$ is independent of $j$, is

$$F(k) = qE(k)S(-k). \tag{1-46}$$
Chapter 1 PARTICLE SIMULATION THEORY

Application of the above ideas, i.e. the effects of the finite grid when one is trying to compare simulation results to plasma theory, may be found in chapter 4, entitled “Cold Plasma Oscillations”.

1.9 PARTICLE POSITION and VELOCITY LOADING

Since a particle code is basically solving for the development in time of an initial value problem, suitable initial conditions must be specified. This means that the \( t = 0 \) positions and velocities of each particle must be loaded into the program.

Given the desired particle density, \( n_0(x) \), and velocity distribution, \( f_0(v) \), one must determine the particle positions and velocities. This is done by inverting the appropriate cumulative distribution function. For simplicity, we will illustrate the procedure by examining the case of a 1-dimensional velocity distribution.

Assume that we wish to load the particles from \( v = 0 \) to \( v = v_{\text{max}} \) using the distribution function \( f(v) = \exp\left(-v^2/2v_{\text{th}}^2\right) \). The cumulative velocity distribution function is defined by

\[
F(v) = \frac{\int_0^v \exp\left(-v'^2/2v_{\text{th}}^2\right) dv'}{\int_0^{v_{\text{max}}} \exp\left(-v'^2/2v_{\text{th}}^2\right) dv'}.
\]  

(1 - 47)

Here \( v_{\text{th}} \) is the thermal velocity, defined by

\[
v_{\text{th}} = \sqrt{\frac{kT}{m}}.
\]  

(1 - 48)

(Note that for a 1-D thermal distribution, the integration over \( f(v) \) must be evaluated numerically).
First, the distribution function $f(v)$ is normalized so that the integral from $-v_{max}$ to $+v_{max}$ equals the number of particles for the species. Then, the cumulative distribution function, $F(v)$, represents the number of particles with velocities $\leq v$. The velocity initialization is obtained by integrating from 0 to $v$ and assigning the velocity $v$ to a particle each time the integral increases by one over its previous integer value, i.e.

$$\frac{\int_{v_i}^{v_i+1} f(v) \, dv}{\int_0^{v_{max}} f(v) \, dv} = 1 \text{ particle.} \quad (1-49)$$

In this way, one-half of the particles are loaded with their velocities in the range 0 to $+v_{max}$. The remaining particles are then assigned the negatives of these velocities.

Initially, the particle positions are loaded uniformly over $L$, the system length. Thus, if the velocities, $v_i$, are chosen ordered, as they are when loaded using cumulative distribution functions, then the positions must be scrambled. One way to do this is to assign positions randomly over the system length. Intuitively, this method would seem good since in real plasmas the particles are distributed randomly in space. Also, the method is easy to implement since random number generators are readily available. However, with random starts one usually gets a certain amount of bunching in phase space. This bunching is equivalent to having correlations among the particles. Thus, for example, although the particles may still conform to a Maxwellian distribution over the total system length, over short sections they may not. The result is that noise levels are introduced into the simulation which may be larger than exist in a laboratory plasma. For some simulation experiments this may not be a problem, but for others, such as Landau damping where the low-amplitude waves are important, it may be unacceptable.
A way around this problem is to use what is known as a *quiet start*. In quiet starts, one tries to load the particles in phase space as smoothly as possible. This is done either by loading the particle velocities in regular arrays or, by using non-random point sets—*bit-reversed sets*. Generally, non-random point sets have been more successful since loading in regular arrays can lead to various non-physical instabilities and recurrences\[11,12\]. Table I-1 shows how base-two and base-three bit reversed fractions are obtained.

<table>
<thead>
<tr>
<th>Base 2 Bit Reversed Fractions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Base 3 Bit Reversed Fractions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

**TABLE I-1  BIT REVERSED FRACTIONS**

Using these number sets the particles may be positioned uniformly (but not ordered) in phase space. Details on implementing this in the simulation code are deferred to section 2.3.
Comparisons between loading schemes that use bit-reversed, Fibonacci, and random number sets have been made by Denavit and Walsh\cite{13}. They found that bit-reversed number sets provide a good means of obtaining quiet starts with low initial noise levels. The noise levels do grow in time but they never exceed the levels obtained when random initializations are used. They also found that for a given mode the duration of the initial quiet period increases with wavelength. This means that quiet starts are especially suitable for problems where the long wavelength modes are most important.

Regardless of which of the above loading schemes are used, the particles are not arranged in phase-space to get Debye shielding. Thus, the simulation generally takes $1/4$ to $1/2$ of a plasma period to adjust so that the fields, positions, and velocities are self-consistent. This adjustment period is especially noticeable in the Landau damping runs where it appears as an abnormally large first peak in the electrostatic field and mode energies.

1.10 COMPUTER VARIABLES

If certain variables in the equations to be solved are normalized, it is possible to eliminate a number of numerical operations from each timestep. To see this, we first re-write equations (1-10) and (1-11) for the leap frog particle mover in order to solve for $v_{new}$ and $x_{new}$. We get

$$v_{new} = v_{old} + \frac{F_{old} \Delta t}{m}, \quad (1 - 50)$$

and

$$x_{new} = x_{old} + v_{new} \Delta t. \quad (1 - 51)$$
If we now write $x$ in terms of $\Delta x$ and $v$ in terms of $\Delta x/\Delta t$, i.e.

$$x \rightarrow \frac{x}{\Delta x}, \quad (1-52)$$

and

$$v \rightarrow \frac{v \Delta t}{\Delta x}, \quad (1-53)$$

equations (1-50) and (1-51) become

$$\left(\frac{v \Delta t}{\Delta x}\right)_{\text{new}} = \left(\frac{v \Delta t}{\Delta x}\right)_{\text{old}} + \frac{q}{m} \frac{E_{\text{old}}(\Delta t)^2}{\Delta x}, \quad (1-54)$$

and

$$\left(\frac{x}{\Delta x}\right)_{\text{new}} = \left(\frac{x}{\Delta x}\right)_{\text{old}} + \left(\frac{v \Delta t}{\Delta x}\right)_{\text{new}}. \quad (1-55)$$

Equation (1-54) may be even further optimized (providing there are many more particles than grid points—the usual case) by normalizing $E$, the electric field, by forming the variable $A$ as follows

$$A = \frac{q}{m} \frac{E_{\text{old}}(\Delta t)^2}{\Delta x}. \quad (1-56)$$

By normalizing $x$, $v$, and $E$ as shown above, the process of updating the positions and velocities is reduced to a series of additions—numerous multiplications by $\Delta t$ each timestep are avoided.
2.1 INTRODUCTION

In this chapter we show how the theory discussed in the previous chapter is implemented in an actual code; in this case a 1-1/2 dimensional electrostatic, relativistic model known as ESR1. A few of the main features of ESR1 are:

(i) it will handle a variety of different particle loading schemes including cold beams, maxwellians, drifting maxwellians, etc.

(ii) it will handle relativistic particle velocities.

(iii) it solves Poisson's equation using efficient fast Fourier transform methods.

The program ESR1 is usually run on the UBC Floating Point Systems FPS-164 Array Processor. Another version of the program is available which runs on the Amdahl. A second program, ES1PLT, contains the routines for reading the unformatted output data from ESR1 and producing plots.

A user's guide and a complete listing of the program are included as Appendix B.
2.2 OVERALL STRUCTURE of ESR1

The program starts by specifying various default values for system variables. These defaults ensure that if no other parameters are input a “cold plasma oscillation” run will be executed. Next, a call to SYS$GETDATE retrieves the system date and time from the FPS-164. Subroutine DATEME then formats this data so that it may be easily used for date and time stamping of all printed and plotted output. Following this, the program reads the changes to the default input data via the NAMELIST statements. ESR1 then calculates several variables and system constants and, if the run is a restart, gets the resart parameters via a call to RESTOR. A printout of the input parameters that characterize the run is then made.

Particle position and velocity loading is done next. The particles are first loaded evenly spaced over the entire system length and given a drift velocity if required. When a non-zero temperature is specified, a call to subroutine CREATR will load the particles with a Maxwellian distribution of velocities. If the temperature is high enough, a relativistic Maxwellian distribution is used.

Calls to SETRHO, FIELDS, and SETV follow. In SETRHO the charge density is accumulated on the grid and the positions are normalized. The initial call to FIELDS sets up the fields on the grid and calculates the field energy. SETV (via a call to ACCEL) accomplishes the task of calculating the velocities a half step backwards in time. This is needed because the simulator loads all positions and velocities at time zero. However, the leap-frog particle mover requires the particle positions, x, to be defined at a time \( \Delta t/2 \) larger than the time at which the velocities, v, are defined.
Now, with $x$ and $E$ defined at $t = 0$, and $v$ defined at $-\Delta t/2$, the program starts to cycle through the timestep loop. First a call to ACCEL is made. Here the velocity is advanced one timestep and the momentum and kinetic energies are calculated. Next, in subroutine MOVE, the particle positions are advanced and the charge is accumulated on the grid. FIELDS then uses these grid charge densities to calculate $\phi$, the electric potential, using Poisson's equation. Also in subroutine FIELDS, the electrostatic field and mode energies are calculated and field "snapshots" are saved. After FIELDS, program control passes back to ACCEL to begin another timestep loop.

At the end of the run (and sometimes during the run if it is very long), subroutine HISTRY is called to save various quantities versus time for later plotting.

The subroutines used in ESR1 will now be discussed in more detail.

### 2.3 Subroutine CREATR

In order to load a non-relativistic Maxwellian distribution of velocities, one does an inversion of the cumulative distribution function, $F(v)$, where

$$F(v) = \frac{\int_0^v \exp\left(-\frac{v^2}{2v_{th}^2}\right) dv}{\int_0^\infty \exp\left(-\frac{v^2}{2v_{th}^2}\right) dv}. \quad (2-1)$$

Here $v_{th}$ is the thermal velocity, defined by

$$v_{th} = \sqrt{\frac{kT}{m}}. \quad (2-2)$$
Chapter 2 PROGRAM ESR1

For a 1-D thermal distribution, the integration over \( f(v) \) must be evaluated numerically.

In the case of a 2-D isotropic, thermal distribution, \( f(v) \), the integration may be done analytically since we have

\[
\text{speed, } v = \sqrt{v_x^2 + v_y^2},
\]

\[
\theta = \arctan \left( \frac{v_y}{v_x} \right),
\]

and

\[
dv = 2\pi uvdu.
\]

If the temperatures are low (non-relativistic), then the distribution functions \( f(v_x) \) and \( f(v_y) \) are each normalized so that the integral from \(-v_{max}\) to \(+v_{max}\) equals the number of particles for the species. The velocity initialization is obtained by integrating from 0 to \( v \) and assigning the velocity \( v \) to a particle each time the integral increases by one over its previous integer value. In this way, one-half of the particles are loaded with their velocities in the range 0 to \( v_{max} \). The remaining particles are assigned the negatives of these velocities. In ESR1, \( v_{max} \) is set to \( 4v_{th} \).

For isotropic distributions with relativistic temperatures, the 2-D relativistic Maxwellian is (\( p = \gamma |v|/c \)):

\[
f(p) = \frac{NPme^2}{2\pi T} \frac{1}{T/mc^2} \exp \left[ \frac{-mc^2}{T} (\gamma - 1) \right]. \tag{2-3}
\]

In ESR1, \( f(p) \) is normalized by setting the integral over all \( p \) equal to the number of particles in the species:

\[
\int_0^{2\pi} d\theta \int_0^\infty f(p)pdp = NP, \tag{2-4}
\]
where

\[ \theta = \cos^{-1}(p_x/p) = \sin^{-1}(p_y/p). \]

The number of particles in the first quadrant is obtained by integrating from 0 to \(\pi/2\). If we also change the integral to a sum (i.e. we integrate numerically) we obtain

\[
\frac{\pi}{2} \sum_{n=1}^{IP} (n\Delta p)\Delta p f(n\Delta p) = \frac{N_p}{4}. \tag{2-5}
\]

\(N_p\) is the number of particles assigned to all quadrants up to the value of \(p\). Each time the running index \(IP\) reaches a value which increases the value of the integral by one over its previous integer value, a particle is assigned the quantity \(p = IP\Delta p\) and an angle \(\theta\) between 0 and \(\pi/2\). To prevent drift and bias in \(v_x\) and \(v_y\), 4 particles with the same \(p\) but the angles \(\theta, \theta + \pi/2, \theta + \pi, \text{ and } \theta + 3\pi/2\) are loaded at each step. The particle velocities are obtained from \(p\) by using the relations

\[ p \cos \theta = \gamma v_x, \tag{2-6} \]

and

\[ p \sin \theta = \gamma v_y. \tag{2-7} \]

If a drift velocity, \(v_0\), is specified, it is added relativistically to each particle velocity[14]:

\[
v'_x = \frac{v_x + v_0}{1 + v_x v_0/c^2} \quad \text{and} \quad v'_y = \frac{v_y \sqrt{1 - v_0^2/c^2}}{1 + v_x v_0/c^2}. \tag{2-8, 9}
\]

For a quiet start, the angle \(\theta\) is generated using a radix-three bit reversed number set and the particle positions are "scrambled" using a radix-two bit reversed set.
If a random start is specified, the angle θ is generated using a random number generating routine and then a call to subroutine SMEAR is made in order to perform random pair exchanges of the particle positions.

One final comment about subroutine CREATR. In ESR1, Δx/Δt is set equal to the speed of light, c. Since subroutine CREATR gives velocities in terms of c, it means that the velocities output from it are already normalized as discussed in section 1.10 ("computer variables").

2.4 Subroutine SETRHO

During initialization, this subroutine is called once for each species. In SETRHO, the particle positions, x, are converted to x/Δx, and the charge is accumulated on the grid points using linear (first order) weighting.

2.5 Subroutine FIELDS

As shown in section 2.6, the three point difference form for Poisson's equation gives

\[
\frac{\phi_{j-1} - 2\phi_j + \phi_{j+1}}{(\Delta x)^2} = -\rho_j \quad \rightarrow \quad K^2(k) \phi(k) = \rho(k) ,
\]

(2-10)

with

\[
K^2(k) = k^2 \frac{d i f \; \theta}{2} \left( \frac{k \Delta x}{2} \right) ,
\]

(2-11)

and

\[
d i f \; \theta = \frac{\sin \theta}{\theta} .
\]

(2-12)
One of the first things done in FIELDS is to calculate $K^2(k)$, the ratio of $\rho(k)$ to $\phi(k)$. At the same time, a quantity called $SM(k)$ is calculated. $SM(k)$ is a smoothing factor which allows one to modify the $k$ spectrum in going from $\rho(k)$ to $\phi(k)$ [4]. It is usually used to eliminate unwanted short wavelengths but it may also be used to enhance long wavelength modes.

Calls to subroutines CPFT and RPFT2 Fourier transform $\rho(x)$ to $\rho(k)$ which is then multiplied by $1/K^2$ ($KSI$), to obtain $\phi(k)$. At this time the electrostatic field energy is calculated using the formula

$$ESE = \frac{1}{L} \sum_{k=k_0}^{k_{max}} \rho(k)\phi'(k). \quad (2-13)$$

The electrostatic energies for specified individual modes are also calculated here.

Next, $\rho(k)$ and $\phi(k)$ are transformed back by calls to RPFTI2 and CPFT and the electric field is solved for by using a centered difference across two cells.

### 2.6 Subroutine SETV

The subroutine CREATR loads particle positions and velocities at time $t = 0$. However, the leap-frog particle moving algorithm requires $x$ to be leading $v$ by $\Delta t/2$. Subroutine SETV is used to change $v(0)$ to $v(-\Delta t/2)$. First, if a magnetic field is present, the fields at $t = 0$ are used to give a half-step rotation backwards. Then, SETV makes a call to subroutine ACCEL with $q$ replaced by $-q/2$. This causes a half-step acceleration $qE$ to be applied.
2.7 Subroutine ACCEL

In this subroutine, the two-step acceleration/rotation scheme as outlined in section 1.7 is implemented. First, the variable

\[ A = \frac{q E_{old} (\Delta t)^2}{m \Delta x}, \]

is formed at each of the grid points. As shown in section 1.10, in this way \( v_{new} \) may be very simply calculated by using only one addition for each particle. Next, based on the value of \( |q/m| \), ACCEL chooses either a relativistic or non-relativistic algorithm for doing the acceleration and rotation.

After the particle velocities have been advanced one timestep the kinetic energy is calculated. The relativistic formulation of the kinetic energy is given by

\[ KE = (\gamma - 1)mc^2. \tag{2 - 14} \]

The calculation is done both relativistically and non-relativistically and then the larger of the two values is used (since for low velocities the relativistic formulation may be too low because of truncation). The kinetic energies output from ESR1 are given in terms of the total rest mass energy of species 1 (electrons).

The drift momentum per unit mass is next calculated. Since the drift momentum is defined as \( m \sum \gamma_i v_i \), the drift momentum per unit mass is given by \( \sum \gamma_i v_i \) (sum over all particles). Since all velocities are normalized to light speed, \( c \), the drift momentum per unit mass as defined here will be in units of \( c \).
Whenever IT, the timestep iteration counter, equals IThERM, the temperature profile plotting frequency, the relative kinetic energy, RELT, is calculated and temperature profiles are saved. RELT is defined as (the KE of the average (drift) motion plus the random motion) minus (the KE of the average motion).

The parameter VLIGHT is presently only a dummy variable. It will eventually be used to specify the ratio of the speed of light, c, to DX/DT. This will allow the program to be used with fewer timesteps when simulating non-relativistic phenomena.

2.8 Subroutine MOVE

Subroutine MOVE calculates the new particle positions, \(x_{\text{new}}\), from \(x_{\text{old}}\) and \(v_{\text{new}}\) using equation (1-11). If \(x_{\text{new}}\) is outside the range of the simulation box, it is shifted one period to the left or to the right in order to have it in the range \(0 \leq x_{\text{new}} \leq L\). After the new positions are calculated, they are used to accumulate the charge on the grid using linear weighting (PIC).

2.9 Miscellaneous Subroutines

Subroutines RITER1 and RITER2 are used to write various system variables and constants to an output file for later printing.

Subroutine CPFT is used to do forward and reverse fast Fourier transforms (FFT's) on complex data[4]. It transforms \(a(x) + ib(x)\) into \(A(k) + iB(k)\). Since in ESR1 we only transform real data, \((\phi \text{ and } \rho)\), a factor of almost two in speed may be gained by setting up pairs of real data and transforming them at the same time. RPFT2 sets up the real
sequences and RPFTI2 extracts the sequences after an inverse transformation. Calls to RPFT and then CPFT will effect a forward transform while calls to RPFTI2 and CPFT will initiate an inverse FFT.

Subroutine SAVEIT is used to periodically save certain system parameters so that the run may be restarted. Subroutine RESTOR is called if a restart is specified. It reads the data stored by SAVEIT back into memory.

Subroutine SMEAR does random pair exchanges on a specified array. It is used in the particle loading routines.

Function RADREV generates a number between 0 and 1 by mixed-radix digit reversal. It is called by subroutine CREATR if a quiet start is specified.

Subroutine PEAKS calculates the frequency and the damping rates from the electrostatic field and mode energies.

2.10 Program ES1PLT

Program ES1PLT first reads the unformatted data produced by program ESR1 and then converts it to data that can be recognized by the Amdahl computer. This conversion is necessary because of the different word lengths on the AP and the Amdahl (64 and 32 bit words respectively). Next the requested plots are generated and put into a plot file. This plot file may then be used to view the plots on a graphics terminal or to produce hardcopy.
Chapter 3

THE PLASMA DISPERSION RELATION

3.1 INTRODUCTION

We start this chapter by first reviewing the derivation of the dispersion relation for a relativistic plasma. We then solve it for three different distribution functions—cold beams, non-relativistic Maxwellsians, and finally relativistic Maxwellsians. In the following chapters these results will serve as benchmarks to which the simulation results will be compared.

3.2 DERIVATION of the DISPERSION RELATION

The index of refraction is defined by

$$n = \frac{c}{v_\psi} = \frac{ek}{\omega}.$$  

Any medium in which the index of refraction is a function of frequency is said to be dispersive. By definition the dispersion relation is the equation that gives $\omega$ as a function of $k$. All the information pertinent to the propagation, growth, and/or decay of waves in a plasma is contained in its dispersion relation and so this equation occupies a key position in plasma theory.
We will derive the dispersion function by solving the collisionless Boltzmann equation (the Vlasov equation). The relativistic generalization of the Vlasov equation is given by [15,16],

\[
\frac{\partial f}{\partial t} + \frac{p}{m\gamma} \cdot \frac{\partial f}{\partial x} + q \left( E + \frac{p \times B}{\gamma mc} \right) \cdot \frac{\partial f}{\partial p} = 0. \tag{3 - 1}
\]

We will consider a plasma with only one species (electrons) in a uniform, immobile neutralizing background of ions and no external magnetic field. We let the distribution function be a combination of an unperturbed velocity distribution function, \( f_0 = f_0(p) \), and a small perturbation, \( f_1 = f_1(x,p,t) \):

\[
f(x,p,t) = f_0(p) + f_1(x,p,t). \tag{3 - 2}
\]

We let the electric field be in the \( x \)-direction and consider waves which vary in only this direction. In the absence of any externally applied fields, the electric field, \( E \), is proportional to \( f_1 \) and we assume that \( f_1 \), and hence \( E_1 \) are proportional to \( \exp(ik \cdot x - \omega t) \).

We now substitute equation (3-2) into (3-1) and look at linearized solutions (i.e. we drop all quadratic and higher terms). The first order equation is:

\[
\frac{\partial f_1}{\partial t} + \frac{p}{m\gamma} \cdot \frac{\partial f_1}{\partial x} + qE_1 \frac{\partial f_0}{\partial p} = 0. \tag{3 - 3}
\]

The Fourier-Laplace transform and its inverse are defined by

\[
f(k,\omega) = \int_0^\infty dt \int \int_{-\infty}^{\infty} dx \exp[-i(k \cdot x + \omega t)] f(x,t), \tag{3 - 4}
\]

and

\[
f(x,t) = \frac{1}{(2\pi)^4} \int_L d\omega \int \int_{-\infty}^{\infty} dk \exp[i(k \cdot x + \omega t)] f(k,t), \tag{3 - 5}
\]
where the integral over $\omega$ is along the Laplace contour. Using equation (3–4) and the fact that the Laplace transform of a differentiated quantity is given by

$$\mathcal{L} \left[ \frac{\partial}{\partial t} g(t) \right] = -i\omega g(\omega) - g(t = 0) , \quad (3 - 6)$$

the first order Vlasov equation, (3–3), becomes

$$-iw f_1(k, p, \omega) + \frac{ik \cdot p}{m\gamma} f_1 + qE_1(\omega) \frac{\partial f_0}{\partial p} = f_1(k, p, t = 0) , \quad (3 - 7)$$

which may be solved for $f_1$ to get

$$f_1(k, p, \omega) = \frac{-qE_1(\omega) \partial_p f_0 + f_1(k, v, t = 0)}{i(k \cdot p/m\gamma - \omega)} , \quad (3 - 8)$$

where $\partial_p \equiv \partial/\partial p$. When transformed as above, Poisson's equation becomes

$$ikE_1(\omega) = -4\pi q n_0 \int dp \ f_1(k, p, \omega) . \quad (3 - 9)$$

Substituting equation (3–8) into (3–9) and rearranging yields

$$ik \left[ 1 - \frac{4\pi q^2 n_0}{ik} \int dp \ \frac{\partial_p f_0}{i(k \cdot p/m\gamma - \omega)} \right] E_1(\omega) = -4\pi q n_0 \int dp \ \frac{f_1(k, p, t = 0)}{i(k \cdot p/m\gamma - \omega)} . \quad (3 - 10)$$

The factor in the square brackets is the dielectric function; working in 1-D, it is

$$\epsilon(k, \omega) = 1 - \frac{\omega_p^2 m}{k^2} \int dp \ \frac{\partial_p f_0}{(p/m\gamma - \omega/k)} . \quad (3 - 11)$$

Since the zeros of the dielectric function give the normal modes for the plasma, we set $\epsilon(k, \omega) = 0$ to get the dispersion relation:

$$k^2 = \omega_p^2 m \int dp \ \frac{\partial_p f_0}{(p/m\gamma - \omega/k)} . \quad (3 - 12)$$
If we have more than one species the right hand side becomes a sum over $j$ species:

$$k^2 = \sum_j \omega_{pj}^2 m_j \int dp \frac{\partial_p f_{0j}}{(p/m_j \gamma_j - \omega/k)} \tag{3 - 13}$$

We will now look at solutions to this dispersion relation for various distribution functions, $f_0$.

### 3.3 Dispersion Relation — Cold Beams

A cold beam is composed of particles that have a drift velocity but no thermal spread. Its distribution function is

$$f_{0j}(p) = \delta(p - \gamma_j m_j v_j) \tag{3 - 14}$$

Substituting (3-14) into (3-13) we get

$$k^2 = \sum_j \omega_{pj}^2 m_j \int dp \frac{\delta'(p - \gamma_j m_j v_j)}{(p/m_j \gamma_j - \omega/k)} \tag{3 - 15}$$

We now use the relation

$$\int_{-\infty}^{\infty} dx \, \delta'(x - a) f(x) = -\int_{-\infty}^{\infty} dx \, \delta(x - a) f'(x) = -f'(a), \tag{3 - 16}$$

to get

$$k^2 = \sum_j \frac{\omega_{pj}^2}{\gamma_j^3} \frac{1}{(v_j - \omega/k)^2} \tag{3 - 17}$$

or

$$k^2 = \sum_j \frac{\omega_{pj}^2}{(v_j - \omega/k)^2}, \tag{3 - 18}$$
where

\[ \omega^2_{rj} = \omega_p^2 / \gamma_j^3, \quad (3-19) \]

is the relativistic longitudinal plasma frequency, and \( \gamma_j \) is defined by:

\[ \gamma_j = (1 - v_j^2/c^2)^{-1/2}. \quad (3-20) \]

If we adopt the convention that these quantities are measured in the observer's coordinate system (so that \( \omega^{2}_{rj}/\gamma_j \) varies as the rest density and thus is invariant), then equation (3–18) is valid in any coordinate system[17].

### 3.4 DISPERSION RELATION — NON-RELATIVISTIC MAXWELLIAN

In the case of non-relativistic plasmas the dispersion relation, equation (3–12), reduces to the usual formula:

\[ 1 = \frac{\omega_p^2}{k^2} \int dv \frac{\partial v_z f_0}{v_x - (\omega/k)}. \quad (3-21) \]

Solution of this equation must be found by integrating along the Landau contour[5,18]. Assuming we have a Maxwellian distribution and that the phase velocity of the wave, \( v_\phi \), is much larger than the thermal velocity, \( v_{th} \), the dispersion equation, (3–21), gives the following results:

\[ \omega^2 = \omega_p^2 + 3k^2v_{th}^2, \quad (3-22) \]

\[ \omega_i = -\omega_p \left( \frac{\pi}{8} \right)^{\frac{1}{2}} \frac{1}{(k\lambda_D)^3} \exp \left( \frac{-3}{2} \right) \exp \left( \frac{-1}{2k^2\lambda_D^2} \right), \quad (3-23) \]

where

\[ \lambda_D \equiv \frac{v_{th}}{\omega_p} \quad \text{and} \quad v_{th} = \sqrt{\frac{kT}{m}}. \quad (3-24,25) \]
The imaginary part of the frequency, $\omega_i$, vanishes for $k \to 0$ and increases rapidly with an increase in the wavenumber.

When grid effects are taken into account, equations (3-22) and (3-23) become:

$$\omega^2 = S^2(k) \omega_p^2 + 3k^2v_{th}^2, \quad (3 - 26)$$

and

$$\omega_i = -S(k) \omega_p \left( \frac{\pi}{8} \right)^{\frac{1}{2}} \left( \frac{S(k)}{k \lambda_D} \right)^3 \exp \left( -\frac{3}{2} \right) \exp \left( \frac{-S^2(k)}{2k^2 \lambda_D^2} \right), \quad (3 - 27)$$

with

$$S(k) = \left( \frac{\sin \left( k \Delta x/2 \right)}{k \Delta x/2} \right)^2, \quad (3 - 28)$$

for first order weighting. The above results are only valid for small wavenumbers (i.e. only for $\omega_i \ll \omega_r$). When this condition is not satisfied the dispersion relation, equation (3-21), must be solved numerically. A listing of well known numerical results is presented in Table III-1.

### 3.5 Dispersion Relation — Relativistic Maxwellian

When very hot distributions are encountered one must use the relativistic dispersion relation, equation (3-12):

$$k^2 = \omega_p^2 m \int dp \frac{\partial_p f_0}{(p/m\gamma - \omega/k)} . \quad (3 - 12)$$

The one dimensional Maxwell-Boltzmann distribution function is given by:

$$f_0 = \frac{1}{2cK_1(A)\exp A} \exp [-A(\gamma - 1)], \quad (3 - 29)$$
TABLE III-1  DISPERSION RELATION — Non-relativistic Maxwellian.

where

\[ A = \frac{mc^2}{kT}, \]
\[ \gamma = \left(1 - \frac{v^2}{c^2}\right)^{-\frac{1}{2}} = \left(1 + \frac{u^2}{c^2}\right)^{\frac{1}{2}}, \]
\[ u = \gamma v, \]

and \( K_1(A) \) is a modified Bessel function of the second kind. Using equation (3–29) for \( f_0 \), the dispersion relation, equation (3–12), was solved numerically for a number of different temperatures (ratios of \( u_{th}/c \)). The results are displayed in Table III–2.

This ends our review of the plasma dispersion relation. In the next three chapters, we compare analytical and numerical solutions to this equation with simulation results for three different distribution functions.
<table>
<thead>
<tr>
<th>$\omega_r/\omega_p$</th>
<th>$\omega_t/\omega_p$</th>
<th>$-\omega_r/\omega_p$</th>
<th>$-\omega_t/\omega_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_h/c = 0.200$</td>
<td>1.797115</td>
<td>0.438776</td>
<td>0.518240</td>
</tr>
<tr>
<td>$u_h/c = 0.150$</td>
<td>1.777574</td>
<td>0.475776</td>
<td>0.493326</td>
</tr>
<tr>
<td>$u_h/c = 0.100$</td>
<td>1.749787</td>
<td>0.490435</td>
<td>0.489113</td>
</tr>
<tr>
<td>$u_h/c = 0.050$</td>
<td>1.714422</td>
<td>0.440955</td>
<td>0.453595</td>
</tr>
<tr>
<td>$u_h/c = 0.025$</td>
<td>1.699117</td>
<td>0.428868</td>
<td>0.429835</td>
</tr>
<tr>
<td>$k/D$</td>
<td>0.80</td>
<td>0.78</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Table III-2 DISPERSION RELATION. Relativistic Maxwellian.
4.1 INTRODUCTION

In doing a computer simulation one converts the differential equations that describe the plasma behaviour into a set of finite difference equations. In the limit of infinitely small timesteps and grid spacings the difference equations give exactly the same results as the continuous differential equations. However, since in any practical simulation work we are limited to finite differences, we must be cognizant that certain properties may differ between the real plasma and the simulation plasma. For example, weakly damped modes of the plasma may be changed into growing ones (this is known as numerical instability); or, growing modes may be damped. In this chapter we will be concerned only with the effects of the finite grid—we will assume that the timestep is small enough that no effects from it are felt. This situation is generally true as long as the timestep used is shorter than any period of interest, for example, $\Delta t \ll \tau_p$, the plasma period.

As mentioned in the previous chapter, all the information pertinent to growth or decay of plasma waves is contained in the dispersion relation. Thus by comparing the computer dispersion relation with that for a real plasma the effects of the finite differencing may
be obtained. The simplest dispersion relation for comparison is the one for "cold plasma oscillations".

To see how cold plasma oscillations arise, let us consider, for simplicity, a uniform distribution of electrons in a fixed, neutralizing background of ions. If the electrons are displaced from their equilibrium positions, electric fields are set up which tend to restore the electrons to their original positions. When the electrons return to their equilibrium positions, the electric fields vanish. However, the electrons overshoot their equilibrium positions due to their inertia and continue moving until the electric field is large enough to stop them. The electrons are then accelerated back towards their equilibrium positions where they again overshoot. In the absence of collisions, they will continue to oscillate indefinitely about these equilibrium positions. The frequency of oscillation is characteristic of the plasma and depends only on the density. It is given by \[5\] :

\[\omega_p^2 = \omega_e^2 + \omega_i^2, \quad (4 - 1)\]

where

\[\omega_e^2 = \frac{4\pi n_e e^2}{m_e} \quad \text{and} \quad \omega_i^2 = \frac{4\pi n_i Z^2 e^2}{m_i},\]

are the electron and ion plasma frequencies respectively (in cgs units), and \(Ze\) is the ion charge. The ions are so massive compared to the electrons that they may usually be considered stationary. We then have

\[\omega_p \approx \omega_e. \quad (4 - 2)\]

We will now look at the dispersion relation for these waves.
4.2 Theory

The dispersion relation for a cold plasma beam was shown (cf. chapter 3) to be:

\[ k^2 = \sum_{j} \frac{\omega_{pj}^2}{\gamma_j^3 (\nu_j - \omega/k)^2} \tag{4-3} \]

In the non-relativistic, single beam limit \((\gamma, j \rightarrow 1)\), we have

\[ k^2 = \omega_p^2 \frac{1}{(\nu - \omega/k)^2} \tag{4-4} \]

For no drift \((\nu = 0)\), i.e. a stationary, cold plasma, we get the Langmuir result:

\[ \omega^2 = \omega_p^2 \tag{4-5} \]

Birdsall and Langdon [4] show that the dispersion relation for a cold plasma taking into account grid effects is given by:

\[ \epsilon(k, \omega) = 0 = 1 - \frac{\omega_p^2}{K^2(k)} \sum_{p=-\infty}^{\infty} \frac{k_p \kappa(k_p) S^2(k_p)}{(\omega - k_p \nu_0)^2} \tag{4-6} \]

where

\[ \kappa(k_p) = k_p \text{dif}(k_p \Delta x) \tag{4-7} \]

\[ K^2(k) = k^2 \text{dif}^2 \left( \frac{k \Delta x}{2} \right) \tag{4-8} \]

and,

\[ S(k) = \int dx \ S(x) e^{(-ik \cdot x)} \tag{4-9} \]

Here \( \text{dif} \theta = \sin \theta/\theta \). For no drift \((\nu_0 = 0)\) equation (4-6) gives

\[ \omega^2 = \omega_p^2 \left[ \frac{1}{K^2(k)} \sum_{p=-\infty}^{\infty} k_p \kappa(k_p) S^2(k_p) \right] \tag{4-10} \]
In a real plasma we have the Langmuir result, equation (4-5), but in the simulation the dispersion is given by equation (4-10). Thus we see that the effect of the grid on the dispersion relation is given by the terms in the square brackets in equation (4-10).

For a momentum conserving algorithm (i.e. one where the force is calculated by interpolating the differenced potential) and CIC/PIC weighting, we have

\[ S(k_p) = \text{dif}^2 \left| \frac{1}{2} k_p \Delta x \right|, \tag{4-11} \]

so that the dispersion relation, equation (4-10) becomes [7],

\[ \omega = \omega_p \cos \left( \frac{1}{2} k \Delta x \right). \tag{4-12} \]

### 4.3 TRIAL RUNS and RESULTS

In order to test the effect of the grid on the behaviour of the simulation plasma, two cold plasma oscillation cases were run (run nos. 105422 and 104923). The input parameters for run 105422 are given in Table IV–1. The input parameters for run 104923 are identical to those for run 105422 with the exception of the number of grid points, NG; for run 104923, NG was increased from 32 to 64.

In figures 4-1(a,b), the Langmuir result (\( \omega = \omega_p \)), the exact solution taking into account grid effects, equation (4-11), and the measured values from the trial runs for the dispersion relation are plotted. Good agreement between the exact and measured values were obtained.
Input Parameters for run 105422

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>$4\pi$</td>
<td>System length</td>
</tr>
<tr>
<td>NQ</td>
<td>32</td>
<td>Number of grid points</td>
</tr>
<tr>
<td>DX</td>
<td>0.3927</td>
<td>L/NG</td>
</tr>
<tr>
<td>DT</td>
<td>0.02</td>
<td>Timestep size</td>
</tr>
<tr>
<td>NT</td>
<td>500</td>
<td>Number of timesteps</td>
</tr>
<tr>
<td>K0</td>
<td>0.5</td>
<td>$2\pi$/fundamental wavelength</td>
</tr>
<tr>
<td>Nl</td>
<td>256</td>
<td>Number of electrons</td>
</tr>
<tr>
<td>Xll</td>
<td>0.5</td>
<td>Position perturbation</td>
</tr>
<tr>
<td>V10</td>
<td>0.0</td>
<td>Drift velocity</td>
</tr>
<tr>
<td>WP1</td>
<td>1.0</td>
<td>Plasma frequency</td>
</tr>
<tr>
<td>WC1</td>
<td>0.0</td>
<td>Cyclotron frequency</td>
</tr>
<tr>
<td>TEMPI</td>
<td>0.0</td>
<td>Species 1 temperature</td>
</tr>
</tbody>
</table>

Table IV-1 INPUT PARAMETERS. Run no. 105422.

Figure 4-2 displays the electrostatic mode energy plots from run 105422 for selected modes.

From these results it is seen that in any system the lowest modes are represented the best. As the number of grid points is increased, the numerical dispersion and the contributions of the aliases (the term in the square brackets in equation 4-10), can be made negligible. Generally one needs $n < NQ/2$, where $n$ is the mode number of interest.

A third cold plasma oscillation case was run in which the initial position perturbation amplitude, Xll, was set to 0.75. All other parameters were the same as in run 105422. This perturbation was high enough so that particle crossings occurred. Figures 4-3(a,b,c,d) show the effect this has on selected mode energies. It is readily apparent that the modes are no longer independent. Particle-crossing effects are also evident on the phase space
plots (figures 4-3(e,f)) as "wave breaking". In wave breaking we have multiple particles at the same position but with different velocities.

The cold plasma oscillation runs were also used to check the energy conservation properties of the simulation. Since ESR1 uses a momentum conserving algorithm, total energy is not conserved. However, in order for the results to be valid, the variation in the total energy should be much less than that of the field energy. This criterion was well met in all the runs made. Typically, the maximum variation in the total energy was less than 0.7 percent, while the field energy varied from zero to 100 percent of the total energy.

Plots of the electric field, electric potential, and charge density, (not shown), were also checked and found to have the proper amplitudes and phases.

Based on the above results, it may be concluded that ESR1 is working properly in the non-relativistic regime. In the next two chapters we subject the program to more extensive tests in which relativistic effects are important.
Figure 4-1(a,b) DISPERSION RELATION. Run nos. 105422 and 104923. Cold plasma oscillations. (a) 32 grid points. (b) 64 grid points.
Figure 4-2 ELECTROSTATIC ENERGY—SELECTED MODES. Run #105422. Cold plasma oscillations. The decreasing frequency for the shorter wavelength modes is due to grid effects.
Figure 4-3 PARTICLE CROSSING EFFECTS. Run #125627. Cold plasma oscillations run showing the effect on selected mode energies (a,b,c,d). In (e,f), phase space plots at \( t=0 \) and at \( t=25 \) are displayed. The particle crossings show up here as wave-breaking.
Chapter 5

TWO STREAM INSTABILITIES

5.1 INTRODUCTION

In this chapter we examine the modelling of the instability that develops when two equal streams of electrons flow through each other. The temporal behaviour of this "two-stream instability" may be explained as follows, [13,22]. As the two electron beams pass through each other, they begin to interact because the density perturbation on one stream is reinforced by the forces due to the other stream and vice versa. Since \( \Delta n_1 \propto n_1 \) the perturbation grows exponentially in time[23]. As the perturbation is growing, a large potential well forms which traps the electrons. The trapped electrons then oscillate in this potential well at the bounce frequency, \( \omega_b \), given by [5],

\[
\omega_b = \sqrt{\frac{qke}{m}},
\]

with \( k \) equal to the wavenumber and \( E \) the electric field amplitude. The growth continues until the electric field amplitude becomes large enough for the bounce frequency, \( \omega_b \), of the trapped electrons to become comparable to the growth rate. At this time the growth saturates due to the non-linear interaction of the trapped electrons with the wave. This non-linear interaction may be described as follows. When the trapped electrons reach their
minimum velocity, the wave reaches its peak amplitude. As the electrons regain velocity (rotate in phase space), conservation of energy dictates that the wave give up energy and damp. As the electrons continue rotating in phase space, energy is traded back and forth between the beam and the wave and so the wave energy oscillates about some mean value.

The two-stream instability has been studied extensively in connection with noise in the solar corona, enhanced diffusion in plasma confinement devices, and as a source for heating plasmas to thermonuclear temperatures [17, 19, 20, 21]. It is an ideal test for a relativistic simulation model since the linear growth rates in both the non-relativistic and the relativistic regimes may be compared to those obtained by numerically solving the dispersion relation. In our simulations we look at both cold and warm beams that have drift velocities ranging from 0.04c (non-relativistic), to 0.5c (very relativistic).

5.2 THEORY

First we will review the theory for non-relativistic cold beams. In chapter 3 we showed that the dispersion relation for cold beams is given by

\[ k^2 = \sum_j \frac{\omega_{p_j}^2}{\gamma_j^3} \frac{1}{(v_j - \omega/k)^2}. \] (5-1)

When we have two species present, this becomes (in the non-relativistic limit),

\[ 1 = \frac{\omega_{p1}^2}{(\omega - kv_1)^2} + \frac{\omega_{p2}^2}{(\omega - kv_2)^2}. \] (5-2)
We will look at two equal density beams travelling with equal and opposite velocities 
\( \omega_{p1} = \omega_{p2} = \omega_p \) and \( v_1 = -v_2 = v_0 \). Equation (5-2) becomes

\[
1 = \frac{\omega_p^2}{(\omega - kv_0)^2} + \frac{\omega_p^2}{(\omega + kv_0)^2} \quad (5-3)
\]

Now we assume that we have complex \( \omega \) and real \( k \) (i.e. there is growth in time but not in space). The dispersion relation, equation (5-3), may be solved for \( \omega \) to yield

\[
\omega = \pm \left[ k^2 v^2 + \omega_p^2 \pm \omega_p (4k^2 v^2 + \omega_p^2)^{1/2} \right]^{1/2} \quad (5-4)
\]

This equation is quartic in \( \omega \) with four independent solutions. For \( 0 < kv/\omega_p < \sqrt{2} \) there are two real and two imaginary roots. For \( kv_0/\omega_p > \sqrt{2} \) there are four real roots.

Equation (5-4) was solved numerically and the results are plotted in Figure 5-1. Appendix A displays a listing of the program used to solve this equation, along with a compilation of the positive imaginary root for \( kv/\omega_p = 0 \) to \( kv/\omega_p = \sqrt{2} \).

Assuming that the perturbation is in mode 1, we can see from this figure that growth will occur whenever

\[
\frac{\omega_p L}{v_0} > \frac{2\pi}{\sqrt{2}} \quad (5-5)
\]

The maximum growth rate is given by:

\[
\gamma_{\text{max}} = \omega_{\text{imaginary}} = \frac{\omega_p}{2} \quad (5-6)
\]

When dealing with relativistic velocities the analysis is identical to that just done except that \( \omega_p \) must be replaced by \( \omega_r \), the longitudinal relativistic plasma frequency,
Figure 5–1 DISPERSION RELATION for the two-stream instability.

where

$$\omega_{\tau j}^2 = \omega_p^2 / \gamma_j^3$$, \hspace{1cm} (5–7)

is the relativistic longitudinal plasma frequency, and \(\gamma_j\) is defined by:

$$\gamma_j = (1 - v_j^2/c^2)^{-1/2}.$$ \hspace{1cm} (5–8)
5.3 TRIAL RUNS AND RESULTS

5.3.1 Non-relativistic Two stream Instability (Cold Beams)

Table V-1 displays the input parameters for run no. 121112 (restart no. 181010). This run used two cold electron beams that were travelling at velocities of \( \pm 0.04c \). Both beams were given an initial density perturbation in mode 1 (i.e. the perturbation wavelength equalled the system length).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>4( \pi )</td>
<td>System length</td>
</tr>
<tr>
<td>NG</td>
<td>32</td>
<td>Number of grid points</td>
</tr>
<tr>
<td>DX</td>
<td>0.3927</td>
<td>L/NG</td>
</tr>
<tr>
<td>DT</td>
<td>0.10</td>
<td>Timestep size</td>
</tr>
<tr>
<td>NT</td>
<td>2500</td>
<td>Number of timesteps</td>
</tr>
<tr>
<td>K0</td>
<td>0.5</td>
<td>2( \pi )/fundamental wavelength</td>
</tr>
<tr>
<td>N1 = N2</td>
<td>512</td>
<td>Number of particles for each species</td>
</tr>
<tr>
<td>X11 = X21</td>
<td>0.02</td>
<td>Position perturbation</td>
</tr>
<tr>
<td>V10 = -V20</td>
<td>0.04</td>
<td>Drift velocity</td>
</tr>
<tr>
<td>WP1 = WP2</td>
<td>0.10</td>
<td>Plasma frequencies</td>
</tr>
<tr>
<td>WC1 = WC2</td>
<td>0.0</td>
<td>Cyclotron frequencies</td>
</tr>
<tr>
<td>TEMP1 = TEMP2</td>
<td>0.0</td>
<td>Species temperatures</td>
</tr>
</tbody>
</table>

**TABLE V-1** INPUT PARAMETERS. Run no. 121112.

Using these values we may calculate some important system constants as follows:

\[
\text{light velocity, } c = \frac{\text{DX}}{\text{DT}} = \frac{0.3927}{0.1} = 3.927 \text{ system units}
\]

\[
v_0 = 0.04c = 0.04(3.927) = 0.1571 \text{ system units}
\]
We may now use equation 5-5 to check that the system is unstable. We have

\[ \frac{\omega_i L}{v_0} = \frac{(0.10)(4\pi)}{0.1571} = 1.273 \ (2\pi) , \]

which is greater than \( 2\pi/\sqrt{2} \) and so we expect growth. To calculate the theoretical growth rate we first determine

\[ \frac{k v_0}{\omega_p} = \frac{(0.5)(0.1571)}{0.10} = 0.786 . \]

We now use figure 5-1, (or Appendix A), to determine

\[ \frac{\omega_i}{\omega_p} \approx 0.49 \]

from which the theoretical growth rate, \( \gamma_{th} \), is:

\[ \gamma_{th} = \omega_i = 0.10(0.49) = 0.049 \]

Figures 5-2 and 5-3 display the electrostatic energies for modes 1, 2, and 3. From Figure 5-2 the measured growth rate is \( \gamma_{meas} = 0.049 \pm 0.003 \), which is in excellent agreement with the theoretical result.

Using a value for the electric field energy where the linear growth just starts to be modified, we may calculate the bounce frequency, \( \omega_b \), to be:

\[ \omega_b = \sqrt{\frac{|q| k E}{m}} = \sqrt{0.5 \ (3 \times 10^{-5})^{1/2}} \approx 0.05 \]

\( (q/m = -1.0) \), which is comparable to the linear growth rate. After the electrostatic field energy saturates (at 36.9% of the total energy) it oscillates with \( \omega_b \approx 0.09 \), (as measured
on figure 5-2). Using an approximate value for the electric field energy at saturation we calculate

\[ \omega_b = \sqrt{\frac{|q| k E}{m}} = \sqrt{0.5 \left(2 \times 10^{-4}\right)^{1/2}} \approx 0.08 \]

which is in reasonable agreement with the measured value of 0.09.

Another point of interest may be observed from figures 5-3. Initially modes two and three are outside the region of growth. However, as the mode 1 energy increases, harmonics are produced via non-linear coupling[4] which causes modes 2 and 3 to grow at twice and three times the growth rate of mode 1. The measured growth rates were 0.0984 and 0.143 for modes two and three respectively (2.01 and 2.92 times the mode one rate).

The maximum variation in the total energy was less than 3% from the initial value. This is much less than the field energy variation which went from almost zero to 37% of the total energy.

The evolution of the run in phase space is depicted in Figure 5-4. The formation of the vortex, which is indicative of trapping, is clearly evident. Figure 5-5 shows the distribution function, \( f(v_x) \), at \( t=0 \) and at \( t=250 \) (after saturation). The thermal spread acquired by the particles is clear even though the small number of particles used (512 per stream) does make the plot "spikey".

The total electron kinetic energy and the \( x \)-momentum for one beam are displayed in Figure 5-6. The kinetic energy falls off at saturation and then oscillates as the energy of the trapped particles is exchanged with the field. The momentum of the first beam drops off sharply at saturation and then oscillates weakly.
5.3.2 Non-relativistic Two Stream Instability (Warm Beams)

In this section, we look at a run (#92919) which is identical to the previous one (#121112) except that two warm beams were used; \( \text{TEMP1} = \text{TEMP2} = 0.0001 \), where \( \text{TEMP1} \) and \( \text{TEMP2} \) are given in units of the rest mass energy of species 1 (i.e. in units of 0.511 MeV).

Figure 5-7 displays the electrostatic energy for mode 1. The theoretical growth rate for this case is the same as for the previous case, i.e. \( \gamma_{\text{th}} = 0.049 \), and the measured growth rate from fig. 5-7 is \( \gamma_{\text{meas}} = 0.045 \pm 0.001 \). Thus it appears that switching to warm beams has slightly reduced the growth rate. No further trials to quantify the effect of the beam thermal spread was done since these runs require the use of a great many particles and timesteps (i.e. CC$\S$). Stringer[24] shows that the threshold for growth for the electron-electron two stream instability is \( v_{\text{drift}} \geq 1.3 v_{\text{thermal}} \). In our model system, the temperature of 0.0001 corresponds to a thermal velocity of 0.01c, \( (\sqrt{\text{TEMP1}} = v_{\text{th}}/c) \), so that we have \( .04 > .01 \) and Stringer's condition for growth is met. However, even though we are above threshold, the ratio of the drift velocity to the thermal velocity is small enough that it causes the growth rate to be slightly decreased.

Other differences between the cold and warm beam cases were also noticed. The maximum electrostatic field energy at saturation was \( \approx 7 \% \) of the total energy as compared to \( \approx 37 \% \) for the cold beam case. However, the maximum variation in the total energy from its initial value for the warm beam case also dropped, (to < 0.05 %), so the variation in the electrostatic energy was still very large in comparison. Another difference was in the oscillation amplitude of the mode 1 energy after saturation. It is much lower in warm
beam case than with the cold beams. These results appear reasonable since with a larger spread in velocities the energy exchange between the particles and the wave would be less coherent.

Figure 5-8 displays the electrostatic energies for modes 2 and 3. We see that the growth in these two modes is much less pronounced than in the cold beam case. However, the growth rates are still roughly 2 and 3 times the mode one rate for modes two and three respectively.

As the two beams interact, large fields are generated (see figures 5-9 and 5-10) which scatter the particles in phase space—forming a vortex where the particles are trapped and increasing the temperature of the beams. In Figure 5-11, the evolution of the warm two stream instability in phase space is presented. These plots clearly show the vortex development. In Figure 5-12, the distribution function, \( f(v_x) \), is displayed at \( t=0 \) and at \( t=275 \) (after saturation). The temperature increase in the beams is readily apparent.

### 5.3.3 Relativistic Two Stream Instability (Cold Beams)

A number of runs were made with different beam velocities and at various values of \( kv/\omega_r \). The results of these runs, and also the non-relativistic runs from the previous section are summarized in Table V-2. Excellent agreement between theoretical and measured growth rates were observed in all cases. Without relativistic corrections, the theoretical growth rates differed by as much as 36% from the measured rates. With the relativistic corrections, the rates agree to within a few percent in all cases.
The above runs verify that the simulation code is indeed working properly in both the non-relativistic and relativistic regimes. The measured growth rates and frequencies were both found to be in excellent agreement with the theoretical results. Also, the total energy variation was found to be small in comparison to the change in the electrostatic energy.

We are now ready to move on to our final test on ESR1—Landau damping.
### RESULTS — NON-RELATIVISTIC TWO STREAM INSTABILITY RUNS

<table>
<thead>
<tr>
<th>Run No.</th>
<th>$V_{10}$ (c)</th>
<th>light speed, c</th>
<th>$\omega_p$</th>
<th>$k v_0 / \omega_p$</th>
<th>$\gamma_h$</th>
<th>$\gamma_{meas}$</th>
<th>$\Delta \gamma$ (%)</th>
<th>Field Energy as % of T.E. at sat'n</th>
<th>Sat'n Time $\omega_p^{-1}$</th>
<th>$\Delta E$ %</th>
</tr>
</thead>
<tbody>
<tr>
<td>121112</td>
<td>.04</td>
<td>3.9270</td>
<td>.10</td>
<td>.786</td>
<td>.049</td>
<td>.049</td>
<td>&lt;1</td>
<td>37</td>
<td>22</td>
<td>3.0</td>
</tr>
<tr>
<td>92919</td>
<td>.04</td>
<td>3.9270</td>
<td>.10</td>
<td>.786</td>
<td>.049</td>
<td>.044</td>
<td>-10</td>
<td>8</td>
<td>23</td>
<td>.5</td>
</tr>
</tbody>
</table>

### RESULTS — RELATIVISTIC TWO STREAM INSTABILITY RUNS

<table>
<thead>
<tr>
<th>Run No.</th>
<th>$V_{10}$ (c)</th>
<th>light speed, c</th>
<th>$\omega_p$</th>
<th>$k v_0 / \omega_r$</th>
<th>$\gamma_h$</th>
<th>$\gamma_{meas}$</th>
<th>$\Delta \gamma$ (%)</th>
<th>Field Energy as % of T.E. at sat'n</th>
<th>Sat'n Time $\omega_p^{-1}$</th>
<th>$\Delta E$ %</th>
</tr>
</thead>
<tbody>
<tr>
<td>234123</td>
<td>.10</td>
<td>1.9635</td>
<td>.10</td>
<td>.989</td>
<td>.048</td>
<td>.048</td>
<td>&lt;1</td>
<td>38</td>
<td>23</td>
<td>.9</td>
</tr>
<tr>
<td>110331</td>
<td>.10</td>
<td>1.9635</td>
<td>.20</td>
<td>.495</td>
<td>.080</td>
<td>.081</td>
<td>&lt;2</td>
<td>30</td>
<td>25</td>
<td>1.2</td>
</tr>
<tr>
<td>135323</td>
<td>.25</td>
<td>1.9635</td>
<td>.20</td>
<td>1.29</td>
<td>.062</td>
<td>.061</td>
<td>&lt;1</td>
<td>38</td>
<td>35</td>
<td>.6</td>
</tr>
<tr>
<td>131827</td>
<td>.25</td>
<td>1.9635</td>
<td>.25</td>
<td>1.03</td>
<td>.114</td>
<td>.114</td>
<td>&lt;1</td>
<td>37</td>
<td>24</td>
<td>.6</td>
</tr>
<tr>
<td>143653</td>
<td>.25</td>
<td>1.9635</td>
<td>.50</td>
<td>.515</td>
<td>.197</td>
<td>.191</td>
<td>&lt;3</td>
<td>28</td>
<td>26</td>
<td>1.4</td>
</tr>
<tr>
<td>123455</td>
<td>.50</td>
<td>1.9635</td>
<td>.50</td>
<td>1.22</td>
<td>.156</td>
<td>.158</td>
<td>&lt;2</td>
<td>32</td>
<td>34</td>
<td>.6</td>
</tr>
<tr>
<td>142644</td>
<td>.50</td>
<td>3.9270</td>
<td>1.0</td>
<td>1.22</td>
<td>.312</td>
<td>.313</td>
<td>&lt;.5</td>
<td>33</td>
<td>34</td>
<td>0.9</td>
</tr>
<tr>
<td>100448</td>
<td>.50</td>
<td>1.9635</td>
<td>1.0</td>
<td>.609</td>
<td>.365</td>
<td>.358</td>
<td>-2</td>
<td>25</td>
<td>28</td>
<td>.9</td>
</tr>
</tbody>
</table>

Table V-2 TWO STREAM INSTABILITY RESULTS. Unless otherwise specified, all quantities are in system units.
Figure 5-2 ELECTROSTATIC ENERGY - MODE 1. Run #121112. Non-relativistic electron-electron two stream instability with cold beams.
Figure 5-3 ELECTROSTATIC ENERGY – MODES 1 and 2. Run #121112. Non-relativistic electron-electron two stream instability with cold beams.
Figure 5-4 EVOLUTION in PHASE SPACE. Run #121112. Non-relativistic electron-electron two stream instability with cold beams. In the upper figure, the dashed lines are at t=0 and the thick lines are at t=200, just before saturation. The lower plot is at t=500, well after saturation.
Figure 5-5 DISTRIBUTION FUNCTION. Run #121112. Non-relativistic electron-electron two stream instability with cold beams. The thermal spread acquired by the particles is clearly evident. Note the change of scale between the two plots.
Figure 5-6 Electron KINETIC ENERGY and MOMENTUM. Run #121112. Non-relativistic electron-electron two stream instability with cold beams. Note the cutout in the time scale.
Figure 5-7 ELECTROSTATIC ENERGY – MODE 1. Run #92919. Non-relativistic electron-electron two stream instability with warm beams.
Figure 5–8 ELECTROSTATIC ENERGY – MODES 1 and 2. Run #92919. Non-relativistic electron-electron two stream instability with warm beams.
Figure 5–9 CHARGE DENSITY. Run #92919. Non-relativistic electron-electron two stream instability with warm beams. Note the change in scale between the two plots.
Figure 5-10 ELECTROSTATIC POTENTIAL. Run #92919. Non-relativistic electron-electron two stream instability with warm beams. Note the change in scale between the two plots.
Figure 5-11 EVOLUTION in PHASE SPACE. Run #92919. Non-relativistic electron-electron two stream instability with warm beams. Clockwise from top left the plots are at $t=0$, $t=200$ (just before saturation), $t=275$ (after saturation), and $t=300$. 
Figure 5-12 DISTRIBUTION FUNCTION. Run #92919. Non-relativistic electron-electron two stream instability with warm beams. The increase in the temperature of the particles is clearly evident. Note the change of scale between the two plots.
6.1 INTRODUCTION

In this chapter we do a final test on ESR1—Landau damping. Landau damping is the collisionless damping of electrostatic plasma waves. It has its origin in the strong interaction between a plasma wave and those particles that have velocities very close to the phase velocity of the wave. Particles moving slower than the wave phase velocity, \( v_\phi \), take energy from the wave while particles moving faster than \( v_\phi \) tend to give energy to it. In a plasma with a Maxwellian distribution of velocities, there are more particles moving slower than the wave phase velocity than there are moving faster. Thus, there will be a net absorption of energy and the wave will be damped.

In this chapter we will look at Landau damping of both a non-relativistic plasma and a hot, relativistic plasma. These runs provide another check on the relativistic modifications to the program as well as offering an opportunity to test the quiet start loading technique that was employed.
The solution of the dispersion relation for non-relativistic and relativistic Maxwellian distributions was presented previously (cf. chapter 3, Tables III—1 and III—2, respectively), and so will not be repeated here.

### 6.2 TRIAL RUNS and RESULTS

#### 6.2.1 NON-RELATIVISTIC LANDAU DAMPING

The input parameters for run number 101708 are shown in Table VI—1. A quiet start was used. The initial position perturbation was \( X_{11} = 164 \) which corresponds to approximately a one percent perturbation in the density.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>( 4\pi )</td>
<td>System length</td>
</tr>
<tr>
<td>NG</td>
<td>16</td>
<td>Number of grid points</td>
</tr>
<tr>
<td>DX</td>
<td>.7854</td>
<td>L/NG</td>
</tr>
<tr>
<td>DT</td>
<td>0.025</td>
<td>Timestep size</td>
</tr>
<tr>
<td>NT</td>
<td>500</td>
<td>Number of timesteps</td>
</tr>
<tr>
<td>K0</td>
<td>0.5</td>
<td>( 2\pi / \text{fundamental wavelength} )</td>
</tr>
<tr>
<td>N1</td>
<td>16384</td>
<td>Number of electrons</td>
</tr>
<tr>
<td>X11</td>
<td>164</td>
<td>Position perturbation</td>
</tr>
<tr>
<td>V10</td>
<td>0.0</td>
<td>Drift velocity</td>
</tr>
<tr>
<td>WP1</td>
<td>1.0</td>
<td>Plasma frequency</td>
</tr>
<tr>
<td>WC1</td>
<td>0.0</td>
<td>Cyclotron frequency</td>
</tr>
<tr>
<td>TEMP1</td>
<td>0.002</td>
<td>Species 1 temperature</td>
</tr>
</tbody>
</table>

**TABLE VI—1  INPUT PARAMETERS. Run no. 101708.**
The initial distribution function and the electrostatic energy for mode 1 are displayed in Figures 6–1(a,b). The initial distribution was a non-relativistic Maxwellian with a temperature of 0.002 (measured in units of the electron rest mass energy, 0.511 Mev). The initial perturbation damps out after only a couple plasma periods. The damping rate, as measured from fig. 6–1(b), is $\omega_i = 0.365$. From the same plot we also find the real part of the frequency to be $\omega_r \approx 1.7$. The theoretical values may be obtained from Table III–1 after we have determined $k/k_D$. We first determine $v_{th}$:

$$v_{th} = \left(\sqrt{\text{TEMP1}}\right) c = (0.0447) (31.416) = 1.4043,$$

where we have used $c = DX/DT$. We now calculate

$$k_D = \frac{\omega_p}{v_{th}} = \frac{1.0}{1.4043} = 0.7121,$$

from which

$$k/k_D = (0.5) / (0.7121) = 0.7021.$$

Referring to Table III–1 we find

$$\omega_r = 1.68 \quad \text{and} \quad \omega_i = 0.395.$$

Upon comparison with the measured values of

$$\omega_r = 1.7 \quad \text{and} \quad \omega_i = 0.365,$$

we find errors of 1.2% and -7.6% for the real and imaginary parts respectively.
Another run was made that was identical to the previous one except that a random start was used. In this case, (not shown), no damping was observed.

A third non-relativistic Landau damping run was made. This case was very similar to the first except that the number of particles was increased to $\approx 65K$. The input parameters are shown in Table VI–2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$4\pi$</td>
<td>System length</td>
</tr>
<tr>
<td>$NG$</td>
<td>16</td>
<td>Number of grid points</td>
</tr>
<tr>
<td>$DX$</td>
<td>.7854</td>
<td>$L/NG$</td>
</tr>
<tr>
<td>$DT$</td>
<td>0.025</td>
<td>Timestep size</td>
</tr>
<tr>
<td>$NT$</td>
<td>400</td>
<td>Number of timesteps</td>
</tr>
<tr>
<td>$K0$</td>
<td>0.5</td>
<td>$2\pi$/fundamental wavelength</td>
</tr>
<tr>
<td>$N1$</td>
<td>65536</td>
<td>Number of electrons</td>
</tr>
<tr>
<td>$X11$</td>
<td>655</td>
<td>Position perturbation</td>
</tr>
<tr>
<td>$V10$</td>
<td>1.0</td>
<td>Drift velocity</td>
</tr>
<tr>
<td>$WP1$</td>
<td>0.0</td>
<td>Plasma frequency</td>
</tr>
<tr>
<td>$WC1$</td>
<td>0.0</td>
<td>Cyclotron frequency</td>
</tr>
<tr>
<td>$TEMP1$</td>
<td>0.002</td>
<td>Species 1 temperature</td>
</tr>
</tbody>
</table>

**TABLE VI–2** INPUT PARAMETERS. Run no. 124958.

The electrostatic energy for mode 1 is shown in figure 6–2. The measured values from this plot are

$$\omega_r = 1.66 \quad \text{and} \quad \omega_i = 0.409.$$ 

Upon comparison with the calculated theoretical values of

$$\omega_r = 1.68 \quad \text{and} \quad \omega_i = 0.395,$$
we find errors of $-1.2\%$ and $3.5\%$ for the real and imaginary parts respectively. Increasing the number of particles decreases the noise level so this improvement over the previous case is expected. The improvement is very noticeable on the mode 1 energy plot where we find that with 65K particles the damping rate was measurable for almost twice as long as in the case with 16K particles.

### 6.2.2 RELATIVISTIC LANDAU DAMPING RUNS

Table VI-3 displays the input parameters for run number 164028.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>$4\pi$</td>
<td>System length</td>
</tr>
<tr>
<td>NG</td>
<td>32</td>
<td>Number of grid points</td>
</tr>
<tr>
<td>DX</td>
<td>0.3927</td>
<td>$L/NG$</td>
</tr>
<tr>
<td>DT</td>
<td>0.078</td>
<td>Timestep size</td>
</tr>
<tr>
<td>NT</td>
<td>200</td>
<td>Number of timesteps</td>
</tr>
<tr>
<td>K0</td>
<td>0.5</td>
<td>$2\pi$/fundamental wavelength</td>
</tr>
<tr>
<td>N1</td>
<td>32768</td>
<td>Number of electrons</td>
</tr>
<tr>
<td>X11</td>
<td>328</td>
<td>Position perturbation</td>
</tr>
<tr>
<td>V10</td>
<td>0.0</td>
<td>Drift velocity</td>
</tr>
<tr>
<td>WP1</td>
<td>1.0</td>
<td>Plasma frequency</td>
</tr>
<tr>
<td>WC1</td>
<td>0.0</td>
<td>Cyclotron frequency</td>
</tr>
<tr>
<td>TEMP1</td>
<td>0.04</td>
<td>Electron temperature</td>
</tr>
</tbody>
</table>

**TABLE VI-3** INPUT PARAMETERS. Run no. 164028.

The temperature in this run corresponds to $u_{th}/c = 0.20$. Figures 6-3(a,b) show the initial distribution function and density perturbation (approximately 1% of the maximum).
The mode 1 electrostatic energy is displayed in figures 6-4(a,b). The real and imaginary parts of the frequency as measured from this plot are

\[ \omega_r = 1.35 \quad \text{and} \quad \omega_i = 0.0924. \]

Upon comparison with the calculated theoretical values of

\[ \omega_r = 1.369 \quad \text{and} \quad \omega_i = 0.0893, \]

we find errors of 1.4% and 3.5% for the real and imaginary parts respectively.

For all the Landau damping runs considered, the damping rates and frequencies were in very good agreement with theory. Additionally, it was observed that the quiet start technique employed works properly and could be very useful in some cases for reducing the number of simulation particles required. When using it, noise levels were low enough that Landau damping could be observed with as few as 16K particles.
Figure 6–1 (a,b) DISTRIBUTION FUNCTION and MODE 1 ENERGY. Run #101708. Non-relativistic Landau Damping, 16384 particles.
Figure 6-2 ELECTROSTATIC ENERGY—MODE 1. Run #124958. Non-relativistic Landau Damping, 65536 particles.
Figure 6-3(a,b) DISTRIBUTION FUNCTION and DENSITY PERTURBATION. Run #164028. Relativistic Landau Damping, 32768 particles.
Figure 6–4(a,b) ELECTROSTATIC ENERGY—MODE 1. Run #164028. Relativistic Landau Damping, 32768 particles.
The objective of this work was to construct and test a 1-1/2 dimensional electrostatic, relativistic, plasma simulation code. This objective has been met. The simulation model known as ESR1 was constructed and implemented on the UBC FPS array processor. Following implementation, the code was used to simulate three well-known plasma problems in order to thoroughly evaluate it in preparation for use in simulating new phenomena.

The first trial runs were on cold plasma oscillations. The variation of frequency with mode number was checked in order to determine if the grid was having the predicted effect on the dispersion relation. The simulation showed clearly the expected variation in frequency. Energy conservation was also checked in this run. Since the code is not energy conserving (it is momentum conserving) we did not expect the total energy to be constant. However, in order for the simulation results to be valid, the change in the total energy should be much less than the variation in the field energy. This is indeed what the simulation showed. Typically, the maximum variation in the total energy was less than 0.7 percent, while the electrostatic energy varied from 0 to 100 percent of the total energy. The cold plasma oscillation runs also provided a simple problem in which the electric field, electric potential and the charge density could be checked for the proper amplitudes and
phases. One run was also made with a large initial perturbation in order to investigate wave breaking. As expected, the mode energies were no longer independent and wave breaking showed up on the phase space plots.

The next test for the code was to make various electron-electron two stream instability runs. These runs were made with both cold and warm beams using drift velocities that varied from non-relativistic to highly relativistic. In all the cases studied, excellent agreement between the measured growth rates from the trial runs and theoretical growth rates were obtained. These runs provided a very good check on the relativistic modifications to the code since the expected growth rates using non-relativistic theory gave values that were up to 36 percent different from the relativistic theory. Energy variation was also checked in these runs and found to be very good.

The third major test for the code was the Landau damping runs. Like the two stream instability runs these were done in both the relativistic and non-relativistic regimes. Damping rates, oscillation frequencies and energy variation were again compared to theoretical results and found to be in very good agreement. The run with 16,384 particles also provided a good check on the quiet start loading technique employed. In a random start using the same number of particles, no damping was observed.

In summary, ESR1, a 1-1/2 dimensional electrostatic, relativistic, plasma simulation code has been constructed, implemented on the UBC FPS array processor, and thoroughly tested. It may now be used for research into various electrostatic problems. It could also be modified to make it fully electromagnetic, in which case it would be useful for simulating many additional processes including laser-plasma interactions.
References


In chapter 5, the dispersion relation for the two-stream instability was presented. It is given by:

$$\omega = \pm \left[ k^2 v^2 + \omega_p^2 \pm \omega_p \left( 4k^2 v^2 + \omega_p^2 \right)^{1/2} \right]^{1/2}.$$ \hfill (A-1)

This equation is quartic in $\omega$ with four independent solutions. For $0 < kv/\omega_p < \sqrt{2}$ there are two real and two imaginary roots. For $\sqrt{2} < kv/\omega_p$ there are four real roots.

Equation (A-1) was solved numerically and the results are plotted in Figure A-1.

From this figure it is clear that growth will occur whenever

$$\frac{\omega_p L}{v_0} > \frac{2\pi}{\sqrt{2}}.$$ \hfill (A-2)

The maximum growth rate is

$$\gamma_{max} = \omega_{\text{imaginary}} = \omega_p/2.$$ \hfill (A-3)
Figure A-1 DISPERSION RELATION for the two-stream instability. Only the positive imaginary root is shown.

Figures A-2(a,b) display a listing of the positive imaginary root for $k v / \omega_p = 0$ to $k v / \omega_p = \sqrt{2}$.

Figures A-3(a,b) are a listing of the program ROOTER that was used to solve equation A-1.
**Figure A-2a** DISPERSION RELATION. Solution of equation A-1 for \( k\nu / \omega_p = 0 \) to \( k\nu / \omega_p = \sqrt{2} \). Only the positive imaginary root is shown.
## Appendix A: TWO-STREAM INSTABILITY + DISPERSION RELATION

<table>
<thead>
<tr>
<th>$k v / \omega_p$</th>
<th>$\omega / \omega_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.500000000</td>
<td>0.296179533</td>
</tr>
<tr>
<td>1.512499803</td>
<td>0.318828583</td>
</tr>
<tr>
<td>1.524999619</td>
<td>0.340403020</td>
</tr>
<tr>
<td>1.537499428</td>
<td>0.361906025</td>
</tr>
<tr>
<td>1.549999237</td>
<td>0.381051838</td>
</tr>
<tr>
<td>1.562500000</td>
<td>0.400381088</td>
</tr>
<tr>
<td>1.574999809</td>
<td>0.41971095</td>
</tr>
<tr>
<td>1.587499619</td>
<td>0.437491655</td>
</tr>
<tr>
<td>1.599999428</td>
<td>0.455398871</td>
</tr>
<tr>
<td>1.612499237</td>
<td>0.472943008</td>
</tr>
<tr>
<td>1.625000000</td>
<td>0.490160525</td>
</tr>
<tr>
<td>1.637499809</td>
<td>0.507086039</td>
</tr>
<tr>
<td>1.649999619</td>
<td>0.523748040</td>
</tr>
<tr>
<td>1.662499428</td>
<td>0.540171385</td>
</tr>
<tr>
<td>1.674999237</td>
<td>0.556377351</td>
</tr>
<tr>
<td>1.687500000</td>
<td>0.572384715</td>
</tr>
<tr>
<td>1.699999809</td>
<td>0.588209927</td>
</tr>
<tr>
<td>1.712499619</td>
<td>0.603867590</td>
</tr>
<tr>
<td>1.724999428</td>
<td>0.619370639</td>
</tr>
<tr>
<td>1.737499237</td>
<td>0.634730518</td>
</tr>
<tr>
<td>1.750000000</td>
<td>0.649957716</td>
</tr>
<tr>
<td>1.762499809</td>
<td>0.665061474</td>
</tr>
<tr>
<td>1.774999619</td>
<td>0.680050194</td>
</tr>
<tr>
<td>1.787499428</td>
<td>0.694931507</td>
</tr>
<tr>
<td>1.799999237</td>
<td>0.709712327</td>
</tr>
<tr>
<td>1.812500000</td>
<td>0.724399030</td>
</tr>
<tr>
<td>1.824999809</td>
<td>0.738997340</td>
</tr>
<tr>
<td>1.837499619</td>
<td>0.753512502</td>
</tr>
<tr>
<td>1.849999428</td>
<td>0.767549462</td>
</tr>
<tr>
<td>1.862499237</td>
<td>0.782312632</td>
</tr>
<tr>
<td>1.875000000</td>
<td>0.796606123</td>
</tr>
<tr>
<td>1.887499809</td>
<td>0.810837552</td>
</tr>
<tr>
<td>1.899999619</td>
<td>0.824999034</td>
</tr>
<tr>
<td>1.912499428</td>
<td>0.839105308</td>
</tr>
<tr>
<td>1.924999237</td>
<td>0.853155549</td>
</tr>
<tr>
<td>1.937500000</td>
<td>0.867152572</td>
</tr>
<tr>
<td>1.949999809</td>
<td>0.881099045</td>
</tr>
<tr>
<td>1.962499619</td>
<td>0.894997418</td>
</tr>
<tr>
<td>1.974999428</td>
<td>0.908850014</td>
</tr>
<tr>
<td>1.987499237</td>
<td>0.922658980</td>
</tr>
</tbody>
</table>

**Figure A-2b** Continuation of Fig. A-2a. Solution of the dispersion relation, equation A-1, for $kv/\omega_p = 0$ to $kv/\omega_p = \sqrt{2}$. Only the positive imaginary root is shown.
Appendix A: TWO-STREAM INSTABILITY\textsuperscript{*} DISPERSION RELATION

C---------------------------------------------------------------------
C PROGRAM ROOTER
C---------------------------------------------------------------------
C COMPLEX*16 FN, E4, Z(1), WP, K, V, VHOLD
INTEGER NR, MAXIT, IND, IFLAG
REAL*8 E1, E2, E3
REAL X(320), Y1(320), Y2(320), Y3(320), Y4(320)
LOGICAL CONJ
EXTERNAL FN
COMMON /PARAM/ WP, FN, E4, Z(1), WP, K, V, VHOLD, NR, MAXIT, IND, IFLAG

DATA MAXIT / 5000 /
DATA Z / (3.0D0, 0.0D0) /
DATA E4 / (1.0D-6, 1.0D-6) /
DATA CONJ / .FALSE. /
DATA E1 / 1.00000D-6/
DATA E2 / 1.00000D-6/
DATA E3 / 1.00000D-6/
DATA NPTS / 320 /
IFLAG = 0
CALL DSPDEV ( 'PLOT' )

WRITE(*,FMT='(\"INPUT WP, K, V (COMPLEX)\")\')
READ(*,*) WP, K, V
VHOLD = V
WRITE(*,FMT='(\"WP = \", 1PD9.9, 1PD17.9 \")\') WP
WRITE(*,FMT='(\"K = \", 1PD9.9, 1PD17.9 \")\') K
WRITE(*,FMT='(\"V = \", 1PD9.9, 1PD17.9 \")\') V
WRITE(*,15)
FORMAT(//,3X,'K\*V/WP ','3X,'W/WP ','/
     3X,'------------------------','3X,'/
     DO 100 J=1,NPTS
     CALL DCZFUN (FN,NR,MAXIT,Z,IND,E1,E2,E3,E4,CONJ)
     X(J) = (K*V/WP)
     Y1(J) = ABS(Z(1))/WP
     WRITE(*,FMT=('(3X,F13.9,7X,F13.9)\') X(J),Y1(J))
     V = V + (0.0125D0,0.0D0)
     CONTINUE
     DO 110 J=1,NPTS
     CALL DCZFUN (FN,NR,MAXIT,Z,IND,E1,E2,E3,E4,CONJ)
     X(J) = (K*V/WP)
     Y2(J) = ABS(Z(1))/WP
     WRITE(*,FMT=('(3X,F13.9,7X,F13.9)\') X(J),Y2(J))
     V = V + (0.0125D0,0.0D0)
     CONTINUE
     DO 200 J=1,NPTS
     Y3(J) = -Y1(J)
     Y4(J) = -Y2(J)
     CONTINUE
     CALL PLOT (X, Y1, Y2, Y3, Y4, NPTS )
     STOP
     END

Figure A–3a PROGRAM ROOTER.
FUNCTION FN ( Z )
COMMON /PARAM/ WP, K, V, IFLAG
COMPLEX*16 FN, Z, WP, K, V
IF ( IFLAG .EQ. 0 ) THEN
  FN = Z - SQRT( ( (K**2)*(V**2)+(WP**2) ) +
  WP*SQR( 4*(K**2)*(V**2)+(WP**2) ) )
ELSE
  FN = Z - SQRT( ( (K**2)*(V**2)+(WP**2) ) -
  WP*SQR( 4*(K**2)*(V**2)+(WP**2) ) )
END IF
RETURN
END

SUBROUTINE PLOT ( X, Y1, Y2, Y3, Y4, NPTS )
INTEGER TITLE
REAL X(NPTS), Y1(NPTS), Y2(NPTS), Y3(NPTS), Y4(NPTS)
REAL XAXIS, YAXIS, XORIG, XMAX, YORIG, YMAX
CALL PAGE ( 11.0, 8.500 )
XAXIS = 9.5
YAXIS = 6.0
CALL YAXANG ( 0 )
CALL CROSS
CALL AXINTS
CALL AREA2D ( XAXIS, YAXIS )
CALL XNAME (' TWIN STREAM INSTABILITY', 100. -1.5, 2)
CALL YNAME (' DISPERSION RELATIONS', 100. -1.5, 2)
XORIG = -1.5
XMAX = 1.5
YORIG = -0.75
YMAX = 0.75
IMARK = 0
CALL GRAPH ( XORIG, 1.0, XMAX, YORIG, 1.0, YMAX )
DO 55 J=1,NPTS
  WRITE(*,*) J, X(J), Y1(J), Y2(J), Y3(J), Y4(J)
CALL CURVE ( X, Y1, NPTS, IMARK )
CALL GRID(10,10)
CALL CURVE ( X, Y2, NPTS, IMARK )
CALL CURVE ( X, Y3, NPTS, IMARK )
CALL CURVE ( X, Y4, NPTS, IMARK )
CALL ENDPL ( 0 )
RETURN
END

Figure A-3b Continuation of PROGRAM ROUTER.
B.1 Introduction

This appendix contains information on accessing and using the program ESR1.

The source code for ESR1 is in two main files. The first, ESR1.S contains the source code for the "number crunching" part of the program. The second, ESR1.PLT.S, contains the routines for taking the output from ESR1.S and producing plots. ESR1.S must be compiled using a special version of FORTRAN that will enable it to run on the UBC Floating Point Systems FPS-164 Array Processor. ESR1.PLT.S should be compiled using the normal UBC FortranVS compiler since it runs on the UBC Amdahl 5860.

B.2 Setup Procedure

The basic procedure for setting up program ESR1 for use is as follows:

(i) Create the following files—you may substitute your own file names for those filenames followed by an asterisk:
The other filenames may be changed too if you modify the source code.

(ii) Copy the following files:

```
RAPS:ESR1.S(1000,21999) to ESR1.S  
RAPS:ESR1.S(22000,22999) to ESR1.INPUT  
RAPS:ESR1.S(23000,23999) to ESR1.INSERT  
RAPS:ESR1.PLT.S(1000,8999) to ESR1.PLT.S  
RAPS:ESR1.PLT.S(23000,23999) to ESR1.PLT.IN  
RAPS:ESR1.PLT.S(24000,24999) to ESR1.CLICHE  
```

(iii) Compile ESR1.S with the FPS-164 FORTRAN compiler by using the following statement:

```
RUN AP64:APFTN64 SCARDS=ESR1.S(1000,21999) SPUNCH=-APOBJ ...
... SPRINT=-SPR PAR=OPT(3)
```

(iv) Put the object modules in an object library:

```
RUN AP64:APLIBR64 PAR=INP(ESR1.OLIB.AP),INSERT(-APOBJ)
```

(v) Link the program:

```
RUN AP64:APLINK64 PAR=INPUT(ESR1.OLIB.AP), FORCE(ESR1,ESR1D),...
... IMAGE[ESR1.LD]
```

Details on the use of the FPS-164 and its associated software may be found in the manuals located in the Physics department computer terminal room (Rm. 203).
The following run card should be used when replacing modules in the object library (ESR1.OLIB.AP):

RUN AP64:APLIBR64 PAR=INP(ESR1.OLIB.AP),REPLACE(-APOBJ)

After replacing modules in ESR1.OLIB.AP you must re-link the program using AP64:APLINK64 as above. (Be sure to empty ESR1.LD first!).

B.3 RUNNING ESR1 and ES1PLT

A typical batch file for running ESR1 is shown in figure B-1(a) and a typical input file is shown in figure B-1(b). The equivalent files for program ES1PLT are displayed in figures B-2(a,b).

Figures B-3 and B-4 show the “insert” and “include” files for ESR1 and ES1PLT respectively. Complete program listings for ESR1 and ES1PLT are displayed in figures B-5 and B-6.
$\text{SIG} \text{ RAPS \ TIME}=6 \ \text{PRI0}=N \ \text{APTIME}=599
$\text{$$$$$$}
$\text{LOG \ *MSINK* \ ON \ LOGFILE(*L+1)}
$\text{EMPTY \ ESR1.PRNT \ OK}
$\text{EMPTY \ APOUT \ OK}
$\text{EMPTY \ HIST.AP \ OK}
$\text{RUN \ AP64:SJE}
\text{ATTACH/WAIT}
\text{COPYIN/B \ 'ESR1.LD', \ ESR1}
\text{ESR1}
\text{SHOW/CPU}
\text{COPYOUT/B \ HIST, \ 'HIST.AP'}
\text{COPYOUT/B \ APO, \ 'APOUT'}
\text{COPYOUT \ PRNT, \ 'ESR1.PRNT'}
\text{SHOW/CPU}
\text{DETACH}
\text{QUIT}
\text{$\text{SIG}$}

(a)

= \text{RELATIVISTIC TWO STREAM INSTABILITY =}
&IN1
  K0=0.5D0, NT=500, DT=0.100000D0, 
  VLIGHT=1.0000000D0, 
  NG=128, A1=0.00000D0, A2=00000.00D0, 
  E0=0.0D0, W0=0.0D0, IRHO=0000, IRHOS=00, IPHI=000, 
  IE=0000, ISPEC=0000, MPLOT=1,2,3, 
  IXVX=0000, RSTART=\text{.FALSE.}, ISAVE=499, 
  MODE=1, Itherm=0000, QSTART=\text{.FALSE.}, LSMEAR=\text{.FALSE.},
&END
&IN2
  N1=2048, WP1=3.000000D00, QM1=-1.00000D0, 
  TEMP1=0.0000000000000D00, V10=0.50000000D0, 
  X11= 000.020000000000D00, THETX1=0.00000000D00, WC1=0.000D0, 
&END
&IN3
  N2=2048, WP2=3.0000000D0, QM2=-1.000D0, 
  TEMP2=0.0000000D0, V20=-0.50000000D0, 
  X21=000.0200D0, THETX2=0.0000000000D0, 
&END

(b)

Figure B-1 INPUT FILES. Program ESR1. (a) Typical batch file to run ESR1. (b) Typical input file (ESR1.INPUT) for ESR1.
Appendix B: ESR1 USERS GUIDE+and+PROGRAM LISTING

$SIG RAPS T=20 PRIORITY=normal
$$$$$$ $LOG *MSINK* ON LOGFILE(*L+1)
$EMPTY ESR1.DISp OK
$EMPTY ESR1.PLOT OK
$COPY APOUT -APOUT
$RUN ESR1.PLT.OBJ+ESR1.OLIB+*DISSPLA9 2=ESR1.PLT.IN(200,218)
7=ESR1.MNMX 6=ESR1.DISp 9=-PLOTZ
$CONTROL *QMS* DELIVERY=PHYS PACKAGE=LOOSE
$RUN *QMSPLLOT 0=-PLOTZ
$REL *QMS*
$CONTROL *PRINT* CANCEL
$COPY -PLOTZ TO ESR1.PLOT
$SIG

(a)

&IN
NPTCL1=000, NPTCL2=0, NPTCLT=000, IFVX01=.FALSE., IFVY01=.FALSE., IFVX1=000, IFVY1=0,
IFVX02=.FALSE., IFVY02=.FALSE., IFVX2=00, IFVY2=0,
IFVX0T=.FALSE., IFVXT=0000, ITMP01=.FALSE., ITMP1=0000,
ITMP2=.FALSE., ITMP2=00,
IXVX1=000, IXVY1=000, IVXVY1=000, IXVX2=0, IXVY2=0, IVXVY2=0,
IXVXT=0000, IXVYT=0, IXVXVYT=0, IRHOS0=.FALSE., IRHOS1=000, IH01=0000, IE0=0000,
IRHO0=.FALSE., IRHO1=0000, ISPEC0=.FALSE., ISPEC1=000,
ITE=.FALSE., IKE1=.FALSE., IELKE=.FALSE., IESE=.FALSE., IESEM=.TRUE., IXT=.FALSE.,
IP1X=.FALSE., IP1Y=.FALSE., IP2X=.FALSE., IP2Y=.FALSE.,
LHSTRY=.TRUE., LFIELD=.FALSE., LSPACE=.FALSE.,
JFLAG=1,
XXORIG=0.00D0, XXMAX=6.2831850D0,
YYORIG=-5.000000D0, YYMAX=5.000000D0
&END
(b)

Figure B–2 INPUT FILES. Program ES1PLT. (a) Typical batch file to run ES1PLT.
(b) Typical input file (ESR1.PLT.IN) for ES1PLT.
### Appendix B: ESR1 USERS GUIDE and PROGRAM LISTING

**VARIABLE PLOT**

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>PLOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>LHSTRT</td>
<td>if .TRUE. - make history plots.</td>
</tr>
<tr>
<td>LFIELD</td>
<td>if .TRUE. - make field plots.</td>
</tr>
<tr>
<td>LSPACE</td>
<td>if .TRUE. - make phase space plots.</td>
</tr>
<tr>
<td>NPTCL1</td>
<td>electrons - dist'bn in space</td>
</tr>
<tr>
<td>NPTCL2</td>
<td>species 2 - dist'bn in space</td>
</tr>
<tr>
<td>NPTCLT</td>
<td>all species - dist'bn in space</td>
</tr>
<tr>
<td>IFVX01</td>
<td>electrons - x-velc'y dist'bn @ t=0.</td>
</tr>
<tr>
<td>IFVY01</td>
<td>electrons - y-velc'y dist'bn @ t=0.</td>
</tr>
<tr>
<td>IFVX1</td>
<td>electrons - x-velc'y dist'bn.</td>
</tr>
<tr>
<td>IFVY1</td>
<td>electrons - y-velc'y dist'bn.</td>
</tr>
<tr>
<td>IFVX02</td>
<td>species 2 - x-velc'y dist'bn @ t=0.</td>
</tr>
<tr>
<td>IFVY02</td>
<td>species 2 - y-velc'y dist'bn @ t=0.</td>
</tr>
<tr>
<td>IFVX2</td>
<td>species 2 - x-velc'y dist'bn.</td>
</tr>
<tr>
<td>IFVY2</td>
<td>species 2 - y-velc'y dist'bn.</td>
</tr>
<tr>
<td>IFVXT</td>
<td>all species - x-velc'y dist'bn.</td>
</tr>
<tr>
<td>ITMP01</td>
<td>electrons - temp profile @ t=0.</td>
</tr>
<tr>
<td>ITMP1</td>
<td>electrons - temp profile.</td>
</tr>
<tr>
<td>ITMP02</td>
<td>species 2 - temp profile @ t=0.</td>
</tr>
<tr>
<td>ITMP2</td>
<td>species 2 - temp profile.</td>
</tr>
<tr>
<td>IXVX1</td>
<td>electrons - x-vx phase space.</td>
</tr>
<tr>
<td>IXVY1</td>
<td>electrons - x-vy phase space.</td>
</tr>
<tr>
<td>IXVXY1</td>
<td>electrons - vx-vy space every.</td>
</tr>
<tr>
<td>IXVX2</td>
<td>species 2 - x-vx phase space.</td>
</tr>
<tr>
<td>IXVY2</td>
<td>species 2 - x-vy phase space.</td>
</tr>
<tr>
<td>IXVXT2</td>
<td>species 2 - vx-vy space.</td>
</tr>
<tr>
<td>IXVXT</td>
<td>all species - vx-vx phase space.</td>
</tr>
<tr>
<td>IXVYT</td>
<td>all species - vx-vy phase space.</td>
</tr>
<tr>
<td>IRHO0</td>
<td>charge density.</td>
</tr>
<tr>
<td>IRHOS0</td>
<td>smoothed charge density.</td>
</tr>
<tr>
<td>IPHI0</td>
<td>electric potential.</td>
</tr>
<tr>
<td>IE0</td>
<td>electric field.</td>
</tr>
<tr>
<td>ISPEC0</td>
<td>energy spectrum.</td>
</tr>
<tr>
<td>IRHO1</td>
<td>charge density.</td>
</tr>
<tr>
<td>IRHOS1</td>
<td>smoothed charge density.</td>
</tr>
<tr>
<td>IPHI1</td>
<td>electric potential.</td>
</tr>
<tr>
<td>IE1</td>
<td>electric field.</td>
</tr>
<tr>
<td>ISPEC1</td>
<td>energy spectrum.</td>
</tr>
<tr>
<td>ITE</td>
<td>total energy.</td>
</tr>
<tr>
<td>IKE1</td>
<td>electrons - thermal energy.</td>
</tr>
<tr>
<td>IKE2</td>
<td>species 2 - thermal energy.</td>
</tr>
<tr>
<td>IELKE</td>
<td>electrons - kinetic energy.</td>
</tr>
<tr>
<td>IEESE</td>
<td>electric field mode energies.</td>
</tr>
<tr>
<td>IXT</td>
<td>particle 1 - X position.</td>
</tr>
<tr>
<td>IVXT</td>
<td>particle 1 - X velocity.</td>
</tr>
<tr>
<td>IP1X</td>
<td>electrons - avg. X momentum.</td>
</tr>
<tr>
<td>IP1X</td>
<td>electrons - avg. X momentum.</td>
</tr>
<tr>
<td>IP1X</td>
<td>species 2 - avg. X momentum.</td>
</tr>
<tr>
<td>IP1X</td>
<td>species 2 - avg. Y momentum.</td>
</tr>
</tbody>
</table>

Scales control in field plotting routine (PLOTF):

<table>
<thead>
<tr>
<th>JFLAG</th>
<th>scales set via XXORIG, XXMAX, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>= 0</td>
<td>calc. max &amp; min values for use in PLOTF.</td>
</tr>
<tr>
<td>= 0</td>
<td>scales free. (ie. calc'd by program.)</td>
</tr>
<tr>
<td>XXORIG</td>
<td>X origin.</td>
</tr>
<tr>
<td>XXMAX</td>
<td>X maximum value.</td>
</tr>
<tr>
<td>YYORIG</td>
<td>Y origin.</td>
</tr>
<tr>
<td>YTMAX</td>
<td>Y maximum value.</td>
</tr>
</tbody>
</table>

Figure B-2(cont.) VARIABLE NAMES. Used in ESR1.PLT.IN.
IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
INTEGER ZERO, ONE
PARAMETER ( ZERO = 0, ONE = 1 )
PARAMETER ( DZERO = 0.0D0 )
PARAMETER ( TWOPI = 6.2831 85307 17958 DO )
DOUBLE PRECISION L, M1, M2, KE
COMMON / PARAM / L, DX, DT, IDATE, ITIME
PARAMETER ( NGMAX = 256 )
PARAMETER ( NG1M = NGMAX + 1 )
COMMON / CFIELD / AEL, CGSHL, ROHO, A1, A2,
    RH0(NG1M), PHI(NG1M), E(NG1M),
    EO, WO, NG
REAL ZX, ZV
COMMON / PRTCLS / X (65540), VX (65540), VY (65540),
    ZX(17660), ZV(17660)
COMMON / CNTRL / TIME, IT, ITHL, IRHO, IRHOS, IPHI,
    IE, IXVX, ISPEC
DOUBLE PRECISION KE1, KE2, NM1, NM2
PARAMETER ( NTH = 2500, MMAX = 20 )
PARAMETER ( NTH1 = NTH+1, NTH2 = NTH+2 )
COMMON / CTIME / ESE(NTH1), KE1(NTH1), KE2(NTH1),
    P1X(NTH2), P1Y(NTH2), NM1, NM2,
    P2X(NTH2), P2Y(NTH2), TE(NTH1),
    XT (NTH1), VXT(NTH1), ELKE(NTH1),
    ESEM(NTH1, MMAX), MPLOT(MMAX)
COMMON / THERML / ITHERM
DOUBLE PRECISION M2C2EV
COMMON / UNITS / RNM1C2, RNM2C2, RM1C2I, RM2C2I, M2C2EV
PARAMETER ( IMX = 65540 )
COMMON / REL / GAMMA(IMX)
DOUBLE PRECISION THRNX1 (NG1M), THRNX2 (NG1M),
    VX0SC (NG1M), V2SX (NG1M),
    VY0SC (NG1M)
EQUIVALENCE ( THRNX1, THRNX2, V2SX, PHI )
DIMENSION N(NG1M)

Figure B-3 ESR1.INSERT listing.
Appendix B: ESR1 USERS GUIDE + and + PROGRAM LISTING

104

Figure B-4 ESR1_CLICHE listing.
PROGRAM ESR1 — LISTING
PROGRAM ESR1

$INSERT 'ESR1.INSERT(100.999)'

DOUBLE PRECISION M1C2, M2C2, KO
LOGICAL RSTART, OSTART, LSMEAR

CHARACTER*131 ATITLE
CHARACTER*20 DATER

NAMELIST /IN1/ KO, NT, DT, NG, A1, A2, VLIGHT,
EO, WO, IRHO, IRHOS, IPHI, IE, MPLOT,
IXVX, ITERM, RSTART, ISAVE, ISPEC, MODE,
QSTART, LSMEAR

NAMELIST /IN2/ N1, WP1, QM1, TEMP1, V10, XI1, THETX1, WC1

NAMELIST /IN3/ N2, WP2, QM2, TEMP2, V20, XI2, THETX2

C

C Default input parameters:

DATA WP1, WP2, WC1, WC2 / 2*1.000, 2*0.000 /
DATA QM1, QM2 / -1.000, -1.000 /
DATA V10, V20 / 0.000, 0.000 /
DATA N1, N2 / 128, 128 /
DATA TEMPI, TEMP2 / 0.000, 0.000 /
DATA XI1, XI2, MODE / 2*0.000, 1 /
DATA THETX1, THETX2 / 0.000, 0.000 /
DATA VLIGHT / 1.000 /

OPEN( 2,FILE=':HOSTCHR:ESR1.INPUT',STATUS='OLD')
OPEN( 7,FILE='PRNT')
OPEN(12,FILE='AP0',FORM='UNFORMATTED')
OPEN(11,FILE='HIST',FORM='UNFORMATTED')
OPEN( 3,FILE=':RAPS(RON):SAVE',FORM='UNFORMATTED')

CALL SYSSGETDATE ( DATER )
CALL DATEME ( DATER, IDATE, ITIME )

NT = 150
ITH = 0

C Read title and namelist input.

READ (2,975) ATITLE
READ (2,IN1)
READ (2,IN2)
READ (2,IN3)

C At T=0, zero arrays.

CALL ZEROS

L = TWOP1/KO
DX = L / NG
HDX = 0.500*DX
T1 = WC1*DT / 2.000
NP = N1 + N2
Q1 = WP1*WP1 / (N1*QM1)
M1 = Q1 / QM1
NM1 = N1 * M1
Q2 = DZERO
IF ( N2 .NE. 0 ) Q2 = WP2*WP2 / (N2*QM2)
M2 = Q2 / QM2
T2 = DZERO
IF ( M2 .GT. DZERO ) T2 = M1*T1 / M2
NM2 = N2*M2
N11 = N1+1
NG1 = NG+1
GN = FLDAT(NG)
RHOO = -(N1*Q1 + N2*Q2) / L
IFMT = 50
CGSHL = L

C Scaling in units of rest mass energy density.
C M1C2EV = 0.5110041 DO6 electron volts.
C
M2C2EV = (M2/M1)*0.5110041 DO6
M1C2 = M1*(DX/DT)**2
M2C2 = M2*(DX/DT)**2
RM1C2 = 1.000 / (N1*M1C2)
RM2C2 = 1.000 / (N2*M2C2)
RM2C2I = N2 / 1024
RNM2C2 = 1.0 D0 / M2C2
RNM2C2 = 1.0 D0 / (N2*M2C2)
IRDATE = ZERO
IRTIME = ZERO
NRECS = NP/1024
IF (N2.NE.0) RM2C2I = N2 / 1024
IRDATE = NRECS+1
IRS = 0
IF (RSTART) IRS=1
IF (.NOT. RSTART) GO TO 30

C If restarting, then restore saved quantities.
CALL RESTOR (IRDATE,IRTIME,ITH,M1,M2,K0.01,02,T1,T2,
           N11,NP,VLIGHT)
CALL RITER1 (NTH,MMAX,RSTART,IRDATE,IRTIME,MODE,ATITLE,
             NT,NP,VLIGHT,K0)
CALL RITER2 (M1,01,WP1,WC1,QM1,T1,V10,X11,THETX1,N1,TEMP1,
             M2,02,WP2,WC2,QM2,T2,V20,X21,THETX2,N2,TEMP2)

C Save miscellaneous variables for use in plotting routines.
REWRITE 11
WRITE(11) IT, ITH, ITHL, NP, NG, IDATE, ITIME, IRDATE,
          ITIME, N11, IE, IPHI, IRHO, IRHOS, ITERM,
          IXVX, N1, N2, NT, IRS, ISPEC, MPLLOT
REWRITE 12
WRITE(12) DT, DX, L, TIME, T1, TEMP1, TEMP2
GO TO 140

C Position and velocity loading.
C Electrons.
C
DDX1 = L / N1

DO 40 I = 1,N1
  X(I) = (1-.500)*DDX1
  VX(I) = V10
  VY(I) = DZERO
40

C CALL CREATOR ( V10, ONE, N1, TEMP1, QSTART )
ISEED = 56437
C IF ( LSMEAR ) CALL SMEAR ( X, ONE, N1, ISEED )
C----------------------------------------------
C For ordered velocities, scramble positions to reduce
C correlations. Scrambling is done by bit-reversed counter.
C Compare sorter in CFPFT.
C XS=.000,.100,.010,.110,.001,.101,.011,.111
C (binary fractions).
C----------------------------------------------
C IF ( QSTART ) THEN
1148 XS = DZERO
1149 C
1150 DO 60 I = 1,N1
1151 X(I) = XS * L + 0.5D0 * DDX1
1152 XSI = 1.0D0
1153 50 XSI = 0.5D0 * XSI
1154 XS = XS - XSI
1155 IF ( XS .GE. DZERO ) GO TO 50
1156 XS = XS + 2.0D0*XSI
1157 XSI = 0.5D0 * XSI
1158 DO 60 CONTINUE
1159 60 CONTINUE
1160 ENDIF
C----------------------------------------------
C Add perturbation.
C Loading X(T=0). so no DT/2 correction.
C
1161 DO 70 I = 1, N1
1162 THETA = TWOPI * MODE * X(I) / L
1163 X(I) = X(I) + X11 * COS (THETA + THETX1)
1164 70 CONTINUE
C----------------------------------------------
C Position and velocity loading.
C Species 2.
C
1165 IF ( N11 .LE. NP ) THEN
1166 DDX2 = L / N2
1167 C
1168 DO 80 I = N11, NP
1169 X(I) = (I-.5D0-N1)*DDX2
1170 VX(I) = DZERO
1171 VY(I) = V20
1172 VX(I) = V20
1173 80 CONTINUE
1174 C
1175 CALL CREATR ( V20, N11, NP, TEMP2, QSTART )
1176 ISEED = 39284
1177 IF ( LSMEAR ) CALL SMEAR ( X, N11, NP, ISEED )
C----------------------------------------------
C For ordered velocities, scramble positions
C to reduce correlations.
C
1178 IF ( QSTART ) THEN
1179 XS = DZERO
1180 C
1181 DO 100 I = N11, NP
1182 X(I) = XS * L + 0.5D0 * DDX2
1183 XSI = 1.0D0
1184 90 XSI = 0.5D0 * XSI
1185 XS = XS - XSI
1186 IF ( XS .GE. DZERO ) GO TO 90
1187 XS = XS + 2.0D0*XSI
1188 XSI = 0.5D0 * XSI
1189 DO 100 CONTINUE
1190 100 CONTINUE
1191 ENDIF
C----------------------------------------------
C Add perturbation.
C
1192 DO 110 I = N11, NP
1193 THETA = TWOPI * MODE * X(I) / L
1194 X(I) = X(I) + X21 * COS (THETA + THETX2)
1195 110 CONTINUE
1196 ENDIF
C Save miscellaneous variables for use in plotting routines.
C
REWIND 11
WRITE(11) IT, ITH, ITHL, NP, NG, IDATE, ITIME, IRDATE,
 IRTIME, N11, IE, IPHI, IRHO, IRHOS, ITERM,
 IXVX, N1, N2, NT, IRS, ISPEC, MPLAT
C
REWIND 12
WRITE(12) IT, ITH, ITHL, NP, NG, IDATE, ITIME, IRDATE,
 IRTIME, N11, IE, IPHI, IRHO, IRHOS, ITERM,
 IXVX, N1, N2, NT, IRS, ISPEC, MPLAT
C
C Save X, VX and VY at T=0.
C
JS = 1
JE = MIN(1024,NP)
DO 120 I = 1,NRECS
WRITE(12) ( X(J), J=JS,JE )
JS = JE + 1
JE = JE + 1024
120 CONTINUE
C
JS = 1
JE = MIN(1024,NP)
DO 121 I = 1,NRECS
WRITE(12) ( VX(J), J=JS,JE )
JS = JE + 1
JE = JE + 1024
121 CONTINUE
C
JS = 1
JE = MIN(1024,NP)
DO 122 I = 1,NRECS
WRITE(12) ( VY(J), J=JS,JE )
JS = JE + 1
JE = JE + 1024
122 CONTINUE
C
CALL SETHO ( ONE, N1, Q1 )
CALL SETHO ( N11, NP, Q2 )
CALL FIELDS ( ZERO )
CALL SETV ( ONE, N1, Q1, M1, T1, P1X(1), P1Y(1), VLIGHT )
CALL SETV ( N11, NP, Q2, M2, T2, P2X(1), P2Y(1), VLIGHT )
C
IF ( NT .EQ. 0 ) STOP
C
XT(1) = X(1)
VXT(1) = VX(1)
C
130 CONTINUE
C If making a restart file, here is where you save everything.
C
IF ( (IT/ISAVE)*ISAVE .NE. IT ) GO TO 140
C
CALL SAVEIT ( ITH,M1,M2,Q1,Q2,T1,T2,N11,NP,VLIGHT )
140 CONTINUE
C
CALL ACCEL ( ONE, N1, Q1, M1, T1, P1X(ITH+2), P1Y(ITH+2),
 KE(ITH+1), KE, VLIGHT )
ELKE(ITH+1) = KE/NM1
CALL ACCEL ( N11, NP, Q2, M2, T2, P2X(ITH+2), P2Y(ITH+2), KE2(ITH+1), KE, VLIGHT )

P = P1X(ITH+2) + P2X(ITH+2)
KE = KE/NM1
TE(ITH+1) = KE + ESE(ITH+1)
IF ( IT .EQ. 0 ) THEN
  WRITE(7,850)
ENDIF
GO TO 129
ENDIF
IF ( IT .EQ. 1 ) GO TO 129
IF ( (IT-1)/IFMT*IFMT .EQ. (IT-1) ) THEN
  WRITE(7,825)
  WRITE(7,850)
ENDIF
CONTINUE
TEDEV = (( TE(1)-TE(ITH+1) )/TE(1) ) * 100.0DD
FEPER = ESE(ITH+1)/TE(ITH+1) * 100.0DD
WRITE(7,950) TIME, TE(ITH+1), ESE(ITH+1), KE, P, ELKE(ITH+1), P1X(ITH+2), TEDEV, FEPER
IF ( IXVX .NE. ZERO ) THEN
  IF ( (IT/IXVX) * IXVX .EQ. IT ) THEN
    SAVE: X @ t=IT , VX,VY @ t=IT+.5
    JS = 1
    JE = MIN(1024,NP)
    DO 150 I = 1,NRECS
      WRITE(12) ( X(J), J=JS,JE )
      JS = JE + 1
      JE = JE + 1024
      JE = MIN(JE,NP)
    CONTINUE
    JS = 1
    JE = MIN(1024,NP)
    DO 151 I = 1,NRECS
      WRITE(12) ( VX(J), J=JS,JE )
      JS = JE + 1
      JE = JE + 1024
      JE = MIN(JE,NP)
    CONTINUE
    IF ( T1 .NE. DZERO ) THEN
      JS = 1
      JE = MIN(1024,NP)
      DO 152 I = 1,NRECS
        WRITE(12) ( VY(J), J=JS,JE )
        JS = JE + 1
        JE = JE + 1024
        JE = MIN(JE,NP)
      CONTINUE
      ENDIF
    ENDIF
  ENDIF
ENDIF
IF ( IT .GE. NT ) GO TO 210
IF ( ITH .EQ. NTH ) THEN
  WRITE history arrays to file.
  TL = ITHL * DT
  MTH = IT - ITHL + 1
  MTH1 = MTH + 1
  CALL PEAKS ( MTH )
NREC2 = MTH/1024
IF ( MOD(MTH,1024) .GT. 0 ) NREC2 = NREC2+1
DO 170 KM = 1, MMAX
JSTAR = 1
JEND = MIN(1024, MTH)

KK = MPLOT(KM)
IF (KK .NE. 0) THEN
DO 169 JJ = 1, NREC2
WRITE(11) ( ESEM(J, KM), J = JSTAR, JEND )
JSTAR = JEND + 1
JEND = JEND + 1024
JEND = MIN(JEND, MTH)
CONTINUE
169
ENDIF
170
CONTINUE

C
NREC2 = MTH1/1024
IF (MOD(MTH1, 1024) .GT. 0) NREC2 = NREC2+1
JSTAR = 1
JEND = MIN(1024, MTH)
JEND1 = JEND + 1
IF (JEND1 .GT. 1024) JEND1 = 1024

C
DO 179 JJ = 1, NREC2
IF (NREC2 .NE. 1) THEN
IF (JJ .EQ. NREC2) THEN
IF (JSTAR .EQ. JEND) GO TO 175
ENDIF
ENDIF
WRITE(11) ( KE1(J), J = JSTAR, JEND )
WRITE(11) ( KE2(J), J = JSTAR, JEND )
WRITE(11) ( TE(J), J = JSTAR, JEND )
WRITE(11) ( ESE(J), J = JSTAR, JEND )
WRITE(11) ( ELKE(J), J = JSTAR, JEND )
WRITE(11) ( XT(J), J = JSTAR, JEND1 )
WRITE(11) ( VXT(J), J = JSTAR, JEND1 )
WRITE(11) ( P1X(J), J = JSTAR, JEND1 )
WRITE(11) ( P2X(J), J = JSTAR, JEND1 )
WRITE(11) ( P1Y(J), J = JSTAR, JEND1 )
WRITE(11) ( P2Y(J), J = JSTAR, JEND1 )
JSTAR = JEND + 1
JEND = JEND + 1024
JEND = MIN(JEND, MTH)
IF ((JEND-JSTAR) .LT. 1023) THEN
JEND1 = JEND + 1
ELSE
JEND1 = JEND
ENDIF
179
CONTINUE

C
ITHL = IT

C Last values are now first values for next time interval.

C
ITHL = IT
P1X(1) = P1XL
P1Y(1) = P1YL
P1X(2) = P1X(MTH+1)
P1Y(2) = P1Y(MTH+1)

C
IF (NM2 .NE. DZERO) THEN
P2X(1) = P2XL
P2Y(1) = P2YL
P2X(2) = P2X(MTH+1)
P2Y(2) = P2Y(MTH+1)
ENDIF

C
TE(1) = TE(MTH)
ESE(1) = ESE(MTH)
XT(1) = XT(MTH)
VXT(1) = VXT(MTH)
ELKE(1) = ELKE(MTH)

DO 190 I=2,MTH
   P1X(I+1) = DZERO
   P1Y(I+1) = DZERO
   P2Y(I+1) = DZERO
   P2X(I+1) = DZERO
   TE(I) = DZERO
   ESE(I) = DZERO
   ELKE(I) = DZERO
190 CONTINUE

C       DO 200 KM = 1,MMAX
C       ESEM(1,KM) = ESEM(MTH,KM)
C       DO 200 I = 2,MTH
C       ESEM(I,KM) = DZERO
C       ENDIF

C       IT = IT+1
C       TIME = IT*DT
C       ITH = IT-ITHL

C--- Save time variables.
C
WRITE(12) IT,ITH

C--- Advance positions from IT to IT+1.
C
CALL MOVE ( ONE, N1 , 01 )
CALL MOVE ( N11, NP, 02 )
XT(ITH+1) = X(1)
VXT(ITH+1) = VX(1)

C--- Get fields at time step IT.
C
CALL FIELDS ( ITH )

GO TO 130

C==========================================================
C END OF RUN.
C==========================================================

C--- Particle X is at time NT*DT, VX & VY at (NT-.5)*DT.

210 CONTINUE

C--- Write history arrays to file.

MTH = IT - ITHL + 1
MTH1 = MTH + 1
CALL PEAKS ( MTH )

C
NREC2 = MTH/1024
IF ( MOD(MTH,1024) .GT. 0 ) NREC2 = NREC2+1

C
DO 230 KM = 1,MMAX
JSTAR = 1
JEND = MIN(1024,MTH)
KK = MPLDT(KM)
IF ( KK .NE. 0 ) THEN
   DO 229 JJ = 1,NREC2
      WRITE(11) ( ESEM(J,KM), J=JSTAR,JEND )
   229 CONTINUE
ENDIF
230 CONTINUE

C
NREC2 = MTH1/1024
IF ( MOD(MTH1,1024) .GT. 0 ) NREC2 = NREC2+1
JSTAR = 1
JEND = MIN(1024, MTH)
JEND1 = JEND + 1
IF (JEND1 .GT. 1024) JEND1 = 1024
C
DO 239 JJ = 1, NREC2
IF (NREC2 .NE. 1) THEN
   IF (JJ .EQ. NREC2) THEN
      IF (JSTAR .EQ. JEND) GO TO 235
   ENDIF
ENDIF
WRITE(11) (KE1(J), J=JSTAR, JEND)
WRITE(11) (KE2(J), J=JSTAR, JEND)
WRITE(11) (TE(J), J=JSTAR, JEND)
WRITE(11) (ESE(J), J=JSTAR, JEND)
WRITE(11) (ELKE(J), J=JSTAR, JEND)
WRITE(11) (XT(J), J=JSTAR, JEND1)
WRITE(11) (VXT(J), J=JSTAR, JEND1)
WRITE(11) (P1X(J), J=JSTAR, JEND1)
WRITE(11) (P2X(J), J=JSTAR, JEND1)
WRITE(11) (P1Y(J), J=JSTAR, JEND1)
WRITE(11) (P2Y(J), J=JSTAR, JEND1)
JSTAR = JEND + 1
JEND = JEND + 1024
JEND = MIN(JEND, MTH)
IF ((JEND-JSTAR) .LT. 1023) THEN
   JEND1 = JEND + 1
ENDIF
239 CONTINUE
C
ITHL = IT
STOP
C
825 FORMAT(//,41X,'ENERGIES ARE IN UNITS OF (N1*M1)',
          'C**2)',//,41X,'MOMENTA ARE IN ',
          'UNITS OF C.')
850 FORMAT(H1,102X,'% DEVIATION FIELD ENERGY ',/,
          'TOTAL TOTAL KINETIC ENERGY',
          'ELECTRON FROM INITIAL AS % OF ',/,
          'TIME TOTAL ENERGY FIELD ENERGY KINETIC ENERGY',
          'MOMENTUM TOTAL ENERGY ELECTRONS',
          '-------------------------- ==============',
          '-------------------------- ==============',
          '-------------------------- ==============',
          '/

950 FORMAT(1X,F10.3,1X,F10.5,1X,F10.5,1X,F10.5,1X,F10.5,1X,F10.5,
          1X,F10.5)
975 FORMAT(A131)
C
END
C
END
SUBROUTINE CREATR ( VO, ILP, IUP, TEMP, OSTART )

The VX array is used as a dummy array for the total momentum, P, and the total velocity, V.

NSTEPS is the average number of integration steps per particle.

INSERT ':ESR1.INSERT(100,199)'
INSERT ':ESR1.INSERT(300,399)'

LOGICAL OSTART

Statement function definitions.

\[ F(XJ) = C \times DP^2 \times \exp \left( -\sqrt{1.0+DP^2 \cdot XJ^2}/T \right) \]
\[ G(XJ) = C \times DP^2 \times \exp \left( -0.5 \times DP^2 \cdot XJ^2/T \right) \]

IL = ILP
IU = IUP
IF ( IL .GE. IU .OR. TEMP .EQ. DZERO ) RETURN
T = TEMP
NF0UR = (IU-IL+1)/4
NF0UR = NF0UR*4
N = IU-IL+1
NSTPMX = 1000000
NSTEPS = MIN ( 200, NSTPMX/N )
IF ( NSTEPS .LT. 2 ) NSTEPS = 2

Velocities are in units of C and momenta in units of MC.

PMAX = 4.0DO \times \sqrt{T+4.0 \times T^2}
DP = PMAX/(N*NSTEPS)
DP2 = DP^2
PI = 0.5DO \times TWOPI
XJ = 1.0DO
I = ONE
AINT = DZERO
IF ( T .GT. 0.002 ) THEN
C = 0.25DO \times N \times \exp(1.0DO/T) / (T+T*T)
ELSE
C = 0.25DO \times N / (T+T*T)
ENDIF

IF ( T .GT. 0.002 ) THEN
AINT = AINT + XJ*F(XJ)
I4 = 4*I+IL-4
IF ( AINT .GT. I ) VX(I4) = XJ*DP
IF ( AINT .GT. I ) I = I+1
IF ( 4*I .GT. NF0UR .OR. XJ*DP .GT. PMAX ) GO TO 100
XJ = XJ+1.0DO
GO TO 50
ELSE
AINT = AINT + XJ*G(XJ)
I4 = 4*I+IL-4
IF ( AINT .GT. I ) VX(I4) = XJ*DP
IF ( AINT .GT. I ) I = I+1
IF ( 4*I .GT. NF0UR .OR. XJ*DP .GT. PMAX ) GO TO 100
XJ = XJ+1.0DO
GO TO 51
ELSE
ENDIF
GO TO 100
CONTINUE
IF ( 4*I .GT. N ) GO TO 170
I4 = 4*I + IL - 4

DO 150 K=I4,IU
VX(K) = DZERO
CONTINUE

DO 180 I=IL,IU

VX(I) = VX(I) / SQRT( 1.0D0 + VX(I)**2 )

C NFL = NFOUR-4

IF ( QSTART ) THEN

DO 200 J=1,NFL,4

I = J-1+IL

THETA = RADREV(J,3)*TWOPI

PX = VX(I)

VX(I) = PX*COS(THETA)

VY(I) = PX*SIN(THETA)

VX(I+1) = -VX(I)

VY(I+1) = -VY(I)

VY(I+2) = VX(I)

VX(I+2) = -VX(I)

VY(I+3) = VY(I)

VX(I+3) = VY(I)

CONTINUE

IF ( NFOUR .EQ. NILNF = IL+NFOUR

DO 201 J=ILNF,IU

VX(J) = DZERO

VY(J) = DZERO

CONTINUE

ELSE

DO 300 J=1,NFL,4

I = J-1+IL

THETA = RAN(ISD)*0.5D0*PI

PX = VX(I)

VX(I) = PX*COS(THETA)

VY(I) = PX*SIN(THETA)

VX(I+1) = -VX(I)

VY(I+1) = -VY(I)

VX(I+2) = VX(I)

VY(I+2) = -VX(I)

VY(I+3) = VY(I)

VX(I+3) = VY(I)

CONTINUE

IF ( NFOUR .EQ. N ILNF = IL+NFOUR

DO 301 J=ILNF,IU

VX(J) = DZERO

VY(J) = DZERO

CONTINUE

ENDIF

END
SUBROUTINE SETRHO ( IL, IU, Q )

C Accumulates charge density.

$INSERT 'ESR1.INSERT(100.399)'

IF ( IL .GT. IU ) RETURN

QDX = Q / DX
DXI = 1.000 / DX
XN = NG

C If it is the first group of particles then clear RHO.

IF ( IL .EQ. 1 ) THEN
    DO 1 J=1,NG
    1 RHO(J) = RHO0
    RHO(NG+1) = DZERO
ENDIF

C Linear.

DO 10 I=IL,IU
    X(I) = X(I)*DXI
    C
    IF ( X(I) .LT. DZERO ) X(I) = X(I) + XN
    IF ( X(I) .GE. XN ) X(I) = X(I) - XN
    J = X(I)
    DRHO = QDX * ( X(I)-J )
    RHO(J+1) = RHO(J+1) - DRHO + QDX
    RHO(J+2) = RHO(J+2) + DRHO
10 CONTINUE

RETURN

END

C=============================================================

C SUBROUTINE FIELDS ( ITH )

C Solves for PHI and E, computes field energy, etc.

$INSERT 'ESR1.INSERT(100.299)'
$ INSERT 'ESR1.INSERT(400,599)'
SINSERT 'ESR1.INSERT(700.799)'

DOUBLE PRECISION RHOK(1), PHIK(1), SCRACH(1), EK(257)
EQUIVALENCE ( RHO,RHOK ), ( PHI,PHIK ), ( E,SCRACH )
PARAMETER ( NG2M = NGMAX / 2 )
PI = 4.0 * ATAN(1.0)
HDX = 0.5DO * DX
HDXI = 0.5DO / DX
LI = 1.0DO / L
NG1 = NG+1

C First time step duties.

DATA NG2 / O /
IF ( NG2 .EQ. 0 ) THEN
    NG2 = NG / 2
ENDIF

C Set up ratio PHIK/RHOK.

C A2 > O gives a short-wavelength cutoff (smoothing).
C A1 > O gives a mid-range boost to compensate for
C slight attenuation in force calculation.

DO 2 K = 1,NG2
    KDX2 = (PI/NG)*K
4033 $\text{SM}(K) = \exp\left( A_1 \sin(KDX2)^2 - A_2 \tan(KDX2)^4 \right)$
4034 $\text{KSOI}(K) = \frac{\text{CGSHL}}{((2.0 \sin(KDX2)/DX)^2)*\text{SM}(K)^2}$
4035 1 CONTINUE
4036 ENIF
4037 C-----------------------------------------------:
4038 C Transform charge density.
4039 C-----------------------------------------------:
4040 RHO(1) = RHO(1) + RHO(NG1)
4041 RHO(NG1) = RHO(1)
4042 C-----------------------------------------------:
4043 C Save charge density values.
4044 C-----------------------------------------------:
4045 IF ( IRHO .NE. ZERO ) THEN
4046 IF ( (ITH/IRHO) * RHO .EQ. ITH ) THEN
4047 WRITEH2) ( RHO(J), J=1,NG1 )
4048 ENDIF
4049 ENDIF
4050 C
4051 DO 10 J = 1,NG
4052 RHOK(J) = RHO(J)*HDX
4053 SCRACH(J) = DZERO
4054 10 CONTINUE
4055 C
4056 CALL CPFT ( RHOK, SCRACH, NG, ONE, ONE )
4057 CALL RPFT2 ( RHOK, SCRACH, NG, ONE )
4058 RHOK(1) = DZERO
4059 C-----------------------------------------------:
4060 C Calculate PHIK and field energy.
4061 C-----------------------------------------------:
4062 ESES = DZERO
4063 PHIK(1) = DZERO
4064 C
4065 DO 20 K = 2,NG2
4066 KK = NG + 2 - K
4067 PHIK(K) = KSOI(K-1) * RHOK(K)
4068 PHIK(KK) = KSOI(K-1) * RHOK(KK)
4069 EK(K) = RHOK(K)*PHIK(K) + RHOK(KK)*PHIK(KK)
4070 ESES = ESES + EK(K)
4071 EK(1) = EK(K)*RNM1C2/L
4072 RHOK(K) = SM(K-1) * RHOK(K)
4073 RHOK(KK) = SM(K-1) * RHOK(KK)
4074 20 CONTINUE
4075 C
4076 PHIK(NG2+1) = KSOI(NG2) * RHOK(NG2+1)
4077 ESES(ITH+1) = (2.0D0*ESES + RHOK(NG2+1)*PHIK(NG2+1))
4078 /((2.0D0*L)*RNM1C2
4079 EK(NG2+1) = RHOK(NG2+1)*PHIK(NG2+1)*RNM1C2/(2.0D0*L)
4080 EK(1) = 0.0D0
4081 RHOK(NG2+1) = SM(NG2) * RHOK(NG2+1)
4082 C-----------------------------------------------:
4083 C Save specified mode energies.
4084 C-----------------------------------------------:
4085 DO 21 KM = 1,MMAX
4086 K = MPL0T(KM)+1
4087 IF ( K .EQ. 1 ) GO TO 22
4088 KK = NG + 2 - K
4089 ESEM(ITH+1,KM) = ( RHOK(K) * PHIK(K) +
4090 RHOK(KK) * PHIK(KK) )*RNM1C2 / L
4091 IF ( K .EQ. KK ) ESEM(ITH+1,KM) = 0.25*ESEM(ITH+1,KM)
4092 21 CONTINUE
4093 22 CONTINUE
4094 C-----------------------------------------------:
4095 C Inverse transform PHI.
4096 C-----------------------------------------------:
4097 DO 30 K = 1,NG
4098 RHO(K) = RHOK(K) * LI
4099 PHI(K) = PHIK(K) * LI
4100 30 CONTINUE
4101 C
4102 CALL RPFT12 ( PHI, RHO, NG, ONE )
CALL CPFT ( PHI, RHO, NG, ONE, -ONE )

PHI(NG1) = PHI(1)
RHO(NG1) = RHO(1)

C Save smoothed charge density and electric potential.

C-----
IF ( IRHOS .NE. ZERO ) THEN
  IF ( (ITH/IRHOS)*IRHOS .EQ. ITH ) THEN
    WRITE(12) ( RHO(J), J = 1,NG1 )
  ENDIF
ENDIF

C-----
IF ( IPHI .NE. ZERO ) THEN
  IF ( (ITH/IPHI)*IPHI .EQ. ITH ) THEN
    WRITE(12) ( PHI(J), J = 1,NG1 )
  ENDIF
ENDIF

C Uniform field.

EOT = EO*COS(WO*TIME)

C Centered difference across 2 cells.

DO 101 J = 2,NG
  E(J) = ( PHI(J-1)-PHI(J+1) )* HDXI + EOT
101 CONTINUE
E(1) = ( PHI(NG)-PHI(2) )* HDXI + EOT
E(NG1) = E(1)

C Save electric field values.

C-----
IF ( IE .NE. ZERO ) THEN
  IF ( (ITH/IE)*IE .EQ. ITH ) THEN
    WRITE(12) ( E(J), J = 1,NG1 )
  ENDIF
ENDIF

C Save electrostatic energy spectrum k-space modes.

IF ( ISPEC .NE. ZERO ) THEN
  IF ( (ITH/ISPEC)*ISPEC .EQ. ITH ) THEN
    NG2P = NG2+1
    WRITE(12) ( EK(J), J = 1,NG2P )
  ENDIF
ENDIF

C Electric field has not been normalized yet.

AEL = 1.0DO

C
RETURN
END
SUBROUTINE SETV ( IL, IU, Q, M, TT, PX, PY, VLIGHT )
C
C Converts particle velocities at t=0 to t= -DT/2.
C
DOUBLE PRECISION M
T = TT
C DTDX = DT/DX
C
C Rotate V thru angle +0.5*WC*DT (non-rel) and normalize VY. If T=0, there is no magnetic field. The rotation is omitted and no references are made to VY.
C
IF ( T .NE. DZERO ) THEN
  T = TAN (T)
  C = 1.0D0 / SQRT( 1.0D0 + T*T )
  S = C * T
  DO 1 I=IL,IU
       VX(I) = C*VXX - S*VY(I)
       VY(I) = S*VXX + C*VY(I)
  1 CONTINUE
ENDIF
C
C Only electrons are treated relativistically.
C
IF ( ABS(Q/M) .GE. 0.2D0 ) THEN
  DO 99 I = IL,IU
        V2 = VX(I)*VX(I) + VY(I)*VY(I)
        VX(I) = VX(I) * VLIGHT * DTDX
        VY(I) = VY(I) * VLIGHT * DTDX
        GAMMA(I) = 1.0D0 / SQRT( 1.0D0 - V2 )
  99 CONTINUE
ENDIF
C
C Electric impulse to go back 1/2 time step.
C
DATA DUM1, DUM2, DUM3 / 0.0D0, 0.0D0, 0.0D0 /
QQQ = -0.5D0 * Q
CALL ACCEL (IL,IU,QQQ,M,DUM1,PX,PY,DUM2,DUM3,VLIGHT)
C
RETURN
END
SUBROUTINE ACCEL (ILP, IUP, Q, M, TT, PX, PY, THERMT, KE, VLIGHT)

C Advances velocity one time step, computes momentum and kinetic energy.

C

$INSERT 'ESR1.INSERT(100,499)'

DOUBLE PRECISION M, MC2, NRKE, NONREL

DOUBLE PRECISION A(NG1M)

EQUIVALENCE (A,E)

C

VLITE2 = ( VLIGHT * VLIGHT )

C

IL = ILP
IU = IUP
NG1 = NG+1

IF ( IL .GT. IU ) RETURN

C

DO 100 J = 1, NG1

THRMX1(J) = DZERO
N(J) = 0
VXOSC(J) = DZERO
VYOSC(J) = DZERO

100 CONTINUE

C Renormalize acceleration if need be.

C

DXDT = DX/DT
AE = 0.5DO * (Q/M)*(DT/DXDT)

IF ( AE .NE. AEL ) THEN

TEM = AE / AEL

DO 1 J=1, NG1

A(J) = A(J) * TEM

1 CONTINUE

AEL = AE

ENDIF

C

IF ( IL .EQ. 1 ) KE = DZERO

IF ( ABS(Q/M) .LT. 0.1DO ) GO TO 5

C Linear, momentum conserving. Includes magnetic field and relativistic electrons.

C

RELKE = DZERO
NRKE = DZERO

C Half accel two-step rot'n method with quick rot'n scheme.

C

DO 250 I=IL, IU

J = X(I)

XX = X(I)-J

T = TT

AA = A(J+1) + XX*( A(J+2) - A(J+1) )

GVXX = VX(I)*GAMMA(I) + AA

GVYY = VY(I)*GAMMA(I)

GV2 = GVXX*GVXX + GVYY*GVYY

GAMMA2 = 1.0DO + GV2

GAMMA1 = SQRT (1.0DO + GV2)

VX(I) = GVX/GAMMA(I)

VY(I) = GVY/GAMMA(I)
6071  VY(I) = GVY/GAMMA(I)
6072  PY = PY + GVY
6073  PX = PX + GVX
6074  K = X(I) + 0.5DO
6075  N(K+1) = N(K+1) + 1
6076  VX0SC(K+1) = VX0SC(K+1) + VX(I)
6077  VY0SC(K+1) = VY0SC(K+1) + VY(I)
6078  250 CONTINUE
6079  C
6080  N(NG1) = N(1) + N(NG1)
6081  N(1) = N(NG1)
6082  VX0SC(NG1) = VX0SC(1) + VX0SC(NG1)
6083  VX0SC(1) = VX0SC(NG1)
6084  VY0SC(NG1) = VY0SC(1) + VY0SC(NG1)
6085  VY0SC(1) = VY0SC(NG1)
6086  KE = KE + MAX (RELKE.NRKE)
6087  C
6088  C Drift momentum per unit mass in units of C.
6089  C
6090  PY = PY/(IU-IL+1)
6091  PX = PX/(IU-IL+1)
6092  IF ( ITHERM .LE. O ) GO TO 3
6093  IF ( (IT/ITHERM)*ITHERM .NE. IT ) GO TO 3
6094  VPLAS = DZERO
6095  C
6096  DO 300 J=1,NG1
6097  IF ( N(J) .EQ. 0 ) GO TO 300
6098  VXOSC(J) = VXOSC(J)/ N(J)
6099  VYOSC(J) = VYOSC(J)/ N(J)
6100  C VPLAS = VPLAS + VXOSC(J)/ NG1
6101  300 CONTINUE
6102  C
6103  DO 350  I=IL.IU
6104  K = X(I) + 0.5DO
6105  VX0SC(K+1) = VX0SC(K+1) - VPLAS
6106  GG = 1.0D0 / SORT ( 1.0D0 - (VX0SC(K+1)*VX0SC(K+1))
6107  - (VY0SC(K+1)*VY0SC(K+1)) )
6108  C
6109  C Relativistic calculation. Non-relativistically correct.
6110  C
6111  PYTHRM = VY(I) * GAMMA(I)
6112  PXTHRM = VX(I) * GAMMA(I)
6113  PXSQ = GG * VX0SC(K+1) * GG * VX0SC(K+1)
6114  PYSQ = GG * VY0SC(K+1) * GG * VY0SC(K+1)
6115  C
6116  C Rel. KE defined as KE(avg. motion + jiggle)-KE(avg. motion):
6117  C
6118  RELT = SORT ( 1.0D0 + PXTHRM*PXTHRM
6119  + PYTHRM*PYTHRM )
6120  . - SORT (1.0D0 + PXSQ*PYSQ)
6121  . - SORT (1.0D0 + PXSQ*PYSQ)
6122  RELT = .5110041D06 * RELT
6123  C
6124  C Non-relativistic calculation.
6125  C
6126  VXTHRM = VX(I) - VX0SC(K+1)
6127  VYTHRM = VY(I) - VY0SC(K+1)
6128  NONREL = .5100410D0*(VXTHRM*VXTHRM + VYTHRM*VYTHRM)
6129  TEMPP = MAX ( RELT, NONREL )
6130  THERMT = THERMT + TEMPP/(IU-IL+1)
6131  THRMX1(K+1) = THRMX1(K+1) + TEMPP/N(K+1)
6132  350 CONTINUE
6133  C
6134  C Save electron temperature profile.
6135  C
6136  WRITE(12) ( THRMX1(J), J=1,NG1 )
6137  GOTO 3
6138  GOTO 3
6139  C
6140  5 CONTINUE
IONS:
Linear, momentum conserving. Includes magnetic field. Non-relativistic.

C Half accel two-step rotation scheme with quick (Buneman) rotation.

DO 251 I=IL, IU
    J = X(I)
    XX = X(I) - J
    T = TAN(TT)
    S = (T+T) / (1.0DDO + T*T)
    AA = A(J+1) + XX*(A(J+2) - A(J+1))
    VX = VX(I) + AA
    VY = VY(I)
    V25 = V25 + VX*VXX + VYY*VYY
    VXX = VXX + T*VYY
    VYY = VYY - S*VXX
    VX(I) = VX(I) + T*VYY + AA
    VY(I) = VY(I)
    K = X(I) + 0.5DDO
    N(K+1) = N(K+1) + 1
    VXOS = VXOS(J) + VXOSC(J) + VX(I)
    VXOS(J) = VXOSC(J) + VXOS(J+1) + VX(I)
    V2SX(J) = V2SX(J) + V2SX(J+1) + VX(I)*VXX(I) + VYY(I)*VYY(I)
    V1XS = V1XS + VX(I)
    V1YS = V1YS + VY(I)

251 CONTINUE

DO 400 J=1, NG
    IF ( N(J).EQ.0 ) GO TO 400
    V250SC = ( VXOS(J) + VXOS(J) + VXOSC(J) + VXOSC(J) + VXOS(J) )/N(J)
    THRXX(J) = 0.5DDO*M2C2EV*V250SC
    THRXX(J) = THRXX(J)/N(J)

400 CONTINUE

WRITE(12) ( THRXX(J), J=1, NG1 )

C ---

Drift momentum per unit mass in units of C.

PX = V1XS / (IU-IL+1)
PY = V1YS / (IU-IL+1)

3 CONTINUE

RETURN
SUBROUTINE MOVE (ILP, IUP, Q)
C Advances position one time step and accumulates
C charge density.
C
$INSERT 'ESR1.INSERT(100,399)'
$INSERT 'ESR1.INSERT(500,899)'
C
IL = ILP
IU = IUP
IF (IL .GT. IU) RETURN
ODX = Q/DX
XN = NG
NG1 = NG+1
C
C If it is the first group of particles then clear RHO.
C
IF (IL .EQ. 1) THEN
   DO 1 J = 1,NG
      RHO(J) = RHOO
   CONTINUE
   RHO(NG+1) = DZERO
ENDIF
DO 2 I=IL,IU
   Linear weighting using old positions.
   X(I) = X(I) + VX(I)
   IF (X(I) .LT. DZERO) X(I) = X(I) + XN
   IF (X(I) .GE. XN) X(I) = X(I) - XN
C
   Linear weighting using new positions.
   J = X(I)
   DRHO = ODX * (X(I)-J)
   RHO(J+1) = RHO(J+1) - DRHO + ODX
   RHO(J+2) = RHO(J+2) + DRHO
2 CONTINUE
RETURN
END

SUBROUTINE RITER1 (NTH,MMAX,RSTART,IRDATE,IRTIME,MODE,
ATITLE,NT,NP,VLIGHT,KO)
C
INSERT 'ESR1.INSERT(100,299)'
INSERT 'ESR1.INSERT(400,499)'
INSERT 'ESR1.INSERT(600,799)'
C
DOUBLE PRECISION KO
LOGICAL RSTART
CHARACTER*131 ATITLE
WRITE(7,110)
WRITE(7,120)
WRITE(7,440) ATITLE
WRITE(7,100)
IF (RSTART) THEN
   WRITE(7,130) IRTIME,
ENDIF
WRITE(7,160) ITIME, RM1C2I
WRITE(7,150) IDATE, RNM1C2
WRITE(7,200) L.RM2C2I
WRITE(7,210) NT, RNM2C2
WRITE(7,220) DT, M2C2EV

C
RETURN
END

C
SUBROUTINE RITER1 (NTH,MMAX,RSTART,IRDATE,IRTIME,MODE,
ATITLE,NT,NP,VLIGHT,KO)
C

100 FORMAT(1X, ' = ', 129X, ' = ', /, 1X, ' == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == =
A1 = ', 1PD12.5, 'Write interval for charge density . . .  
IRHO = ', 14 
260 FORMAT(1X, 'Mid-range boost  
A2 = ', 1PD12.5, 'Write interval for smoothed density  
IRHOS = ', 14 
270 FORMAT(1X, 'EO-add uniform field EO*COS(WO*TIME)  
EO = ', 1PD12.5, 'Write interval for potential  
IPHI = ', 14 
280 FORMAT(1X, 'WO-add uniform field EO*COS(WO*TIME)  
WO = ', 1PD12.5, 'Write interval for electric field  
IE = ', 14 
440 FORMAT(1X  
450 FORMAT(1X, 'Light velocity in terms of DX/DT  
VLIGHT = ', 1PD12.5, 'No. timesteps between history writes  
NTH = ', 15 
470 FORMAT(1X, 'System fundamental wavenumber  
KO = ', 1PD12.5, 'Write interval for energy spectrum  
ISPEC = ', 15 
480 FORMAT(1X,  
17X, 'Uniform background charge density  
RHOO = ', 1PD12.5)
SUBROUTINE CPFT ( R, I, N, INCP, SIGNP )

IMPLICIT DOUBLE PRECISION ( A-H, O-Z )

C Table of sines.

DATA SINES(1) /0.000/

IF ( SINES(1) .EQ. 1.000 ) GO TO 1

SINES(1) = 1.000

T = DATAN(1.000)

DO 2 IS = 2,15

SINES(IS) = SIN(T)

T = T / 2.000

1 CONTINUE

C Set up various indices.

IF ( N .EQ. 1 ) RETURN

C==================================================

SUBROUTINE CPFT ( R, I, N, INCP, SIGNP )

IMPLICIT DOUBLE PRECISION ( A-H, O-Z )

C Table of sines.

DATA SINES(1) /0.000/

IF ( SINES(1) .EQ. 1.000 ) GO TO 1

SINES(1) = 1.000

T = DATAN(1.000)

DO 2 IS = 2,15

SINES(IS) = SIN(T)

T = T / 2.000

1 CONTINUE

C Set up various indices.

IF ( N .EQ. 1 ) RETURN

C==================================================
10027 INC = INCP
10028 SGN = SIGNP
10029 NINC = N*INC
10030 SPAN = NINC
10031 IT = N/2
10032 C
10033 DO 3 IS=1,15
10034 IF ( IT .EQ. 1 ) GO TO 12
10035 3 IT = IT/2
10036 C
10037 10 T = S + ( SO*C - CO*S )
10038 C = C - ( CO*C + SO*S )
10039 S = T
10040 C Replication loop.
10041 C
10042 11 K1 = KO + SPAN
10043 RO = R(KO+1)
10045 R1 = R(K1+1)
10046 IO = I(KO+1)
10047 I1 = I(K1+1)
10048 R(KO+1) = RO + R1
10049 I(KO+1) = IO + I1
10050 RO = RO - R1
10051 IO = IO - I1
10052 R(K1+1) = C*RO - S*IO
10053 I(K1+1) = S*RO + C*IO
10054 KO = K1 + SPAN
10055 C
10056 IF ( KO .LT. NINC ) GO TO 11
10057 C
10058 K1 = KO - NINC
10059 C = -C
10060 KO = SPAN - K1
10061 C
10062 IF ( K1 .LT. KO ) GO TO 11
10063 C
10064 KO = KO + INC
10065 C
10066 IF ( KO .LT. K1 ) GO TO 10
10067 C Recursion to next level.
10068 C
10069 12 CONTINUE
10070 SPAN = SPAN/2
10071 KO = 0
10072 C
10073 C Angle = 0 loop.
10074 C
10075 13 K1 = KO + SPAN
10076 RO = R(KO+1)
10078 R1 = R(K1+1)
10079 IO = I(KO+1)
10080 I1 = I(K1+1)
10081 R(KO+1) = RO + R1
10082 I(KO+1) = IO + I1
10083 R(K1+1) = RO - R1
10084 I(K1+1) = IO - I1
10085 KO = K1 + SPAN
10086 C
10087 IF ( KO .LT. NINC ) GO TO 13
10088 C Are we finished.....?
10089 C
10090 10091 IF ( SPAN .EQ. INC ) GO TO 20
10092 C If no - prepare non-zero angles.
10093 C
10094 10095 CO = 2.000* SINES(IS)**2
10096 IS = IS-1


```
S = SIGN(SIN(IS), SGN)
SO = S
C = 1.000 - CO
KO = INC
GO TO 11

20 N1 = NINC - INC
N2 = NINC/2
RC = O
IJ = O
JI = O

C
GO TO 11

C Even:

21 IJ = N1 - IJ
Jl = N1 - Jl
T = R(IJ+1)
R(IJ+1) = R(JI+1)
R(JI+1) = T
T = I(IJ+1)
I(IJ+1) = I(JI+1)
I(JI+1) = T

C
IF (I J .GT. N2) GO TO 21

C Odd:

22 IJ = IJ + INC
Jl = JI + N2
T = R(IJ+1)
R(IJ+1) = R(JI+1)
R(JI+1) = T
T = I(IJ+1)
I(IJ+1) = I(JI+1)
I(JI+1) = T

C
IF (U • LT. JI) GO TO 21
IF (IJ .LT. N2) GO TO 22

C Increment reversed counter.

23 IT = IT/2
RC = RC - IT

C
IF (RC .GE. O) GO TO 23
RC = RC + 2*IT
IJ = RC

C
IF (I J .LT. JI) GO TO 21
IF (IJ .LT. N2) GO TO 22

C
RETURN
END
```
SUBROUTINE RPFT2 ( A, B, N, INC)

IMPLICIT DOUBLE PRECISION ( A-H, O-Z )

DOUBLE PRECISION A(1), B(1)

INC  =  INC
NINC  =  N*INC
A(1)  =  A(1) + A(1)
B(1)  =  B(1) + B(1)
LP  =  INC
LM  =  NINC - LP

IF ( LP .GE. LM ) GO TO 2

1
RP  =  A(LP+1)
RM  =  A(LM+1)
IP  =  B(LP+1)
IM  =  B(LM+1)
A(LP+1)  =  RM + RP
B(LP+1)  =  RM - RP
B(LP+1)  =  IP + IM
A(LM+1)  =  IP - IM

LP  =  LP + INC
LM  =  NINC - LP

IF ( LP .LT. LM ) GO TO 1

RETURN

END
SUBROUTINE SAVEIT (ITH,M1,M2,Q1,Q2,T1,T2,N11,NP,VLIGHT)

REAL KO
REWIND 3
MTH = IT-ITHL+1
NG1 = NG + 1
MTH1 = MTH + 1
KO = TWOPI/L
WRITE(3) A1, A2, AEL, DT, DX, CGSHL, EO, L, VLIGHT,
        TIME, RHOO, WO, M1, M2, Q1, Q2, KO,
        T1, T2, NM1, NM2, IT, ITH, ITHL,
        NG, IDATE, ITIME, N11, NP
WRITE(3) ( RHO(I), I = 1,NG1 )
WRITE(3) ( PHI(I), I = 1,NG1 )
WRITE(3) ( E(I), I = 1,NG1 )
NREC2 = MTH/1024
IF ( MOD(MTH,1024) .GT. 0 ) NREC2 = NREC2+1
DO 20 KM = 1,MMAX
JSTAR = 1
JEND = MIN(1024,MTH)
DO 19 JJ = 1,NREC2
WRITE(3) ( ESEM(J,KM), J = JSTAR,JEND )
JSTAR = JEND + 1
JEND = JEND + 1024
JEND = MIN(JEND,MTH)
19 CONTINUE
20 CONTINUE
NREC2 = MTH1/1024
IF ( MOD(MTH1,1024) .GT. 0 ) NREC2 = NREC2+1
JSTAR = 1
JEND = MIN(1024,MTH)
JEND1 = JEND + 1
IF ( JEND1 .GT. 1024 ) JEND1=1024
DO 29 JJ = 1,NREC2
WRITE(3) ( KE1(J), J = JSTAR,JEND )
WRITE(3) ( KE2(J), J = JSTAR,JEND )
WRITE(3) ( TE(J), J = JSTAR,JEND )
WRITE(3) ( ESE(J), J = JSTAR,JEND )
WRITE(3) ( XT(J), J = JSTAR,JEND1 )
WRITE(3) ( VXT(J), J = JSTAR,JEND1 )
WRITE(3) ( P1X(J), J = JSTAR,JEND1 )
WRITE(3) ( P2X(J), J = JSTAR,JEND1 )
WRITE(3) ( P1Y(J), J = JSTAR,JEND1 )
WRITE(3) ( P2Y(J), J = JSTAR,JEND1 )
JSTAR = JEND + 1
JEND = JEND + 1024
JEND = MIN(JEND,MTH)
IF ( (JEND-JSTAR) .LT. 1023 ) THEN
JEND1 = JEND + 1
ELSE
JEND1 = JEND
ENDIF
29 CONTINUE
NRECS = NP/1024
IF ( MOD(NP,1024) .GT. 0 ) NRECS = NRECS+1
JS = 1
JE = MIN( NP, 1024 )
DO 30 I = 1, NRECS
  WRITE(3) ( X(J), J=JS, JE )
  JS = JE + 1
  JE = JE + 1024
  JE = MIN(JE, NP)
30 CONTINUE

C

JS = 1
JE = MIN( NP, 1024 )
DO 40 I = 1, NRECS
  WRITE(3) ( VX(J), J=JS, JE )
  JS = JE + 1
  JE = JE + 1024
  JE = MIN(JE, NP)
40 CONTINUE

C

JS = 1
JE = MIN( NP, 1024 )
DO 50 I = 1, NRECS
  WRITE(3) ( GAMMA(J), J=JS, JE )
  JS = JE + 1
  JE = JE + 1024
  JE = MIN(JE, NP)
50 CONTINUE

C

WRITE(3) RNM1C2, RNM2C2, RM1C2I, RM2C2I, M2C2EV
RETURN
END

SUBROUTINE RESTOR ( IRDATE, IRTIME, ITH, M1, M2, KO, Q1, Q2, T1, T2, N11, NP, VLIGHT )

C

REAL KO
REWIND 3
C
READ (3) A1, A2, AEL, DT, DX, CGSHL, EO, L, VLIGHT,
          TIME, RH00, W0, M1, M2, Q1, Q2, KO,
          T1, T2, NM1, NM2, IT, ITH, ITHL,
          NG, IRDATE, IRTIME, N11, NP
C
MTH = IT-ITHL+1
NG1 = NG + 1
GN = FLOAT(NG)
MTH1 = MTH + 1
C
READ(3) ( RHO(I), I = 1, NG1 )
READ(3) ( PHI(I), I = 1, NG1 )
READ(3) ( E(I), I = 1, NG1 )
C
NREC2 = MTH/1024
IF ( MOD(MTH,1024) .GT. 0 ) NREC2 = NREC2+1
C
DO 20 KM = 1, MMAX
  USTAR = 1
JEND = MIN(1024,MTH)
DO 19 JJ = 1,NREC2
READ(3) ( ESEM(J,KM), J=JSTAR,JEND )
JSTAR = JEND + 1
JEND = JEND + 1024
JEND = MIN(JEND,MTH)
19 CONTINUE

NREC2 = MTH1/1024
IF ( MOD(MTH1,1024) .GT. 0 ) NREC2 = NREC2+1
JEND = MIN(1024,MTH)
JEND1 = JEND + 1
IF ( JEND1 .GT. 1024 ) JEND1=1024

DO 29 JJ = 1,NREC2
READ(3) ( KE1(J), J=JSTAR,JEND )
READ(3) ( KE2(J), J=JSTAR,JEND )
READ(3) ( TE(J), J=JSTAR,JEND )
READ(3) ( ESE(J), J=JSTAR,JEND )
READ(3) ( XT(J), J=JSTAR,JEND1 )
READ(3) ( VX(T), J=JSTAR,JEND1 )
READ(3) ( P1X(J), J=JSTAR,JEND1 )
READ(3) ( P2X(J), J=JSTAR,JEND1 )
READ(3) ( PY(Y(J), J=JSTAR,JEND1 )
READ(3) ( P2Y(J), J=JSTAR,JEND1 )
JSTAR = JEND + 1
JEND = JEND + 1024
JEND = MIN(JEND,MTH)
IF ( (JEND-JSTAR) .LT. 1023 ) THEN
JEND1 = JEND + 1
ELSE
JEND1 = JEND
ENDIF
29 CONTINUE

NRECS = NP/1024
IF ( MOD(NP,1024) .GT. 0 ) NRECS = NRECS+1
JS = 1
JE = MIN( NP,1024 )
DO 30 I = 1,NRECS
READ(3) ( X(J), J=JS,JE )
JS = JE + 1
JE = JE + 1024
JE = MIN(JE,NP)
30 CONTINUE

JS = 1
JE = MIN( NP,1024 )
DO 40 I = 1,NRECS
READ(3) ( VX(J), J=JS,JE )
JS = JE + 1
JE = JE + 1024
JE = MIN(JE,NP)
40 CONTINUE

JS = 1
JE = MIN( NP,1024 )
DO 50 I = 1,NRECS
READ(3) ( VY(J), J=JS,JE )
JS = JE + 1
JE = JE + 1024
JE = MIN(JE,NP)
50 CONTINUE

JS = 1
JE = MIN( NP,1024 )
DD 60 I = 1,NRECS
READ(3) ( GAMMA(J), J=JS,JE )
JS = JE + 1
JE = JE + 1024
JE = MIN(JE,NP)
60 CONTINUE

C
READ (3) RNM1C2, RNM2C2, RM1C2I, RM2C2I, M2C2EV
C
RETURN
END

C=====================================================================
SUBROUTINE SMEAR ( X, IL, IU, ISEED )
C
This subroutine randomizes a vector array by random pair exchange.
C=====================================================================
IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
DIMENSION X(1)
NUM = IU-IL+1
ISD = ISEED
DO 70 I=IL,IU
II = NUM*RAN(ISD) + IL
XX = X(I)
X(I) = X(II)
X(II) = XX
70 CONTINUE
RETURN
END

FUNCTION RADREV ( I, IB )
Generates a number between 0 and 1 for PIC quiet start by mixed-radix digit reversal.
IB=arithmetic base
IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
II = I*IB
RADREV = DZERO
DO 10 N = 17,1,-1
IBN = IB**N
IX = II/IBN
IF ( IX .NE. 0 ) THEN
   RADREV = RADREV + FLOAT(IX) / FLOAT(IBN)
ENDIF
10 CONTINUE
RETURN
END
C======================================================================
C      SUBROUTINE ZEROS
C======================================================================
C At T=0, zero arrays.
C
17001  C
17002  SUBROUTINE ZEROS
17003  C
17004  C======================================================================
17005  C At T=0 , zero arrays.
17006  C
17007  $INSERT 'ESR1.INSERT(100,199)'
17008  $INSERT 'ESR1.INSERT(300,399)'
17009  $INSERT 'ESR1.INSERT(500,599)'
17010  C
17011  C       MTH = NTH+1
17012  C
17013  DO 10 I=1,MTH
17014     KE1(I) = DZERO
17015     KE2(I) = DZERO
17016     P1X(I+1) = DZERO
17017     P1Y(I+1) = DZERO
17018     P2X(I+1) = DZERO
17019     P2Y(I+1) = DZERO
17020     TE(I) = DZERO
17021     ESE(I) = DZERO
17022     XT(I) = DZERO
17023     VXT(I+1) = DZERO
17024  10 CONTINUE
17025  C
17026  P1X(1) = DZERO
17027  P1Y(1) = DZERO
17028  P2X(1) = DZERO
17029  P2Y(1) = DZERO
17030  VXT(1) = DZERO
17031  C
17032  DD 20 KM=1,MMAX
17033  DD 19 I=1,MTH
17034  ESEM(I,KM) = DZERO
17035  19 CONTINUE
17036  20 CONTINUE
17037  C
17038  RETURN
17039  END
C======================================================================
C      SUBROUTINE DATEME ( DATERS, IDATE, ITIME )
C======================================================================
C Converts character date & time as output by the array

C processor routine SYS$GETDATE to Integer data.
C
18000  C
18001  C      SUBROUTINE DATEME ( DATERS, IDATE, ITIME )
18002  C
18003  C
18004  C======================================================================
18005  C Converts character date & time as output by the array
18006  C processor routine SYS$GETDATE to Integer data.
18007  C======================================================================
18008  C
18009  C      SUBROUTINE DATEME ( DATERS, IDATE, ITIME )
18010  C
18011  C======================================================================
18012  C
18013  C
18014  C
18015  C
18016  C
18017  C
18018  C
18019  C
18020  C
18021  C
18022  C
18023  C
18024  C IF ( DATE(4) .EQ. 'S' ) THEN
18025  IIIDATE(11) = 0
18026  IIIDATE(12) = 9
18027  ELSE
18028  C IF ( DATE(4) .EQ. 'O' ) THEN
18029  IIIDATE(11) = 1
18030  IIIDATE(12) = 0
ELSE
IF ( DATE(4) .EQ. 'N' ) THEN
IIDATE(11) = 1
IIDATE(12) = 1
ELSE
IF ( DATE(4) .EQ. 'F' ) THEN
IIDATE(11) = 0
IIDATE(12) = 2
ELSE
IIDATE(11) = 1
IIDATE(12) = 2
ELSE
IF ( DATE(4) .EQ. 'A' ) THEN
IIDATE(11) = 0
IF ( DATE(5) .EQ. 'P' ) THEN
IIDATE(12) = 4
ELSE
IIDATE(12) = 8
ENDIF
ELSE
IF ( DATE(4) .EQ. 'M' ) THEN
IIDATE(11) = 0
IF ( DATE(6) .EQ. 'V' ) THEN
IIDATE(12) = 5
ELSE
IIDATE(12) = 3
ENDIF
ELSE
IF ( DATE(4) .EQ. 'J' ) THEN
IIDATE(11) = 0
IF ( DATE(5) .EQ. 'A' ) THEN
IIDATE(12) = 1
ELSE
IIDATE(12) = 6
ELSE
IIDATE(12) = 7
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ELSE
IF ( CDATE(I) .EQ. 'O' .OR. CDATE(I) .EQ. ' ' ) THEN
IIDATE(I) = 0
ELSE
IF ( CDATE(I) .EQ. '1' ) THEN
IIDATE(I) = 1
ELSE
IF ( CDATE(I) .EQ. '2' ) THEN
IIDATE(I) = 2
ELSE
IF ( CDATE(I) .EQ. '3' ) THEN
IIDATE(I) = 3
ELSE
IF ( CDATE(I) .EQ. '4' ) THEN
IIDATE(I) = 4
ELSE
IF ( CDATE(I) .EQ. '5' ) THEN
IIDATE(I) = 5
ELSE
IF ( CDATE(I) .EQ. '6' ) THEN
IIDATE(I) = 6
ELSE
C
DO 10 I=1,10
IF ( CDATE(I) .EQ. 'O' .OR. CDATE(I) .EQ. ' ' ) THEN
IIDATE(I) = 0
ELSE
IF ( CDATE(I) .EQ. '1' ) THEN
IIDATE(I) = 1
ELSE
IF ( CDATE(I) .EQ. '2' ) THEN
IIDATE(I) = 2
ELSE
IF ( CDATE(I) .EQ. '3' ) THEN
IIDATE(I) = 3
ELSE
IF ( CDATE(I) .EQ. '4' ) THEN
IIDATE(I) = 4
ELSE
IF ( CDATE(I) .EQ. '5' ) THEN
IIDATE(I) = 5
ELSE
IF ( CDATE(I) .EQ. '6' ) THEN
IIDATE(I) = 6
ELSE
C
DO 10 I=1,10
IF ( CDATE(I) .EQ. '7' ) THEN
   IIDATE(I) = 7
ELSE
   IF ( CDATE(I) .EQ. '8' ) THEN
      IIDATE(I) = 8
   ELSE
      IF ( CDATE(I) .EQ. '9' ) THEN
         IIDATE(I) = 9
      ENDIF
   ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
10 CONTINUE
IDATE = IIDATE(3)*100000 + IIDATE(4)*10000 + IIDATE(11)*1000 +  
       IIDATE(12)*100 + IIDATE(1)*10 + IIDATE(2)
ITIME = IIDATE(5)*100000 + IIDATE(6)*10000 + IIDATE(7)*1000 +  
        IIDATE(8)*100 + IIDATE(9)*10 + IIDATE(10)
C RETURN
C
END
C=====================================================================
C SUBROUTINE PEAKS ( MTH )
C=====================================================================
C Computes (i) time of field energy peaks (TIME).
C (ii) time between peaks ( DELTA T ).
C (iii) real part of plasma frequency (OMEGA).
C (iv) the ratio of the peak amplitudes (RATIO).
C (v) the imaginary part of the frequency (GAMMA).
C
DOUBLE PRECISION TIM(NTH1)
PARAMETER ( PI = 3.14159265358979D0 )
INTEGER IPEAKS(750)
TL = ITHL*DT
C
DO 1 I = 1,MTH
   TIM(I) = (I-1)*DT + TL
1 CONTINUE
C Locate peaks.
C
MTHM2 = MTH-2
IPEAKS(1) = 1
J = 2
DO 10 I = 3,MTHM2
   IF ( ESE(I).GT.ESE(I-2) .AND. ESE(I).GT.ESE(I-1) .AND.  
      ESE(I).GT.ESE(I+1) .AND. ESE(I).GT.ESE(I+2) ) THEN
      IPEAKS(J)=I
      J=J+1
   ENDIF
10 CONTINUE
NPEAKS=J-1
C Calculate and write out peak data.
C
IF ( NPEAKS .GT. 2 ) THEN
   WRITE(7,3)
   WRITE(7,4)
   DO 20 J=2,NPEAKS
      I = IPEAKS(J-1)
      II = IPEAKS(J)
      DELT = TIM(II)-TIM(I)
   20   CONTINUE
RATIO = SORT(ESE(II)/ESE(I))
OMEGA = PI/DELT
GAMMA = LOG(RATIO)/DELT
WRITE(7,5) TIM(II), DELT, OMEGA, RATIO, GAMMA
CONTINUE
ENDIF
C Locate peaks for mode electrostatic energies.
DO 50 KM = 1,MMAX
MODE = MPLLOT(KM)
IF ( MODE .NE. ZERO ) THEN
IPEAKS(1) = 1
J = 2
DO 30 I = 3,MTHM2
IF ( ESEM(I,KM) .GT. ESEM(I-2,KM) .AND. ESEM(I,KM) .GT. ESEM(I-1,KM) .AND. ESEM(I,KM) .GT. ESEM(I+1,KM) .AND. ESEM(I,KM) .GT. ESEM(I+2,KM) ) THEN
IPEAKS(J) = I
J = J+1
ENDIF
30 CONTINUE
NPEAKS = J-1
C Calculate and write out peak data.
IF ( NPEAKS .GT. 2 ) THEN
WRITE(7,6) MODE
WRITE(7,4)
DO 40 J = 2,NPEAKS
I = IPEAKS(J-1)
II = IPEAKS(J)
DELTA = TIM(II) - TIM(I)
RATIO = SORT(ESEM(I,KM)/ESEM(I,KM))
OMEGA = PI/DELT
GAMMA = LOG(RATIO)/DELT
WRITE(7,5) TIM(II), DELT, OMEGA, RATIO, GAMMA
40 CONTINUE
ENDIF
50 CONTINUE
RETURN
3 FORMAT(/,22X,'Field Energy calculated data.',/.,21X,'-------------/',9X,'TIME',7X,'DELTA T',6X,'OMEGA',7X,'RATIO',7X,'GAMMA')
4 FORMAT(7X,'============',3X,'============',3X,'============')
5 FORMAT(7X,3X,F9.5,3X,F9.5,3X,F9.5,3X,F9.5,3X,F9.5)
6 FORMAT(/,14X,'Mode ',12,' Electrostatic Energy', ' calculated data./.,13X,'--------------------/',9X,'TIME',7X,'DELTA T',6X,'OMEGA',7X,'RATIO',7X,'GAMMA')
END
RATIO = SORT(ESE(I))
OMEGA = PI/DELT
GAMMA = LOG(RATIO)/DELT
WRITE(7,5) TIM(I), DELT, OMEGA, RATIO, GAMMA
20 CONTINUE
ENDIF
C-----------------------------------------------
C Locate peaks for mode electrostatic energies.
C-----------------------------------------------
DO 50 KM = 1, MMAX
    MODE = MNPLOT(KM)
    IF ( MODE .NE. ZERO ) THEN
        IPEAKS(1) = 1
        J = 2
        DO 30 I = 3, MTHM2
            IF ( ESEM(I,KM) .GT. ESEM(I-2,KM) ) THEN
                IPEAKS(J) = I
                J = J+1
            ENDIF
        CONTINUE
        NPEAKS = J-1
    ENDIF
50 CONTINUE
END
C----------------------------------------------------------C
1000  C
1001  C
1002  C  ES1PLT - A plotting program for ESR1.  C
1003  C
1004  C----------------------------------------------------------C
1005  PROGRAM ES1PLT
1006  INCLUDE ( MPARAM )
1007  INCLUDE ( MFIELD )
1008  INCLUDE ( MPTCL )
1009  INCLUDE ( MCNTRL )
1010  INCLUDE ( MTIME )
1011  INCLUDE ( MTHERM )
1012  INCLUDE ( AJUNK )
1013  INCLUDE ( HPLOT )
1014  INTEGER INBUF1(2052), INPAR1(1024), GETUR, STATUS,
1015  INBUF2(2052), INPAR2(1024), FPI2HI, FPR2HD
1016  DOUBLE PRECISION APDAT1(1024), HODAT1(1024),
1017  APDAT2(1024), HODAT2(1024)
1018  NAMELIST /IN1/ NPTCL1, NPTCL2, NPTCLT,
1019  IFVX01, IFVY01, IFVX02, IFVY02, IFVXOT,
1020  IFVX1, IFVY1, IFVX2, IFVY2, IFVXT,
1021  ITMP01, ITMP1, ITMP2, ITMP3,
1022  IVXV1, IVXV2, IVXV3, IVXV4,
1023  IVXV2, IVXV3, IVXV4, IVXV1,
1024  IRHO0, IRHOS0, IPHIO, IEO, ISPEC0,
1025  IRH01, IRH0S1, IPHI1, IE1, ISPEC1,
1026  ITE, IIE1, IIE2, IIE3, IIESE,
1027  IP1X, IP1Y, IP2X, IP2Y, IXT,
1028  IVXT, IELKE,
1029  JFLAG, XXORIG, XXMAX, YYORIG, YYMAX,
1030  LHSTRY, LFIELD, LSPACE
1031  C----------------------------------------------------------C
1032  C Default (set below) is to make certain history plots only.  C
1033  C----------------------------------------------------------C
1034  DATA NPTCL1, NPTCL2, NPTCLT / 3 * 0 /
1035  DATA IFVX01, IFVX02, IFVXOT / .FALSE., .FALSE., .FALSE. /
1036  DATA IFVY01, IFVY02 / .FALSE., .FALSE., .FALSE. /
1037  DATA ITMP01, ITMP02 / .FALSE., .FALSE. /
1038  DATA IPHIO, IEO, ISPEC0 / .FALSE., .FALSE., .FALSE. /
1039  DATA IFVX1, IFVY1, IFVX2, IFVY2, IFVXT / 5 * 0 /
1040  DATA ITMP1, ITMP2 / 2 * 0 /
1041  DATA IVXV1, IVXV2, IVXV3, IVXV4 / 5 * 0 /
1042  DATA IVXV2, IVXV3, IVXV4 / 4 * 0 /
1043  DATA IRH01, IRH0S1, IPHI1, IE1, ISPEC1 / 5 * 0 /
1044  UFLAG = 1
1045  IRS = 0
1046  ITE = .TRUE.
1047  IKE1 = .TRUE.
1048  IKE2 = .FALSE.
1049  IIE3 = .TRUE.
1050  IIESE = .TRUE.
1051  IP1X = .TRUE.
1052  IP1Y = .FALSE.
1053  IP2X = .FALSE.
1054  IP2Y = .FALSE.
1055  IXT = .FALSE.
1056  IVXT = .FALSE.
1057  IELKE = .FALSE.
1058  LHSTRY = .TRUE.
1059  LSPACE = .FALSE.
1060  LFIELD = .FALSE.
1061  LSPACE = .FALSE.
1062  STATUS = INITIB ( INBUF1, 'APOUT', 6 )
1063  STATUS = INITIB ( INBUF2, 'HIST.AP', 7 )
1064  CALL DSPDEV ('PLOT ')
1065  READ(2.IN1)
1066  C----------------------------------------------------------C
1067  C If LHSTRY is true, then read history file and make  :
1068  C the required history plots.:
:  C----------------------------------------------------------C
1069  IF ( LHSTRY ) THEN
1071 C
1072 STATUS = GETUR ( INBUF2, APDAT2, LEN )
1073 NPARS = LEN/8
1074 STATUS = FP2H1 ( APDAT2, INPAR2, NPARS, 4 )
1075 IT = INPAR2(1)
1076 ITH = INPAR2(2)
1077 ITHL = INPAR2(3)
1078 NP = INPAR2(4)
1079 NG = INPAR2(5)
1080 IDATE = INPAR2(6)
1081 ITIME = INPAR2(7)
1082 IDATE = INPAR2(8)
1083 IRTIME = INPAR2(9)
1084 N11 = INPAR2(10)
1085 IE = INPAR2(11)
1086 IPHI = INPAR2(12)
1087 IRHO = INPAR2(13)
1088 IRHOS = INPAR2(14)
1089 IHERM = INPAR2(15)
1090 JXVX = INPAR2(16)
1091 N1 = INPAR2(17)
1092 N2 = INPAR2(18)
1093 NT = INPAR2(19)
1094 IRS = INPAR2(20)
1095 ISPEC = INPAR2(21)
1096 JK = 22
1097 DO 5 J = 1,MMAX
1098 MPLOT(J) = INPAR2(JK)
1099 JK = JK+1
1100 5 CONTINUE
1101 STATUS = GETUR ( INBUF2, APDAT2, LEN )
1102 NPARS = LEN/8
1103 STATUS = FP2H2D ( APDAT2, HODAT2, NPARS )
1104 DT = HODAT2(1)
1105 DX = HODAT2(2)
1106 L = HODAT2(3)
1107 TIME = HODAT2(4)
1108 T1 = HODAT2(5)
1109 TEMP1 = HODAT2(6)
1110 TEMP2 = HODAT2(7)
1111 IIPLOT = ZERO
1112 NG1 = NG + 1
1113 GN = NG
1114 C
1115 IF ( NT .LE. NTH ) THEN
1116 JJ=1
1117 ELSE .
1118 JJ = (NT/NTH)
1119 IF ( MOD(NT,NTH) .GT. 0 ) JJ=JJ+1
1120 JJ = JJ - (ITHL/NTH)
1121 ENDIF
1122 C
1123 DO 100 J=1, JJ
1124 C
1125 IF ( NT .LE. NTH ) THEN
1126 IT = NT
1127 TIME = IT*DT
1128 MTH = IT + 1
1129 MTH1 = MTH + 1
1130 ELSE .
1131 ITMP = IT + NTH
1132 IF = MIN (ITMP, NT)
1133 IF ( ITHL .LE. 0 ) IT=NTH
1134 TIME = IT*DT
1135 MTH = IT - ITHL + 1
1136 MTH1 = MTH + 1
1137 ENDIF
1138 C
1139 NREC2 = MTH/1024
1140 IF ( MOD(MTH,1024) .GT. 0 ) NREC2 = NREC2+1
1141 C
1142 DD 40 KM = 1,MMAX
1143 KK = MPLDT(KM)
1144 IF ( KK .NE. O ) THEN
1145 JO = 1
1146 DO 35 II = 1,NREC2
1147 STATUS = GETUR ( INBUF2, APDAT2, LEN )
1148 NPARS = LEN/8
1149 STATUS = FPR2HD ( APDAT2, HODAT2, NPARS )
1150 DO 30 I = 1,NPARS
1151 ESEM(JO,KM) = HODAT2(I)
1152 JO = JO+1
1153 30 CONTINUE
1154 35 CONTINUE
1155 ENDIF
1156 40 CONTINUE
1157 C
1158 J1 = 1
1159 J2 = 1
1160 J3 = 1
1161 J4 = 1
1162 J5 = 1
1163 J6 = 1
1164 J7 = 1
1165 J8 = 1
1166 J9 = 1
1167 J10 = 1
1168 J11 = 1
1169 NREC2 = MTH1/1024
1170 IF ( MOD(MTH1,1024) .GT. O ) NREC2 = NREC2+1
1171 C
1172 DO 45 II = 1,NREC2
1173 IF ( II .EQ. NREC2 ) THEN
1174 IF ( MOD(MTH1,1024) .EQ. 1 ) GOTO 59
1175 ENDIF
1176 C
1177 C KE1 electron thermal energy.
1178 C
1179 STATUS = GETUR ( INBUF2, APDAT2, LEN )
1180 NPARS = LEN/8
1181 STATUS = FPR2HD ( APDAT2, HODAT2, NPARS )
1182 DO 48 I = 1,NPARS
1183 KE1(J1) = HODAT2(I)
1184 J1 = J1 + 1
1185 48 CONTINUE
1186 C
1187 C KE2 species 2 thermal energy.
1188 C
1189 STATUS = GETUR ( INBUF2, APDAT2, LEN )
1190 NPARS = LEN/8
1191 STATUS = FPR2HD ( APDAT2, HODAT2, NPARS )
1192 DO 49 I = 1,NPARS
1193 KE2(J2) = HODAT2(I)
1194 J2 = J2 + 1
1195 49 CONTINUE
1196 C
1197 C TE total energy.
1198 C
1199 STATUS = GETUR ( INBUF2, APDAT2, LEN )
1200 NPARS = LEN/8
1201 STATUS = FPR2HD ( APDAT2, HODAT2, NPARS )
1202 DO 50 I = 1,NPARS
1203 TE(J3) = HODAT2(I)
1204 J3 = J3 + 1
1205 50 CONTINUE
1206 C
1207 C ESE electrostatic field energy.
1208 C
1209 STATUS = GETUR ( INBUF2, APDAT2, LEN )
1210 NPARS = LEN/8
I1Z

1211 STATUS = FPR2HD ( APDAT2, HODAT2, NPARS )
1212 DO 51 I=1,NPARS
1213 ESE(J4) = HODAT2(I)
1214 J4 = J4 + 1
1215 CONTINUE

51

1216 C-----------------------------
1217 C ELKE -- electron kinetic energy.
1218 C-----------------------------
1219 STATUS = GETUR ( INBUF2, APDAT2, LEN )
1220 NPARS = LEN/8
1221 STATUS = FPR2HD ( APDAT2, HODAT2, NPARS )
1222 DO 52 I=1,NPARS
1223 ELKE(J5) = HODAT2(I)
1224 J5 = J5 + 1
1225 CONTINUE

52 CONTINUE

59 CONTINUE

1226 C-----------------------------
1227 C XT position of particle No.1.
1228 C-----------------------------
1229 STATUS = GETUR ( INBUF2, APDAT2, LEN )
1230 NPARS = LEN/8
1231 STATUS = FPR2HD ( APDAT2, HODAT2, NPARS )
1232 DO 53 I=1,NPARS
1233 XT(J6) = HODAT2(I)
1234 J6 = J6 + 1
1235 CONTINUE

53

1236 C-----------------------------
1237 C VXT velocity of particle No.1.
1238 C-----------------------------
1239 STATUS = GETUR ( INBUF2, APDAT2, LEN )
1240 NPARS = LEN/8
1241 STATUS = FPR2HD ( APDAT2, HODAT2, NPARS )
1242 DO 54 I=1,NPARS
1243 VXT(J7) = HODAT2(I)
1244 J7 = J7 + 1
1245 CONTINUE

54 CONTINUE

1246 C-----------------------------
1247 C P1X electron X momentum.
1248 C-----------------------------
1249 STATUS = GETUR ( INBUF2, APDAT2, LEN )
1250 NPARS = LEN/8
1251 STATUS = FPR2HD ( APDAT2, HODAT2, NPARS )
1252 DO 55 I=1,NPARS
1253 P1X(J8) = HODAT2(I)
1254 J8 = J8 + 1
1255 CONTINUE

55

1256 C-----------------------------
1257 C P2X species 2 X momentum.
1258 C-----------------------------
1259 STATUS = GETUR ( INBUF2, APDAT2, LEN )
1260 NPARS = LEN/8
1261 STATUS = FPR2HD ( APDAT2, HODAT2, NPARS )
1262 DO 56 I=1,NPARS
1263 P2X(J9) = HODAT2(I)
1264 J9 = J9 + 1
1265 CONTINUE

56 CONTINUE

1266 C-----------------------------
1267 C P1Y electron Y momentum.
1268 C-----------------------------
1269 STATUS = GETUR ( INBUF2, APDAT2, LEN )
1270 NPARS = LEN/8
1271 STATUS = FPR2HD ( APDAT2, HODAT2, NPARS )
1272 DO 57 I=1,NPARS
1273 P1Y(J10) = HODAT2(I)
1274 J10 = J10 + 1
1275 CONTINUE

57 CONTINUE

1276 C-----------------------------
1277 C P2Y species 2 Y momentum.
1278 C-----------------------------
1279 C-----------------------------
1280 C STATUS = GETUR ( INBUF2, APDAT2, LEN )
NPARS = LEN/8

STATUS = FPR2HD ( APDAT2, H0DAT2, NPARS )

DO 58 I=1,NPARS
   P2Y(J11) = H0DAT2(I)
   J11 = J11 + 1
58 CONTINUE

C 45 CONTINUE
C
CALL HISTRY
ITHL = IT
100 CONTINUE
ENDIF
C
C
C If LFIELDS or LSPACE is TRUE, then:
C
C (i) read X, VX, & VY arrays and make phase-space plots
C (ii) read field data and make plots if LFIELD is TRUE.
C
IF ( LFIELD .OR. LSPACE ) THEN
   STATUS = GETUR ( INBUF1, APDAT1, LEN )
   NPARS = LEN/8
   STATUS = FPR2HD ( APDAT1, INPAR1, NPARS, 4 )
   IT = INPAR1(1)
   ITH = INPAR1(2)
   ITHL = INPAR1(3)
   NP = INPAR1(4)
   NG = INPAR1(5)
   IDATE = INPAR1(6)
   ITIME = INPAR1(7)
   IDATE = INPAR1(8)
   IRTIME = INPAR1(9)
   N11 = INPAR1(10)
   IE = INPAR1(11)
   IPHI = INPAR1(12)
   IRHDO = INPAR1(13)
   ITHERM = INPAR1(15)
   IXVX = INPAR1(16)
   N1 = INPAR1(17)
   N2 = INPAR1(18)
   NT = INPAR1(19)
   IRS = INPAR1(20)
   ISPEC = INPAR1(21)
   JK = 22
   DO 6 J = 1,MMAX
      MPLOT(J) = INPAR1(JK)
      JK = JK+1
6 CONTINUE
   STATUS = GETUR ( INBUF1, APDAT1, LEN )
   NPARS = LEN/8
   STATUS = FPR2HD ( APDAT1, H0DAT1, NPARS )
   DT = HODAT1(1)
   DX = HODAT1(2)
   L = HODAT1(3)
   TIME = HODAT1(4)
   T1 = HODAT1(6)
   TEMP1 = HODAT1(6)
   TEMP2 = HODAT1(7)
   IIPLOT = ZERO
   NG1 = NG + 1
   GN = NG
   NRECS = NP/1024
   IF ( MOD(NP,1024) .GT. 0 ) NRECS = NRECS+1
   IF ( IRS .GT. 0 ) GO TO 1000
C
C Read X, VX, & VY at T=0.
C
C
JS = 1
JE = 1024
DO 10 I = 1,NRECS
STATUS = GETUR (INBUF1, APDAT1, LEN)
NPARS = LEN/8
STATUS = FPR2HD (APDAT1, X(JS), NPARS)
JS = JE + 1
JE = JE + NPARS
10 CONTINUE
C
JS = 1
JE = 1024
DO 11 I = 1,NRECS
STATUS = GETUR (INBUF1, APDAT1, LEN)
NPARS = LEN/8
STATUS = FPR2HD (APDAT1, VX(JS), NPARS)
JS = JE + 1
JE = JE + NPARS
11 CONTINUE
C
JS = 1
JE = 1024
DO 12 I = 1,NRECS
STATUS = GETUR (INBUF1, APDAT1, LEN)
NPARS = LEN/8
STATUS = FPR2HD (APDAT1, VY(JS), NPARS)
JS = JE + 1
JE = JE + NPARS
12 CONTINUE
C
=============================================================
C Plot the number of particles in each grid cell @ t=0.
C=============================================================
C Electrons.
C
IF ( NPTCL1 .GT. ZERO ) THEN
  DUMI = DZERO
  CALL PLTFOV (ONE,N1,DUMI,GN,X,NG,ONE,NPTCL1,TEMP1,TEMP2)
ENDIF
C
C Species No. 2.
C
IF ( NPTCL2 .GT. ZERO ) THEN
  DUMI = DZERO
  CALL PLTFOV (N11,NP,DUMI,GN,X,NG,TWO,NPTCL2,TEMP1,TEMP2)
ENDIF
C
C All particles.
C
IF ( NPTCLT .GT. ZERO ) THEN
  DUMI = DZERO
  CALL PLTFOV (ONE,NP,DUMI,GN,X,NG,THREE,NPTCLT,TEMP1,TEMP2)
ENDIF
C
C Plot the distribution function(s) @ t=0.
C=============================================================
C Electrons - F(VX).
C
IF ( IFVX01 ) THEN
  DUMI = DZERO
  DUMI2 = DZERO
  DUMI3 = ONE
  CALL PLTFOV (ONE,N1,DUMI,DUMI2,VX,100,FOUR,DUMI3,TEMP1,TEMP2)
ENDIF
C
C Electrons - F(VY).
C
IF ( IFVV01 ) THEN
1421 DUMI = DZERO
1422 DUMI2 = DZERO
1423 DUMI3 = ONE
1424 CALL PLTFOV (ONE, N1, DUMI, DUMI2, VX, 100, FIVE, DUMI3, TEMP1, TEMP2)
1425 ENDIF
1426
1427 C IF ( N11  LE. NP ) THEN
1428 C-----------------------------
1429 C Ions - F(VX).
1430 C-----------------------------
1431 C IF ( IFVX02 ) THEN
1432 DUMI = DZERO
1433 DUMI2 = DZERO
1434 DUMI3 = ONE
1435 CALL PLTFOV (N11, NP, DUMI, DUMI2, VX, 100, SIX, DUMI3, TEMP1, TEMP2)
1436 ENDIF
1437 C-----------------------------
1438 C-----------------------------
1439 C Ions - F(VY).
1440 C-----------------------------
1441 C IF ( IFVY02 ) THEN
1442 DUMI = DZERO
1443 DUMI2 = DZERO
1444 DUMI3 = ONE
1445 CALL PLTFOV (N11, NP, DUMI, DUMI2, VY, 100, SEVEN, DUMI3, TEMP1, TEMP2)
1446 ENDIF
1447 C-----------------------------
1448 C All particles - F(VX).
1449 C-----------------------------
1450 C IF ( IFVXOT ) THEN
1451 DUMI = DZERO
1452 DUMI2 = DZERO
1453 DUMI3 = ONE
1454 CALL PLTFOV (ONE, NP, DUMI, DUMI2, VX, 100, EIGHT, DUMI3, TEMP1, TEMP2)
1455 ENDIF
1456
1457 C Read/plot charge density values @ t=0.
1458 C===================================
1459 IF ( IRHO  NE. ZERO  ) THEN
1460 STATUS = GETUR ( INBUF1, APDAT1, LEN )
1461 STATUS = FPR2HD ( APDAT1, RHO, NG1 )
1462 IF ( IRHOO  ) THEN
1463 CALL PLOTF ( RHO, ONE )
1464 ENDIF
1465 ENDIF
1466
1467 C Read/plot smoothed charge density @ t=0.
1468 C===================================
1469 IF ( IRHOS  NE. ZERO  ) THEN
1470 STATUS = GETUR ( INBUF1, APDAT1, LEN )
1471 STATUS = FPR2HD ( APDAT1, RHO, NG1 )
1472 IF ( IRHOSO  ) THEN
1473 CALL PLOTF ( RHO, TWO )
1474 ENDIF
1475 ENDIF
1476
1477 C Read/plot electric potential @ t=0.
1478 C===================================
1479 IF ( IPHI  NE. ZERO  ) THEN
1480 STATUS = GETUR ( INBUF1, APDAT1, LEN )
1481 STATUS = FPR2HD ( APDAT1, PHI, NG1 )
1482 IF ( IPHIO  ) THEN
1483 CALL PLOTF ( PHI, THREE )
1484 ENDIF
1485 ENDIF
1486
1487 C Read/plot charge density values @ t=0.
1488 C===================================
1489 IF ( IRHO  NE. ZERO  ) THEN
1490 STATUS = GETUR ( INBUF1, APDAT1, LEN )
1491 STATUS = FPR2HD ( APDAT1, RHO, NG1 )
1492 IF ( IRHOO  ) THEN
1493 CALL PLOTF ( RHO, ONE )
1494 ENDIF
1495 ENDIF
1496
1497 C Read/plot smoothed charge density @ t=0.
1498 C===================================
1499 IF ( IRHOS  NE. ZERO  ) THEN
1500 STATUS = GETUR ( INBUF1, APDAT1, LEN )
1501 STATUS = FPR2HD ( APDAT1, RHO, NG1 )
1502 IF ( IRHOSO  ) THEN
1503 CALL PLOTF ( RHO, TWO )
1504 ENDIF
1505 ENDIF
1506
1507 C Read/plot electric potential @ t=0.
1508 C===================================
1509 IF ( IPHI  NE. ZERO  ) THEN
1510 STATUS = GETUR ( INBUF1, APDAT1, LEN )
1511 STATUS = FPR2HD ( APDAT1, PHI, NG1 )
1512 IF ( IPHIO  ) THEN
1513 CALL PLOTF ( PHI, THREE )
1514 ENDIF
1515 ENDIF
1516 ENDIF
C Read/plot electric field values \( \phi \) t=0.

C=============================================================================
IF (IE .NE. ZERO ) THEN
  STATUS = GETUR ( INBUF1, APDAT1, LEN )
  STATUS = FPR2HD ( APDAT1, E, NG1 )
ENDIF
C=============================================================================
C Read/plot electrostatic energy spectrum.
C=============================================================================
IF (ISPEC .NE. ZERO ) THEN
  NG2P = NG/2 + 1
  STATUS = GETUR ( INBUF1, APDAT1, LEN )
  STATUS = FPR2HD ( APDAT1, EK, NG2P )
  IF (ISPECO) THEN
    CALL PSPEC
  ENDIF
ENDIF
C=============================================================================
C Read relative electron (ion) temperature.
IF (ITHERM .GT. ZERO ) THEN
  STATUS = GETUR ( INBUF1, APDAT1, LEN )
  STATUS = FPR2HD ( APDAT1, THRMX1, NG1 )
  IF (N2 .GE. ONE ) THEN
    STATUS = GETUR ( INBUF1, APDAT1, LEN )
    STATUS = FPR2HD ( APDAT1, THRMX2, NG1 )
  ENDIF
ENDIF
C=============================================================================
C Plot temperature profiles \( \phi \) t=0.
C=============================================================================
C Electrons.
IF ( ITMP01 ) THEN
  CALL PLOTF ( THRMX1, FIVE )
ENDIF
C=============================================================================
C Ions.
IF (ITMP02) THEN
  CALL PLOTF ( THRMX2, SIX )
ENDIF
C=============================================================================
1000 CONTINUE
C=============================================================================
C Read relative electron (ion) temperatures.
IF (ITHERM .GT. ZERO ) THEN
  IF ( (IT/ITHERM)*ITHERM .EQ. IT ) THEN
    STATUS = GETUR ( INBUF1, APDAT1, LEN )
    STATUS = FPR2HD ( APDAT1, THRMX1, NG1 )
    IF (N2 .GE. ONE ) THEN
      STATUS = GETUR ( INBUF1, APDAT1, LEN )
      STATUS = FPR2HD ( APDAT1, THRMX2, NG1 )
    ENDIF
  ENDIF
C=============================================================================
C Plot temperature profiles.
C=============================================================================
C Electrons.
IF (ITMP1) THEN
  IF (ITMP1 .NE. ZERO ) THEN
    IF ( (IT/ITMP1)*ITMP1 .EQ. IT ) THEN
      ENDIF
    ENDIF
  ENDIF
C=============================================================================
C Ions.
IF (ITMP2) THEN
  ENDIF
ENDIF
CALL PLOTF ( THRNX1, FIVE )
ENDIF
ENDIF
C-----------------------------------------------
C Ion.
C-----------------------------------------------
IF ( ITMP2 .NE. ZERO ) THEN
  IF (((IT/ITMP2)*ITMP2 .EQ. IT ) THEN
    CALL PLOTF ( THRNX2, SIX )
  ENDIF
ENDIF
ENDIF
ENDIF
C-----------------------------------------------
C Read: X @ t=IT , VX, Vy @ t=IT+.5
C-----------------------------------------------
IF ( (IT/IXVX) * IXVX .EQ. IT ) THEN

C========================================
JS = 1
JE = 1024
DO 20  I = 1,NRECS
  STATUS = GETUR ( INBUF1, APDAT1, LEN )
  NPARS = LEN/8
  STATUS = FPR2HD ( APDAT1, X(JS), NPARS )
  JS = JE + 1
  JE = JE + NPARS
20  CONTINUE
C
JS = 1
JE = 1024
DO 21  I = 1,NRECS
  STATUS = GETUR ( INBUF1, APDAT1, LEN )
  NPARS = LEN/8
  STATUS = FPR2HD ( APDAT1, VX(JS), NPARS )
  JS = JE + 1
  JE = JE + NPARS
21  CONTINUE
C
IF ( T1 .NE. DZERO ) THEN

C========================================
C Phase-space and velocity-space plots of all species.
C========================================
C X-VX. Electrons.
C========================================
IF ( (IT/IXVX) * IXVX1 .EQ. IT ) THEN
  VL = DZERO
  VU = DZERO
  CALL PLOTXV ( ONE, N1, VL, VU, ONE, ONE )
ENDIF
ENDIF
C========================================
C X-VX. Species No. 2.
C========================================
1631 IF ( N2 .GE. 1 ) THEN
1632 IF ( IXVX2 .NE. ZERO ) THEN
1633 IF ( ((IT/IXVX2)*IXVX2 .EQ. IT ) THEN
1634 VL = DZERO
1635 VU = DZERO
1636 CALL PLOTXV ( N1, NP, VL, VU, ONE, TWO )
1637 ENDIF
1638 ENDIF
1639 C-------------------------------------
1640 C X-VX. All species.
1641 C-------------------------
1642 IF ( IXVXT .NE. ZERO ) THEN
1643 IF ( ((IT/IXVXT)*IXVXT .EQ. IT ) THEN
1644 VL = DZERO
1645 VU = DZERO
1646 CALL PLOTXV ( ONE, NP, VL, VU, ONE, THREE )
1647 ENDIF
1648 ENDIF
1649 C-------------------------------------
1650 C X-VY. Electrons.
1651 C-------------------------
1652 IF ( T1 .NE. DZERO ) THEN
1653 IF ( IXVY1 .NE. ZERO ) THEN
1654 IF ( ((IT/IXVY1)*IXVY1 .EQ. IT ) THEN
1655 VL = DZERO
1656 VU = DZERO
1657 CALL PLOTXV ( ONE, N1, VL, VU, TWO, ONE )
1658 ENDIF
1659 ENDIF
1660 C-------------------------------------
1661 C VX-VY. Electrons.
1662 C-------------------------
1663 IF ( IXVYT .NE. ZERO ) THEN
1664 IF ( ((IT/IXVYT)*IXVYT .EQ. IT ) THEN
1665 VL = DZERO
1666 VU = DZERO
1667 CALL PLOTXV ( ONE, NP, VL, VU, TWO, TWO )
1668 ENDIF
1669 C-------------------------------------
1670 C X-VY. Species No. 2.
1671 C-------------------------
1672 IF ( IXVY2 .NE. ZERO ) THEN
1673 IF ( ((IT/IXVY2)*IXVY2 .EQ. IT ) THEN
1674 VL = DZERO
1675 VU = DZERO
1676 CALL PLOTXV ( N11, NP, VL, VU, TWO, TWO )
1677 ENDIF
1678 C-------------------------------------
1679 C VX-VY. Species No. 2.
1680 C-------------------------
1681 IF ( IXVY2 .NE. ZERO ) THEN
1682 IF ( ((IT/IXVY2)*IXVY2 .EQ. IT ) THEN
1683 VMU = DZERO
1684 CALL PLTVXY ( ONE, N1, VMU, ONE )
1685 ENDIF
1686 C-------------------------------------
1687 C X-VY. All species.
1688 C-------------------------
1689 IF ( IXVYT .NE. ZERO ) THEN
1690 IF ( ((IT/IXVYT)*IXVYT .EQ. IT ) THEN
1691 VL = DZERO
1692 VU = DZERO
1693 CALL PLOTXV ( ONE, NP, VL, VU, TWO, THREE )
1694 ENDIF
1695 C-------------------------------------
1696 C VX-VY. All species.
C-------------------------------------------------------------------------------------------------
1701 IF ( IVXVYT .NE. ZERO ) THEN
1702 IF ( (IT/IVXVYT)*IVXVYT .EQ. IT ) THEN
1703 VMU = DZERO
1704 CALL PLTVXY ( ONE, NP, VMU, TRHEE )
1705 ENDIF
1706 ENDIF
1707 ENDIF
1708 ENDIF
1709 ENDIF
1710 ENDIF
1711 C-----------------------------------------------------------------:
1712 C Check to see if finished.
1713 C-----------------------------------------------------------------:
1714 IF ( IT .GE. NT ) GO TO 2000
1715 C-----------------------------------------------------------------:
1716 C Read time variables.
1717 C-----------------------------------------------------------------:
1718 STATUS = GETUR ( INBUF1, APDAT1, LEN )
1719 NPARS = LEN/8
1720 STATUS = FPI2HI ( APDAT1, INPAR1, NPARS, 4 )
1721 IT = INPAR1(1)
1722 ITH = INPAR1(2)
1723 TIME = IT*DT
1724 IF ( IXVX .NE. ZERO ) THEN
1725 IF ( (IT/IXVX) * IXVX .EQ. IT ) THEN
1726 C===========================================================
1727 C Plot the number of particles in each grid cell.
1728 C===========================================================
1729 C Electrons.
1730 C-----------------------------------------------------------------
1731 IF ( NPTCL1 .GT. ZERO ) THEN
1732 IF ( (IT/NPTCL1) * NPTCL1 .EQ. IT ) THEN
1733 DUMI = DZERO
1734 CALL PLTFOV (ONE,N1,DUMI,GN,X,NG,ONE,NPTCL1,
1735 TEMPI,TEMP2)
1736 ENDIF
1737 ENDIF
1738 C-----------------------------------------------------------------
1739 C Species No. 2.
1740 C-----------------------------------------------------------------
1741 IF ( NPTCL2 .GT. ZERO ) THEN
1742 IF ( (IT/NPTCL2) * NPTCL2 .EQ. IT ) THEN
1743 DUMI = DZERO
1744 CALL PLTFOV (N11,NP,DUMI,GN,X,NG,TWO,NPTCL2,
1745 TEMP1,TEMP2)
1746 ENDIF
1747 ENDIF
1748 C-----------------------------------------------------------------
1749 C All particles.
1750 C-----------------------------------------------------------------
1751 IF ( NPTCLT .GT. ZERO ) THEN
1752 IF ( (IT/NPTCLT) * NPTCLT .EQ. IT ) THEN
1753 DUMI = DZERO
1754 CALL PLTFOV (ONE,NP,DUMI,GN,X,NG,THREE,NPTCLT,
1755 TEMP1,TEMP2)
1756 ENDIF
1757 ENDIF
1758 C================================================================================:
1759 C Plot the distribution functions.
1760 C================================================================================:
1761 C F(VX) - Electrons.
1762 C-----------------------------------------------------------------
1763 IF ( IFVX1 .NE. ZERO ) THEN
1764 IF ( (IT/IFVX1) * IFVX1 .EQ. IT ) THEN
1765 VRL = DZERO
1766 VRU = DZERO
1767 CALL CARTMM ( N1, VRL, VRU, VX )
1768 VM = MAX ( ABS(VRL), ABS(VRU) )
1769 CALL PLTFOV (ONE,N1,-VM,VM,VM,CENT,FOUR,IFVX1,
1770 TEMP1,TEMP2)
ENDIF

C-------------------
C F(VX) - Ions.
C-------------------
IF ( N2 .GE. ONE ) THEN
  IF ( (IT/IFVX2) * IFVX2 .EQ. IT ) THEN
    RU = VX(N11)
    RL = RU
    DO 60 I = N11,NP
      RU = MAX ( RU, VX(I) )
      RL = MIN ( RL, VX(I) )
    CONTINUE
    VM = MAX ( ABS(RL), ABS(RU) )
    CALL PLTFOV (N11,NP,-VM,VM,VX,CENT,SIX,IFVX2, TEMP1,TEMP2)
  ENDIF
ENDF

C-------------------
C F(VY) - Electrons.
C-------------------
IF ( T1 .NE. DZERO ) THEN
  IF ( (IT/IFVY1) * IFVY1 .EQ. IT ) THEN
    VRL = DZERO
    VRU = DZERO
    CALL CARTMM ( N1, VRL, VRU, VY )
    VM = MAX ( ABS(VRL), ABS(VRU) )
    CALL PLTFOV (ONE,N1,-VM,VM,VY,CENT,FIVE,IFVY1, TEMP1,TEMP2)
  ENDIF
ENDF

C-------------------
C F(VY) - Ions.
C-------------------
IF ( N2 .GE. ONE ) THEN
  IF ( (IT/IFVY2) * IFVY2 .EQ. IT ) THEN
    RU = VY(N11)
    RL = RU
    DO 70 I = N11,NP
      RU = MAX ( RU, VY(I) )
      RL = MIN ( RL, VY(I) )
    CONTINUE
    VM = MAX ( ABS(RL), ABS(RU) )
    CALL PLTFOV (N11,NP,-VM,VM,VY,CENT,SEVEN,IFVY2, TEMP1,TEMP2)
  ENDIF
ENDF

C-------------------
C F(VX) - All particles.
C-------------------
IF ( N2 .GE. ONE ) THEN
  IF ( (IT/IFVXT) * IFVXT .EQ. IT ) THEN
    VM = 1.0
    CALL PLTFOV (ONE,NP,-VM,VM,VX,CENT,EIGHT,IFVXT, TEMP1,TEMP2)
  ENDIF
ENDF

C==-------------------------------------------------------------
C  Read/plot charge density values.
C==-------------------------------------------------------------

C==============================================================
1841 IF ( IRHO .NE. ZERO ) THEN
1842   IF ( ( IT/IRHO )* IRHO .EQ. IT ) THEN
1843     STATUS = GETUR ( INBUF1, APDAT1, LEN )
1844     STATUS = FPR2HD ( APDAT1, RHO, NG1 )
1845     IF ( IRHO1 .GT. ZERO ) THEN
1846       IF ( ( IT/IRHO1 )*IRHO1 .EQ. IT ) THEN
1847         CALL PLOTF ( RHO, ONE )
1848       ENDIF
1849   ENDIF
1850 ENDIF
1851 ENDIF
1852 C=============================================================================
1853  C Read/plot smoothed charge density (RHO).
1854  C=============================================================================
1855 IF ( IRHOS .NE. ZERO ) THEN
1856   IF ( ( IT/IRHOS )*IRHOS .EQ. IT ) THEN
1857     STATUS = GETUR ( INBUF1, APDAT1, LEN )
1858     STATUS = FPR2HD ( APDAT1, RHO, NG1 )
1859     IF ( IRHOS1 .GT. ZERO ) THEN
1860       IF ( ( IT/IRHOS1 )*IRHOS1 .EQ. IT ) THEN
1861         CALL PLOTF ( RHO, TWO )
1862       ENDIF
1863   ENDIF
1864 ENDIF
1865 ENDIF
1866 C=============================================================================
1867  C Read/plot electric potential (Phi).
1868  C=============================================================================
1869 IF ( IPHI .NE. ZERO ) THEN
1870   IF ( ( IT/IPHI )*IPHI .EQ. IT ) THEN
1871     STATUS = GETUR ( INBUF1, APDAT1, LEN )
1872     STATUS = FPR2HD ( APDAT1, PHI, NG1 )
1873     IF ( IPHI1 .GT. ZERO ) THEN
1874       IF ( ( IT/IPHI1 )*IPHI1 .EQ. IT ) THEN
1875         CALL PLOTF ( PHI, THREE )
1876     ENDIF
1877 ENDIF
1878 ENDIF
1879 ENDIF
1880 C=============================================================================
1881  C Read/plot electric field values.
1882  C=============================================================================
1883 IF ( IE .NE. ZERO ) THEN
1884   IF ( ( IT/IE )*IE .EQ. IT ) THEN
1885     STATUS = GETUR ( INBUF1, APDAT1, LEN )
1886     STATUS = FPR2HD ( APDAT1, E, NG1 )
1887     IF ( IE1 .GT. ZERO ) THEN
1888       IF ( ( IT/IE1 )*IE1 .EQ. IT ) THEN
1889         CALL PLOTF ( E, FOUR )
1890     ENDIF
1891 ENDIF
1892 ENDIF
1893 ENDIF
1894 C=============================================================================
1895  C Read/plot electrostatic energy spectrum.
1896  C=============================================================================
1897 IF ( ISPEC .NE. ZERO ) THEN
1898   IF ( ( IT/ISPEC )*ISPEC .EQ. IT ) THEN
1899     NG2P = NG/2 + 1
1900     STATUS = GETUR ( INBUF1, APDAT1, LEN )
1901     STATUS = FPR2HD ( APDAT1, EK, NG2P )
1902     IF ( ISPEC1 .GT. ZERO ) THEN
1903       IF ( ( IT/ISPEC1 )*ISPEC1 .EQ. IT ) THEN
1904         CALL PSPEC
1905     ENDIF
1906 ENDIF
1907 ENDIF
1908 ENDIF
1909 C
1910 GOTO 1000
C End of loop.

C-----------------------------
C 2000 CONTINUE
C
C IF ( JFLAG .LE. 0 ) THEN
WRITE(7,1050) CDMIN
WRITE(7,1051) CDMAX
WRITE(7,1052) SDMIN
WRITE(7,1053) SDMAX
WRITE(7,1054) EPMIN
WRITE(7,1055) EPMAX
WRITE(7,1056) EFMIN
WRITE(7,1057) EFMAX
WRITE(7,1058) ETMIN
WRITE(7,1059) ETMAX
WRITE(7,1060) ITMIN
WRITE(7,1061) ITMAX
ENDIF
C
1050 FORMAT (///,3X, 'MIN. CHARGE DENSITY = ', 1PD22.14 )
1051 FORMAT ( 3X, 'MAX. CHARGE DENSITY = ', 1PD22.14, / )
1052 FORMAT ( 3X, 'MIN. SMOOTHED CHARGE DENSITY = ', 1PD22.14 )
1053 FORMAT ( 3X, 'MAX. SMOOTHED CHARGE DENSITY = ', 1PD22.14, / )
1054 FORMAT ( 3X, 'MIN. ELECTRIC POTENTIAL = ', 1PD22.14 )
1055 FORMAT ( 3X, 'MAX. ELECTRIC POTENTIAL = ', 1PD22.14, / )
1056 FORMAT ( 3X, 'MIN. ELECTRIC FIELD = ', 1PD22.14 )
1057 FORMAT ( 3X, 'MAX. ELECTRIC FIELD = ', 1PD22.14, / )
1058 FORMAT ( 3X, 'MIN. ELECTRON TEMPERATURE = ', 1PD22.14 )
1059 FORMAT ( 3X, 'MAX. ELECTRON TEMPERATURE = ', 1PD22.14, / )
1060 FORMAT ( 3X, 'MIN. SPECIES 2 TEMPERATURE = ', 1PD22.14 )
1061 FORMAT ( 3X, 'MAX. SPECIES 2 TEMPERATURE = ', 1PD22.14 )
C
END
END
C
C SUBROUTINE PICTUR (I)

SUBROUTINE PICTUR (I)
DATA( XX1(I), 1=1,5 ) /-.005, 0.005, 0.005, 0.005, -.005/
DATA( YY1(I), 1=1,5 ) /-.005, 0.005, 0.005, 0.005, 0.005/
DATA( XX2(I), 1=1,5 ) /-.005, 0.000, 0.000, 0.000, 0.000/
DATA( YY2(I), 1=1,5 ) /0.000, 0.000, 0.000, 0.000, -0.005/
GO TO ( 1,2 ), I
1 CALL PICTRC ( XX1, YY1, 5 )
RETURN
2 CALL PICTRC ( XX2, YY2, 5 )
RETURN
END
SUBROUTINE HISTRY

C Plot energies, etc. vs time.

INCLUDE ( MPARAM )
INCLUDE ( MCNTRL )
INCLUDE ( MTIME )
INCLUDE ( MPTCL )
INCLUDE ( MOTHERM )
INCLUDE ( HPlot )

DOUBLE PRECISION TIM(NTH1), TIMTH(NTH1)
INTEGER DUMMY, TITLE

PARAMETER ( DUMMY = -1 )

TL = ITHL * DT
MTH = IT - ITHL + 1

DO 1 I = 1, MTH
   TIM(I) = (I-1)*DT + TL
   CONTINUE

C Plot mode energies.

C

IF ( IESEM ) THEN
   TITLE = 1
   DO 50 KM = 1, MMAX
      K = MPLOT(KM)
      IF ( K .EQ. 0 ) GO TO 52
      CALL PLTHST ( ESEM(1,KM), TIM, MTH, TL, TIME, ONE, TITLE, DUMMY, K )
   50 CONTINUE
52 CONTINUE
ENDIF

C Plot field energy.

C

IF ( IESE ) THEN
   TITLE = 2
   CALL PLTHST ( ESE, TIM, MTH, TL, TIME, ZERO, TITLE, DUMMY, DUMMY )
ENDIF

C Plot electron kinetic energy.

C

IF ( IELKE ) THEN
   TITLE = 12
   CALL PLTHST ( ELKE, TIM, MTH, TL, TIME, ZERO, TITLE, DUMMY, DUMMY )
ENDIF

C Plot avg. drift momentum per ptcl in units of MC.

C i.e., plot GAMMA*V/C.

C

P1XL = P1X(MTH)
P1YL = P1Y(MTH)

C Electrons - avg. X direction drift momentum.

C

IF ( IP1X ) THEN
   TITLE = 3
   CALL PLTHST ( P1X, TIM, MTH, TL, TIME, ZERO, TITLE, ONE, DUMMY )
ENDIF

C
C Electrons - avg. Y direction drift momentum.

C=====================================================================
IP1Y THEN
TITLE = 4
CALL PLTHST ( P1Y, TIM, MTH, TL, TIME, ZERO,
TITL, ONE, DUMMY )
ENDIF
IF ( N2 .NE. ZERO ) THEN
P2XL = P2X(MTH)
P2YL = P2Y(MTH)
C=====================================================================
C Species 2 - avg. X direction drift momentum.
C=====================================================================
IF ( IP2X ) THEN
TITLE = 5
CALL PLTHST ( P2X, TIM, MTH, TL, TIME, ZERO,
TITL, TWO, DUMMY )
ENDIF
C=====================================================================
C Species 2 - avg. Y direction drift momentum.
C=====================================================================
IF ( IP2Y ) THEN
TITLE = 6
CALL PLTHST ( P2Y, TIM, MTH, TL, TIME, ZERO,
TITL, TWO, DUMMY )
ENDIF
C=====================================================================
C Plot position of particle no. 1.
C=====================================================================
IF ( IXT ) THEN
TITLE = 7
DO 20 I = 1,MTH
XT(I) = XT(I) * DX
20 CONTINUE
CALL PLTHST ( XT, TIM, MTH, TL, TIME, ZERO,
TITL, DUMMY, DUMMY )
ENDIF
C=====================================================================
C Plot velocity of particle no. 1.
C=====================================================================
IF ( IVXT ) THEN
DXDT = DX/DT
DO 598 I = 1,MTH
TIM(I) = TIM(I) - 0.5D0*DT
598 CALL PLTHST ( VXT, TIM, MTH, TL, TIME, ZERO,
TITL, DUMMY, DUMMY )
DO 599 I = 1,MTH
599 TIM(I) = TIM(I) + 0.5D0*DT
599 ENDIF
C=====================================================================
C Plot total energy.
C=====================================================================
IF ( ITE ) THEN
TITLE = 9
CALL PLTHST ( TE, TIM, MTH, TL, TIME, ONE,
TITL, DUMMY, DUMMY )
ENDIF
C=====================================================================
C Plot electron thermal energy.
C=====================================================================
IF ( IT .EQ. NT ) THEN
IF ( Itherm .NE. ZERO ) THEN
NPTH = IT/Itherm +1
DO 200 I=1,NPTH
TIMTH(I) = (i-1)*iTherm*DT
200 CONTINUE
ENDIF
C=====================================================================
C Plot electron thermal energy.
TITLE = 10
CALL PLTHST ( KE1, TIMTH, NPTH, TL, TIME, ZERO,
               TITLE, ONE, DUMMY )
ENDIF

C
IF ( N2 .NE. ZERO ) THEN
  C Plot species 2 thermal energy.
  IF ( IKE2 ) THEN
    TITLE = 11
    CALL PLTHST ( KE2, TIMTH, NPTH, TL, TIME, ZERO,
                  TITLE, TWO, DUMMY )
  ENDIF
ENDIF
ENDIF
ENDIF
C
RETURN
END

C===================================================================
C SUBROUTINE PLTHST(REC,TIM,MTH,TL,TU,LINLOG,TITLE,IS,MODE)
C===================================================================
C Plot time history, linear or log.
C===================================================================
INCLUDE ( MPARAM )
INCLUDE ( MCNTRL )
INCLUDE ( MPTCL )
INCLUDE ( MTIME )
DOUBLE PRECISION REC (MTH), TIM (MTH)
REAL  XAXIS, YAXIS, XORIG, XMAX, YORIG, YMAX, TIMR(IOOO),
      XSTEP, YCYCLE, RECR(IOOO), TIMER
INTEGER TITLE
C
CALL PAGE ( 11.0, 8.50 )
XAXIS  = 9.5
YAXIS  = 6.0
CALL AREA2D ( XAXIS, YAXIS )
C
50 IF ( TITLE .NE. 1 ) GO TO 100
CALL XNAME ( 'TIMES', 100 )
CALL YNAME ( 'MODE ENERGY', 100 )
CALL HEADIN ( 'MODE ENERGY VS TIMES', 100, 2., 1 )
IJPLOT = IJPLOT + 1
WRITE(*,FMT='(2X,  "PLOT ",I3," - MODE ",I2," ENERGY vs. TIME")')  IJPLOT, MODE
GO TO 1100
C
100 IF ( TITLE .NE. 2 ) GO TO 200
CALL XNAME ( 'TIMES', 100 )
CALL YNAME ( 'FIELD ENERGY', 100 )
CALL HEADIN ( 'FIELD ENERGY VS TIMES', 100, 2., 1 )
IJPLOT = IJPLOT + 1
WRITE(*,FMT='(2X,  "PLOT ",I3," - FIELD", 
                        " ENERGY vs. TIME")') IJPLOT
GO TO 1100
C
200 IF ( TITLE .NE. 3 ) GO TO 300
CALL XNAME ( 'TIMES', 100 )
CALL YNAME ( 'AVERAGE X-MOMENTUM (MC)', 100 )
CALL HEADIN ( 'ELECTRONS: AVG. X-MOMENTUM VS TIMES', 
                              100, 2., 1 )
IJPLOT = IJPLOT + 1
WRITE(*,FMT='(2X,  "PLOT ",I3," - ELECTRONS", 
                  " AVG. X-MMTM. vs. TIME")') IJPLOT
GO TO 1100
C
3050 C
3051 300 IF ( TITLE .NE. 4 ) GO TO 400
3052 CALL XNAME ('TIMES', 100)
3053 CALL YNAME ('AVERAGE Y-MOMENTUM (MC)$', 100)
3054 CALL HEADIN ('ELECTRONS: AVG. Y-MOMENTUM VS TIMES',
3055 100, 2., 1)
3056 IJPLOT = IJPLOT + 1
3057 WRITE(*,FMT='(2X,''PLOT '' ,I3,'', ELECTRONS'',
3058 '' AVG. Y-MOMENT. vs. TIME'')') IJPLOT
3059 GO TO 1100
3060 C
3061 400 IF ( TITLE .NE. 5 ) GO TO 500
3062 CALL XNAME ('TIMES', 100)
3063 CALL YNAME ('AVERAGE X-MOMENTUM (MC)$', 100)
3064 CALL HEADIN ('SPECIES 2: AVG. X-MOMENTUM VS TIMES',
3065 100, 2., 1)
3066 IJPLOT = IJPLOT + 1
3067 WRITE(*,FMT='(2X,''PLOT '' ,I3,'', SPECIES 2'',
3068 '' AVG. X-MOMENT. vs. TIME'')') IJPLOT
3069 GO TO 1100
3070 C
3071 500 IF ( TITLE .NE. 6 ) GO TO 600
3072 CALL XNAME ('TIMES', 100)
3073 CALL YNAME ('AVERAGE Y-MOMENTUM (MC)$', 100)
3074 CALL HEADIN ('SPECIES 2: AVG. Y-MOMENTUM VS TIMES',
3075 100, 2., 1)
3076 IJPLOT = IJPLOT + 1
3077 WRITE(*,FMT='(2X,''PLOT '' ,I3,'', SPECIES 2'',
3078 '' AVG. Y-MOMENT. vs. TIME'')') IJPLOT
3079 GO TO 1100
3080 C
3081 600 IF ( TITLE .NE. 7 ) GO TO 700
3082 CALL XNAME ('TIMES', 100)
3083 CALL YNAME ('X POSITIONS', 100)
3084 CALL HEADIN ('POSITION OF PARTICLE NO. 1$', 100, 2.. 1)
3085 IJPLOT = IJPLOT + 1
3086 WRITE(*,FMT='(2X,''PLOT '' ,I3,'', POSITION '' ,
3087 '' OF PARTICLE 1 vs. TIME'')') IJPLOT
3088 GO TO 1100
3089 C
3090 700 IF ( TITLE .NE. 8 ) GO TO 800
3091 CALL XNAME ('TIMES', 100)
3092 CALL YNAME ('X VELOCITIES', 100)
3093 CALL HEADIN ('VELOCITY OF PARTICLE NO. 1$', 100, 2.. 1)
3094 IJPLOT = IJPLOT + 1
3095 WRITE(*,FMT='(2X,''PLOT '' ,I3,'', VELOCITY '' ,
3096 '' OF PARTICLE 1 vs. TIME'')') IJPLOT
3097 GO TO 1100
3098 C
3099 800 IF ( TITLE .NE. 9 ) GO TO 900
3100 CALL XNAME ('TIMES', 100)
3101 CALL YNAME ('TOTAL ENERGIES', 100)
3102 CALL HEADIN ('TOTAL ENERGIES', 100, 2.. 1)
3103 IJPLOT = IJPLOT + 1
3104 WRITE(*,FMT='(2X,''PLOT '' ,I3,'', TOTAL'' ,
3105 '' ENERGY vs. TIME'')') IJPLOT
3106 GO TO 1100
3107 C
3108 900 IF ( TITLE .NE. 10 ) GO TO 1000
3109 CALL XNAME ('TIMES', 100)
3110 CALL YNAME ('THERMAL ENERGIES (EV)$', 100)
3111 CALL HEADIN ('THERMAL ENERGIES - ELECTRONS$', 100, 2.. 1)
3112 IJPLOT = IJPLOT + 1
3113 WRITE(*,FMT='(2X,''PLOT '' ,I3,'', ELECTRON THERMAL'',
3114 '' ENERGY vs. TIME'')') IJPLOT
3115 GO TO 1100
3116 C
3117 1000 IF ( TITLE .NE. 11 ) GO TO 1050
3118 CALL XNAME ('TIMES', 100)
3119 CALL YNAME ('THERMAL ENERGIES (EV)$', 100)
3120 CALL HEADIN ('THERMAL ENERGIES - SPECIES 2$', 100, 2.. 1)
IJPLOT = IJPLOT + 1
WRITE(*,FMT='(2X,’PLOT ’',I3,’ - SPECIES 2 THERMAL’,
’ ENERGY vs. TIME’')) IJPLOT
GO TO 1100

C
1050 IF ( TITLE .NE. 12 ) GO TO 1100
CALL XNAME ( ’TIMES’, 100 )
CALL YNAME ( ’KINETIC ENERGY (EV)’ , 100 )
CALL HEADIN ( ’ELECTRON KINETIC ENERGY’ , 100, 2., 1)
IJPLOT = IJPLOT + 1
WRITE(*,FMT='(2X,’PLOT ’',I3,’ - ELECTRON KINETIC’,
’ ENERGY vs. TIME’')) IJPLOT
GO TO 1100

C
1100 XORIG = TL
XMAX = TU
YORIG = RL
YMAX = RU
IMARK = 0
IF ( TITLE .NE. 10 ) XORIG = 0.0
CALL YAXANG ( 0. )
ELSE
RLT = MAX( RL, 1.0D-6*RU )
IF ( RLT .EQ. RL ) THEN
YORIG = RLT
YCYCLE = 1.0
ELSE
RLT = MAX( RL, 1.0D-12*RU )
IF ( RLT .EQ. RL ) THEN
CALL MESSAG ( ’SOME OUT OF RANGE POINTS NOT PLOTTED’,
100.0,3.0,3 )
ENDIF
ENDIF
XSTEP = ( TU-TL ) / 9.5
CALL YLOG ( XORIG, XSTEP, YORIG, YCYCLE )
ENDIF

C *************** Code to manually set the scales. ***************
C
IF(TITLE.EQ.1 .AND. LINLOG.EQ.0 .AND. MODE.EQ.1) THEN
YORIG=0.0
YMAX=30.0E-5
CALL GRAF ( XORIG, ’SCALE’, XMAX, YORIG, ’SCALE’, YMAX )
ENDIF
IF(TITLE.EQ.1 .AND. LINLOG.EQ.1 .AND. MODE.EQ.1) THEN
YORIG=2.0E-16
YCYCLE=0.5
XSTEP=(TU-TL)/9.5
CALL YLOG(XORIG,XSTEP,YORIG,YCYCLE)
ENDIF
IF(TITLE.EQ.1 .AND. LINLOG.EQ.0 .AND. MODE.EQ.2) THEN
YORIG=0.0
YMAX=35.0E-5
CALL GRAF ( XORIG, ’SCALE’, XMAX, YORIG, ’SCALE’, YMAX )
ENDIF
IF(TITLE.EQ.1 .AND. LINLOG.EQ.1 .AND. MODE.EQ.2) THEN
YORIG=3.0E-16
YCYCLE=0.5
XSTEP=(TU-TL)/9.5
CALL YLOG(XORIG,XSTEP,YORIG,YCYCLE)
ENDIF
IF(TITLE.EQ.1 .AND. LINLOG.EQ.0 .AND. MODE.EQ.3) THEN
YORIG=0.0
YMAX=7.0E-5
CALL GRAF ( XORIG, ’SCALE’, XMAX, YORIG, ’SCALE’, YMAX )
ENDIF
IF(TITLE.EQ.1.AND.LINLOG.EQ.1.AND.MODE.EQ.3) THEN
  YORIG=6.0E-17
  YCYCLE=0.5
  XSTEP=(TU-TL)/9.5
  CALL YLOG(XORIG,XSTEP,YORIG,YCYCLE)
ENDIF

IF(TITLE.EQ.12) THEN
  YORIG=6.0E-17
  YCYCLE=0.5
  XSTEP=(TU-TL)/9.5
  CALL YLOG(XORIG,XSTEP,YORIG,YCYCLE)
ENDIF

IF(TITLE.EQ.3) THEN
  YORIG=6.0E-17
  YMAX=85.0E-5
  CALL GRAF(XORIG,'SCALE',XMAX,YORIG,'SCALE',YMAX)
ENDIF

*********** Manual scale setting ends here. ***********

If more than 1000 timesteps, plot every 2nd, 3rd, etc.

IK = 1
IF ( MTH.GT.1000 ) THEN
  IK = (MTH/1000)
  IF ( MOD(MTH,1000).GT.0 ) IK = IK+1
ENDIF

JK = MTH/IK
JJ = 1
DO 5 1=1,MTH,IK
  TIMR(JJ) = TIM(I)
  RECR(JJ) = REC(I)
  JJ = JJ+1
5 CONTINUE

TIMER = TIME
CALL CURVE(TIMR,RECR,JK,IMARK)

IF ( MODE.NE. -1 ) THEN
  CALL MESSAG('MODE NO. : $',100,7.2,1.2)
  CALL INTO ( MODE, 'ABUT', 'ABUT')
ENDIF

IF ( IS.NE. -1 ) THEN
  CALL MESSAG('SPECIES : $',100,7.2,1.2)
  CALL INTO ( IS, 'ABUT', 'ABUT')
ENDIF

CALL MESSAG('RUN TIME : $',100,7.2,0.9)
CALL REALNO(TIMER,2,'ABUT','ABUT')
CALL INTNO ( IIPLOT , 'ABUT', 'ABUT' )
CALL ENDPL ( 0 )

IF ( TITLE .EQ. 9 ) THEN
  IF ( LINLOG .NE. 0 ) THEN
    CALL PAGE ( 11.0, 8.50 )
    XAXIS = 9.5
    YAXIS = 6.0
    CALL AREA2D ( XAXIS, YAXIS )
    LINLOG = 0
    GO TO 800
  ENDIF
ENDIF

IF ( TITLE .EQ. 1 ) THEN
  IF ( LINLOG .NE. 0 ) THEN
    CALL PAGE ( 11.0, 8.50 )
    XAXIS = 9.5
    YAXIS = 6.0
    CALL AREA2D ( XAXIS, YAXIS )
    LINLOG = 0
    GO TO 50
  ENDIF
  LINLOG = 1
ENDIF

RETURN
END

SUBROUTINE PLOTXV ( IL, IU, VL, VU, MARK, IMESS )

INCLUDE ( MPARAM )
INCLUDE ( MPTCL )
INCLUDE ( MCNTRL )

REAL XAXIS, YAXIS, XORIG, XMAX, YORIG, YMAX, TIMR, TIMR2

C Set velocity range if necessary.

IF ( MARK .EQ. 1 ) THEN
  IF ( VL .GE. VU ) THEN
    DO 10 I = IL, IU
      VL = MIN ( VL, VX(I) )
    10  CONTINUE
    IF ( ABS(VL) .EQ. ABS(VU) ) THEN
      IF ( VU .EQ. DZERO ) THEN
        VU = 10.0D0
      ELSE
        VU = 1.5D0 * ABS ( VU )
      ENDIF
    ELSE
      VU = 1.5D0 * ABS ( VU )
    ENDIF
    ENDIF
  ENDIF
  IF ( VL .GE. 0.25 ) THEN
    YORIG = -1.0
4042         YMAX = 1.0
4043         ELSE
4044         YORIG = VL
4045         YMAX = VU
4046         ENDF
4047         CALL YAXANG(0.)
4048         CALL YNAME(\'X VELOCITY\$,100)
4049         CALL HEADIN(\'PHASE SPACE - VX VS. X\$,100,2,1)
4050         IJPLT = IJPLT + 1
4051         WRITE(*,FMT=('(2X,\'PLOT \',I3,\'- PHASE SPACE - \'\',
4052                         \'VX vs. X\')')IJPLT
4053      ELSE
4054      IF (VL.GE.VU) THEN
4055         DO 20 I = IL,ILU
4056         VL = MIN(VL,VY(I))
4057         VU = MAX(VU,VY(I))
4058      20 CONTINUE
4059      IF (ABS(VL).LE.ABS(VU)) THEN
4060         IF (VU.EQ.DZERO) THEN
4061         VU = 10.000
4062         ELSE
4063         VU = 1.500 * ABS(VU)
4064      ENDIF
4065      ENDIF
4066      ENDIF
4067      VU = MAX(VU,ABS(VL))
4068      VL = -VU
4069      IF (VL.GE.0.25) THEN
4070         YORIG = -1.0
4071         YMAX = 1.0
4072      ELSE
4073         YORIG = VL
4074         YMAX = VU
4075      ENDIF
4076      CALL YAXANG(0.)
4077      CALL YNAME(\'Y VELOCITY\$,100)
4078      CALL HEADIN(\'PHASE SPACE - VY VS. X\$,100,2,1)
4079      IJPLT = IJPLT + 1
4080      WRITE(*,FMT=('(2X,\'PLOT \',I3,\'- PHASE SPACE - \'\',
4081                         \'VY vs. X\')')IJPLT
4082      ENDIF
4083      C
4084      C CALL GRAF(XORIG,\'SCALE\',XMAX,YORIG,\'SCALE\',YMAX)
4085      C---------------------------------------------------------------------
4086      C If No. particles < 2000 plot pluses, for best visibility.
4087      C---------------------------------------------------------------------
4088      IF (NPTS.LE.2000) THEN
4089      IF (MARK.EQ.1) THEN
4090         DO 30 I = IL,ILU
4091         ZX(I) = X(I)
4092         ZV(I) = VX(I)
4093      30 CONTINUE
4094      ELSE
4095         DO 40 I = IL,ILU
4096         ZX(I) = X(I)
4097         ZV(I) = VY(I)
4098      40 CONTINUE
4099      ENDIF
4100      IMARK = -1
4101      CALL MARKER(3)
4102      CALL CURVE(ZX, ZV, ILU, IMARK)
4103      ELSE
4104      C---------------------------------------------------------------------
4105      C If more than 13000 particles, plot every 3rd, or 5th....
4106      C---------------------------------------------------------------------
4107      INT = 2*(NPTS/13000) + 3
4108      J = 1
4109      IF (MARK.EQ.1) THEN
4110         DO 50 I = IL,ILU,INT
4111         ZX(J) = X(I)
4112   ZV(J) = VX(I)
4113     J = J+1
4114  50  CONTINUE
4115   ELSE
4116   DD 60  I = IL,IU,INT
4117   ZX(J) = X(I)
4118   ZV(J) = VY(I)
4119     J = J+1
4120  60  CONTINUE
4121   ENDIF
4122   II = IU/INT + 1
4123   IMARK = -1
4124   CALL MARKER(-1)
4125   CALL CURVE ( ZX, ZV, II, IMARK )
4126   ENDIF
4127   C
4128   TIM = TIME + 0.5*DT
4129   TIMR = TIM
4130   TIMR2 = TIME
4131   C
4132   IF ( IMESS .EQ. 1 ) THEN
4133   CALL MESSAG ('ELECTRONS$', 100, 7.2, 1.2 )
4134   ELSE
4135   IF ( IMESS .EQ. 2 ) THEN
4136   CALL MESSAG ('SPECIES 2$', 100, 7.2, 1.2 )
4137   ELSE
4138   CALL MESSAG ('BOTH SPECIES$', 100, 7.2, 1.2 )
4139   ENDIF
4140   ENDIF
4141   C
4142   CALL MESSAG ('RUN TIME : $', 100, 7.2, 0.9 )
4143   CALL REALNO ( TIMR, 2, 'ABUT', 'ABUT' )
4144   CALL MESSAG ('DATE : $', 100, 7.2, 0.6 )
4145   CALL INTNO ( IDATE, 'ABUT', 'ABUT' )
4146   CALL MESSAG ('RUN NO. : $', 100, 7.2, 0.3 )
4147   CALL INTNO ( ITIME, 'ABUT', 'ABUT' )
4148   CALL MESSAG ('POSITIONS ARE AT TIME $', 100, 2.3, 0.3 )
4149   CALL REALNO ( TIMR2, 2, 'ABUT', 'ABUT' )
4150   IIPLOT = IIPLOT + 1
4151   CALL MESSAG ('PLOT NO. : $', 100, 8.0, -0.7 )
4152   CALL REALNO ( IIPLOT, 'ABUT', 'ABUT' )
4153   CALL ENDPL ( 0 )
4154   C
4155   RETURN
4156  END
5000   C==============================================
5001   C
5002   SUBROUTINE PLTXY ( IL, IU, VMU, IMESS )
5003   C
5004   C==============================================
5005   C Plot VX-VY phase-space.
5006   C
5007   INCLUDE ( MPARAM )
5008   INCLUDE ( MPTCL )
5009   INCLUDE ( MCNTRL )
5010   C
5011   REAL XAXIS, YAXIS, XORIG, XMAX, YORIG, YMAX, TIMR
5012   C
5013   IF ( IL .GT. IU ) RETURN
5014   CALL PAGE ( 11.0, 8.50 )
5015   XAXIS = 9.5
5016   YAXIS = 6.0
5017   CALL AREA2D ( XAXIS, YAXIS )
5018   NPTS = IU-IL+1
5019   C==============================================
5020   C Set velocity range if necessary.
5021   C==============================================
5022   IF ( VMU .EQ. DZERO ) THEN
5023     VL = VX(IL)
5024     VU = VL
DO I = IL, IU
    VL = MIN ( VL, VX(I), VY(I) )
    VU = MAX ( VU, VX(I), VY(I) )
    CONTINUE
    VMU = MAX ( ABS(VL), ABS(VU) )
ENDIF

C
CALL XNAME ( 'X VELOCITY$', 100 )
CALL YNAME ( 'Y VELOCITY$', 100 )
CALL HEADIN ( 'VELOCITY SPACE - VX VS. VY$', 100, 2., 1 )
IJPLOT = IJPLOT + 1
WRITE(*, FMT='(2X, "PLOT ",I3." - VELOCITY SPACE - ", VX VS. VY")') IJPLOT
IF ( VMU .NE. DZERO ) GO TO 4
VMU = 10.00D0
CONTINUE
4
XORIG = -VMU
XMAX = VMU
YORIG = -VMU
YMAX = VMU
CALL YAXANG ( 0. )
CALL GRAF ( XORIG, 'SCALE', XMAX, YORIG, 'SCALE', YMAX )
C
C If < than 2000 particles, plot pluses for best visibility.
C
IF ( NPTS .GT. 2000 ) GO TO 31
C
DO 3 I = IL, IU
    ZX(I) = VX(I)
    ZV(I) = VY(I)
3 CONTINUE
IMARK = -1
CALL CURVE ( ZX, ZV, IU, IMARK )
GO TO 33
C
C If more than 13000 particles, plot every 3rd, or 5th...
C
31
INT = 2*( NPTS/13000 ) + 1
J = 1
C
DO 32 I = IL, IU, INT
    ZX(J) = VX(I)
    ZV(J) = VY(I)
32
J = J+1
II = IU/INT + 1
IMARK = -1
CALL CURVE ( ZX, ZV, II, IMARK )
33 CONTINUE
TIM = TIME + 0.5D0*DT
TIMR = TIM
C
IF ( IMESS .EQ. 1 ) THEN
    CALL MESSAG ( 'ELECTRONS$', 100, 7.2, 1.2 )
ELSE
    IF ( IMESS .EQ. 2 ) THEN
        CALL MESSAG ( 'SPECIES 2$', 100, 7.2, 1.2 )
    ELSE
        CALL MESSAG ( 'BOTH SPECIES 2$', 100, 7.2, 1.2 )
    ENDIF
ENDIF
C
CALL MESSAG ( 'RUN TIME : $', 100, 7.2, 0.9 )
CALL REALNO ( TIMR, 2, 'ABUT', 'ABUT' )
CALL MESSAG ( 'DATE : $', 100, 7.2, 0.6 )
CALL INTNO ( ITIME, 'ABUT', 'ABUT' )
SUBROUTINE PLTF0V ( IL, IU, VL, VU, NBINS, TITLE, IIFOV, TEMP1, TEMP2)

Plot the distribution function \( f(v) \). If IFOV is negative, make both semi-log and cartesian plots.

C------------------------------------------------------------
C INCLUDE ( MPARAM )
C INCLUDE ( MPTCL )
C INCLUDE ( MCNTRL )
C INCLUDE ( MTHERM )
REAL XAXIS, YAXIS, XORIG, XMAX, YORIG, YMAX, TIMR,
XSTEP, YCYCLE, TVBIN, TFBIN, RTMP1, RTMP2
PARAMETER ( MAXBIN = 100 )
COMMON /SCRATC/ VBIN(MAXBIN), BIN (MAXBIN),
INT DIMENSION VV(IU)
IFOV = IIFOV
RTEMP1 = TEMP1
RTEMP2 = TEMP2
C------------------------------------------------------------
C NBINS must not exceed dimension of these arrays.
C------------------------------------------------------------
C-------------------------------------------------------------
NBINS = MIN ( NBINS, MAXBIN )
IF ( TITLE .EQ. 1 .OR. TITLE .EQ. 2 .OR.
. TITLE .Eq. 3 ) THEN
VL = DZERO
VU = L
ELSE
ENDIF
C Find velocity range.
C-------------------------------------------------------------
IF ( VL .EQ. DZERO .AND. VU .EQ. DZERO ) THEN
VL = VV(IL)
VU = VL
DO 1 I=IL,IU
VL = MIN ( VL, VV(I) )
VU = MAX ( VU, VV(I) )
1 CONTINUE
VU = MAX ( VU, ABS(VL) )
VL = -VU
ENDIF
ENDIF
IF ( VL .EQ. VU .OR. NBINS .LE. 1 ) RETURN
CALL PAGE ( 11.0, 8.50 )
XAXIS = 9.5
YAXIS = 6.0
CALL AREA2D ( XAXIS, YAXIS )
C-------------------------------------------------------------
C Assign bin midpoints and initialize bins to zero.
C-------------------------------------------------------------
DO 10 J = 1,NBINS
BIN(J) = DZERO
VBIN(J) = VL + (VU-VL)*(FLOAT(J)-0.5)/FLOAT(NBINS)
10 CONTINUE
C Assign VV's to bins with linear weighting.
C-------------------------------------------------------------
DO 20 I = IL,IU
XXX = NBINS * ( VV(I)-VL ) / ( VU-VL )
J = XXX + 1
IF ( JJ .GT. NBINS .OR. JJ .LT. 1 ) GO TO 20

BIN(JJ) = BIN(JJ) + 1.0D0

CONTINUE

CALL CARTMM ( NBINS, RMIN, RMAX, BIN )

CONTINUE

C

IF ( TITLE .EQ. 1 ) THEN
 CALL XNAME ( 'X - POSITIONS', 100 )
 CALL YNAME ( 'NO. OF PARTICLES', 100 )
 CALL HEADIN ( 'ELECTRON DISTRIBUTION IN SPACE', 100, 2., 1 )
 IJPLOT = IJPLOT + 1
 WRITE(*,FMT='(2X,' PLOT ',I3,': 'ELECTRON DISTRIBUTION',
 ' IN SPACE.')') IJPLOT
 ELSE
...

ENDIF

C

IF ( TITLE .EQ. 2 ) THEN
 CALL XNAME ( 'X - POSITIONS', 100 )
 CALL YNAME ( 'NO. OF PARTICLES', 100 )
 CALL HEADIN ( 'SPECIES 2 - DISTRIBUTION IN SPACE', 100, 2., 1 )
 IJPLOT = IJPLOT + 1
 WRITE(*,FMT='(2X,' PLOT ',I3,': 'SPECIES 2 DISTRIBUTION',
 ' IN SPACE.')') IJPLOT
 ELSE
...

ENDIF

C

IF ( TITLE .EQ. 3 ) THEN
 CALL XNAME ( 'X - POSITIONS', 100 )
 CALL YNAME ( 'NO. OF PARTICLES', 100 )
 CALL HEADIN ( 'ALL PARTICLES - DISTRIBUTION IN SPACE', 100, 2., 1 )
 IJPLOT = IJPLOT + 1
 WRITE(*,FMT='(2X,' PLOT ',I3,': 'ALL PARTICLES DISTRIBUTION',
 ' IN SPACE.')') IJPLOT
 ELSE
...

ENDIF

C

IF ( TITLE .EQ. 4 ) THEN
 CALL XNAME ( 'VELOCITY', 100 )
 CALL YNAME ( 'DISTRIBUTION FN. - F(VX)', 100 )
 CALL HEADIN ( 'ELECTRONS: F(VX) - VELOCITY DISTRIBUTION', 100, 2., 1 )
 IJPLOT = IJPLOT + 1
 WRITE(*,FMT='(2X,' PLOT ',I3,': 'ELECTRONS',
 ' F(VX).')') IJPLOT
 ELSE
...

ENDIF

C

IF ( TITLE .EQ. 5 ) THEN
 CALL XNAME ( 'VELOCITY', 100 )
 CALL YNAME ( 'DISTRIBUTION FN. - F(VY)', 100 )
 CALL HEADIN ( 'ELECTRONS: F(VY) - VELOCITY DISTRIBUTION', 100, 2., 1 )
 IJPLOT = IJPLOT + 1
 WRITE(*,FMT='(2X,' PLOT ',I3,': 'ELECTRONS',
 ' F(VY).')') IJPLOT
 ELSE
...

ENDIF

C

IF ( TITLE .EQ. 6 ) THEN
 CALL XNAME ( 'VELOCITY', 100 )
 CALL YNAME ( 'DISTRIBUTION FN. - F(VX)', 100 )
 CALL HEADIN ( 'SPECIES 2: F(VX) - VELOCITY DISTRIBUTION', 100, 2., 1 )
 IJPLOT = IJPLOT + 1
 WRITE(*,FMT='(2X,' PLOT ',I3,': 'SPECIES 2',
 ' F(VX).')') IJPLOT
 ELSE
...

ENDIF

C

IF ( TITLE .EQ. 7 ) THEN
 CALL XNAME ( 'VELOCITY', 100 )
 CALL YNAME ( 'DISTRIBUTION FN. - F(VY)', 100 )
 CALL HEADIN ( 'SPECIES 2: F(VY) - VELOCITY DISTRIBUTION', 100, 2., 1 )
 IJPLOT = IJPLOT + 1
 WRITE(*,FMT='(2X,' PLOT ',I3,': 'SPECIES 2',
 ' F(VY).')') IJPLOT
 ELSE
...

ENDIF
IF ( TITLE .EQ. 8 ) THEN
CALL XNAME ('VELOCITY*', 100)
CALL YNAME ('DISTRIBUTION FN. - F(VX)$', 100)
CALL HEADIN ('F(VX) - ALL PARTICLES$', 100, 2, 1)
IJPLOT = IJPLOT + 1
WRITE(*,FMT='(2X,'"PLOT ",IB," - ALL PARTICLES ",F(VX)$')), IJPLOT
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
 предоставлен текст документа в чистом текстовом представлении.
CALL INTNO ( ITIME, 'ABUT', 'ABUT' )
IIPLOT = IIPLOT + 1
CALL MESSAG ('PLOT NO. : $', 100, 8.0,-0.7 )
CALL INTNO ( IIPLOT , 'ABUT', 'ABUT' )
CALL ENDPL ( O )

IF ( IFOV .LT. ZERO ) THEN
CALL PAGE ( 11.0, 8.50 )
XAXIS = 9.5
YAXIS = 6.0
CALL AREA2D ( XAXIS, YAXIS )
IFOV = ABS(IFOV)
GO TO 100
ENDIF
RETURN
END

C===========================================================
SUBROUTINE PLOTF ( F, TITLE )
C  Plot fields or electron/ion temperatures.
INCLUDE ( MPARAM )
INCLUDE ( MFIELD )
INCLUDE ( MCNTRL )
INTEGER TITLE
DOUBLE PRECISION XJ(NG1M), F(1)
REAL XAXIS, YAXIS, XORIG, XMAX, YORIG, YMAX, TIMER.
TXJ(NG1M), TF(NG1M)

NG1 = NG + 1
DATA Xd(2) / O.ODO /
IF ( XJ(2) .NE. DX ) THEN
DO 1 J = 1,NG1
XJ(J) = (J-1)*DX
1 CONTINUE
ENDIF

CALL CARTMM ( NG1, RMIN, RMAX, F )
IF ( JFLAG .EQ. O ) THEN
C If JFLAG .EQ. O then just determine max & min values for
C later use when setting scales via JFLAG .LT. O
IF ( TITLE .EQ. 1 ) THEN
CDMIN = MIN ( CDMIN, RMIN )
CDMAX = MAX ( CDMAX, RMAX )
ELSE
IF ( TITLE .EQ. 2 ) THEN
SDMIN = MIN ( SDMIN, RMIN )
SDMAX = MAX ( SDMAX, RMAX )
ELSE
IF ( TITLE .EQ. 3 ) THEN
EPMIN = MIN ( EPMIN, RMIN )
EPMAX = MAX ( EPMAX, RMAX )
ELSE
IF ( TITLE .EQ. 4 ) THEN
EFMIN = MIN ( EFMIN, RMIN )
EFMAX = MAX ( EFMAX, RMAX )
ELSE
IF ( TITLE .EQ. 5 ) THEN
ETMIN = MIN ( ETMIN, RMIN )
ETMAX = MAX ( ETMAX, RMAX )
ELSE
IF ( TITLE .EQ. 6 ) THEN
ITMIN = MIN ( ITMIN, RMIN )
ITMAX = MAX ( ITMAX, RMAX )
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDING
7054    ENDIF
7055    ENDIF
7056    ENDIF
7057    RETURN
7058    ENDIF
7059    CALL PAGE ( 11.0, 8.50 )
7060    XAXIS = 9.5
7061    YAXIS = 6.0
7062    CALL AREA2D ( XAXIS, YAXIS )
7063
7064    C
7065    IF ( TITLE .EQ. 1 ) THEN
7066        CALL XNAME ( 'X POSITIONS', 100 )
7067        CALL YNAME ( 'CHARGE DENSITY$', 100 )
7068        CALL HEADIN ( 'CHARGE DENSITY$', 100, 2..1 )
7069        IJPLOT = IJPLOT + 1
7070        WRITE(*,FMT='(2X,' 'PLOT ',I3,' - CHARGE',
7071               ' DENSITY.')') IJPLOT
7072    ELSE
7073        IF ( TITLE .EQ. 2 ) THEN
7074            CALL XNAME ( 'X POSITION*', 100 )
7075            CALL YNAME ( 'SMOOTHED DENSITY*', 100 )
7076            CALL HEADIN ( 'SMOOTHED DENSITY*', 100, 2..1 )
7077            IJPLOT = IJPLOT + 1
7078            WRITE(*,FMT='(2X,' 'PLOT ',I3,' - SMOOTHED CHARGE',
7079                   ' DENSITY.')') IJPLOT
7080        ELSE
7081            IF ( TITLE .EQ. 3 ) THEN
7082                CALL XNAME ( 'X POSITIONS', 100 )
7083                CALL YNAME ( 'ELECTROSTATIC POTENTIALS$', 100 )
7084                CALL HEADIN ( 'ELECTROSTATIC POTENTIALS$', 100, 2..1 )
7085                IJPLOT = IJPLOT + 1
7086                WRITE(*,FMT='(2X,' 'PLOT ',I3,' - ELECTROSTATIC',
7087                       ' POTENTIAL.')') IJPLOT
7088            ELSE
7089                IF ( TITLE .EQ. 4 ) THEN
7090                    CALL XNAME ( 'X POSITIONS', 100 )
7091                    CALL YNAME ( 'ELECTRIC FIELDS$', 100 )
7092                    CALL HEADIN ( 'ELECTRIC FIELDS$', 100, 2..1 )
7093                    IJPLOT = IJPLOT + 1
7094                    WRITE(*,FMT='(2X,' 'PLOT ',I3,' - ELECTRIC',
7095                           ' FIELD.')') IJPLOT
7096                ELSE
7097                    IF ( TITLE .EQ. 5 ) THEN
7098                        CALL XNAME ( 'X POSITIONS', 100 )
7099                        CALL YNAME ( 'ELECTRON TEMPERATURES$', 100 )
7100                        CALL HEADIN ( 'RELATIVE ELECTRON TEMP.(EV)$', 100, 2..1 )
7101                        IJPLOT = IJPLOT + 1
7102                        WRITE(*,FMT='(2X,' 'PLOT ',I3,' - REL. ELECTRON',
7103                               ' TEMP.').'') IJPLOT
7104                    ELSE
7105                        IF ( TITLE .EQ. 6 ) THEN
7106                            CALL XNAME ( 'X POSITIONS', 100 )
7107                            CALL YNAME ( 'ION TEMPERATURES$', 100 )
7108                            CALL HEADIN ( 'ION TEMPERATURE (EV)$', 100, 2..1 )
7109                            IJPLOT = IJPLOT + 1
7110                            WRITE(*,FMT='(2X,' 'PLOT ',I3,' - REL. ION',
7111                                 ' TEMP.').'') IJPLOT
7112                        ELSE
7113                            ENDIF
7114                            ENDIF
7115                            ENDIF
7116                            ENDIF
7117                            ENDIF
7118                            ENDIF
7119                            C
7120                            IF ( RMIN .EQ. RMAX ) THEN
7121                                IF ( RMIN .EQ. DZERO ) THEN
7122                                    RMAX = 10.000
7123                                ELSE
7124                                    RMAX = 1.500 * ABS ( RMAX )
C RMAX = MAX( RMAX, ABS(RMIN) )
RMIN = -RMAX
C
IF ( JFLAG .LT. ZERO ) THEN
XORIG = XXORIG
XMAX = XXMAX
YORIG = MIN( YYORIG, RMIN )
YMAX = MAX( YYMAX, RMAX )
ELSE
XORIG = 0.0
XMAX = L
YORIG = RMIN
YMAX = RMAX
ENDIF
C
IMARK = 0
CALL YAXANG ( 0. )
C
DO 7 I=1,NG1
    TXJ(I) = XJ(I)
    TF(I) = F(I)
7   TF(I) = F(I)
C
TIMBR = TIME
CALL GRAF ( XORIG, 'SCALE', XMAX, YORIG, 'SCALE', YMAX )
CALL CURVE ( TXd, TF, NG1, IMARK )
CALL MESSAG ('RUN TIME : $', 100, 7.2, 0.9 )
CALL REALNO ( TIMER, 2, 'ABUT', 'ABUT' )
CALL MESSAG ('RUN NO. : $', 100, 7.2, 0.3 )
CALL INTNO ( ITIME, 'ABUT', 'ABUT' )
IIPLOT = IIPLOT + 1
CALL MESSAG ('PLOT NO. : $', 100, 8.0,-0.7 )
CALL INTNO ( IIPLOT, 'ABUT', 'ABUT' )
CALL ENDPL ( 0 )
C
RETURN
END
C=============================================================================
C SUBROUTINE CARTMM ( N, RL, RU, REC )
C=============================================================================
C Calculate and return MIN & MAX components of vector REC.
C=============================================================================
C IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
DIMENSION REC(N)
C
RU = REC(1)
RL = RU
C
IF ( N .EQ. 1 ) RETURN
C
DO 10 I = 2,N
    RU = MAX ( RU, REC(I) )
    RL = MIN ( RL, REC(I) )
10 CONTINUE
C
RETURN
END
SUBROUTINE PSPEC
C Plot energy spectrum.

INCLUDE ( MPARAM )
INCLUDE ( MFIELD )
INCLUDE ( MCNTRL )

DOUBLE PRECISION XJ(NG1M)
REAL XAXIS, YAXIS, XORIG, XMAX, YORIG, YMAX, TIMER,
      TXJ(NG1M), TF(NG1M)

NG2P = NG/2 + 1

CALL CARTRMM ( NG2P, RL, RU, EK )
CALL PAGE ( 11.0, 8.50 )
XAXIS = 9.5
YAXIS = 6.0
CALL AREA2D ( XAXIS, YAXIS )

CALL XNAME ( 'MODE NUMBERS', 100 )
CALL YNAME ( 'MODE ENERGYS', 100 )
CALL HEADIN ( 'ENERGY SPECTRUMS', 100, 2..1 )
IJPLOT = IIPLOT + 1
WRITE(*,FMT='(2X, "PLOT ",I3," - ENERGY",SPECTRUM.")') IJPLOT

XORIG = 0.0
XMAX = NG2P
YORIG = 0.0
YMAX = 1.0
IMARK = 0
CALL YAXANG ( 0.0 )

DO 7 I=1,NG2P
   TXJ(I) = 1-1
   TF(I) = (EK(I)/RU)
7

TIMER = TIME
CALL GRAF ( XORIG, 'SCALE', XAXIS, XMAX, YORIG, 'SCALE', YAXIS )
CALL CURVE ( TXJ, TF, NG2P, IMARK )
CALL MESSAG ('RUN TIME : $', 100, 7.2, 0.9 )
CALL REALNO ( TIMER, 2, 'ABUT', 'ABUT' )
CALL MESSAG ('DATE $', 100, 7.2, 0.6 )
CALL INTNO ( 'DATE', 'ABUT', 'ABUT' )
CALL MESSAG ('RUN NO. $', 100, 7.2, 0.3 )
CALL INTNO ( 'RUN NO.', 'ABUT', 'ABUT' )
IIPLOT = IIPLOT + 1
CALL MESSAG ('PLOT NO. : $', 100, 8.0.-0.7 )
CALL INTNO ( 'PLOT NO.', 'ABUT', 'ABUT' )
CALL ENDP ( 0 )

RETURN
END