PDW1 USER'S MANUAL

by

W. S. Lawson

Memorandum No. UCB/ERL M84/37

27 April 1984

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

PDW1 USER'S MANUAL

by

W. S. Lawson

Memorandum No. UCB/ERL M84/37

27 April 1984

PDW1 User's Manual

Wm. S. Lawson

ERL, Cory Hall, U. C. Berkeley, 94720

January 1984

Abstract

PDW1 is a one dimensional, magnetized, electrostatic particle simulation with one, two or three velocity coordinates, and non-periodic boundary conditions. It is set up to inject Maxwellian distributions with arbitrary drift velocities and cut-off velocities, but is easily modifiable to suit other distributions. This document is both a user's manual, and a report on the principles of operation of the code. Inquiries should be directed to Prof. C. K. Birdsall at the address above.

I Introduction

This report is intended as a guide to the use and modification of the simulation code PDW1. It includes detailed descriptions of the methods and algorithms used in the code. Some familiarity with numerical methods on the part of the reader is assumed. The report is broken up into six Sections and five Appendices. Section I is, of course, the Introduction, which describes the contents of this write-up, and some general programming concepts used in the design of the code. Section II is a guide to the mechanics of compiling and running the code. Section III describes the various parameters used to completely specify a given problem. Section IV will be helpful to anyone who wants to make minor modifications to the code (which will be almost everyone who uses the code). Section V is an exhaustive list of virtually all the variables in the program. Section VI gives in-depth descriptions of each of the major subroutines, including the algorithms, and descriptions of any subroutines called by the routines. The final five Sections are Appendices. The first Appendix contains flow charts for

all the major routines, the second contains a description of the workings of the ZED interface (written by Niels Otani), the third contains some integrals which will make parameter selection easier, the fourth contains documentation on a minor but useful modification of PDW1 named PDWMAX, and the fifth contains some examples which should illustrate some of the problems PDW1 is capable of solving.

PDW1 is an acronym for Plasma Device Workshop. The code was written during, and for use in, a seminar workshop at the University of California at Berkeley on axially bounded plasma systems during the 1983 spring quarter. Since the code is in large measure a product of the information exchanged during this workshop, it was deemed appropriate to name it after the workshop.

The purpose of PDW1 is to solve the electrostatic magnetized problems with non-periodic boundary conditions. Inherent in non-periodic boundary conditions is an external circuit of some sort, and working this external circuit into the code was perhaps the major step forward. Figure 1 diagrams the assumed set-up. Specifically, the equations to be solved are:

$$\frac{d^2x_i}{dt^2} = \frac{q_i}{m_i}\mathrm{E}(x_i,t)$$

where x_i, q_i and m_i are the position, and charge and mass sheet densities of the i'th particle respectively,

$$\frac{\partial \mathrm{E}}{\partial x} = \frac{1}{\epsilon_0}\sum_i q_i S_i(x - x_i)$$

where S_i is the shape factor for the i'th particle (ϵ_0 is the dielectric constant in a vacuum),

$$\hat{L}\frac{dI}{dt} + \hat{R}I + \frac{Q}{\hat{C}} + V_0(t) = V(t)$$

where $\hat{L}$, $\hat{R}$ and $\hat{C}$ are the inductance, resistance, and capacitance of the external circuit (this series circuit is not the most general circuit one might want, but it is a good start), V_0 is a source voltage, V is the voltage across the plasma region, and I and Q are the current and charge

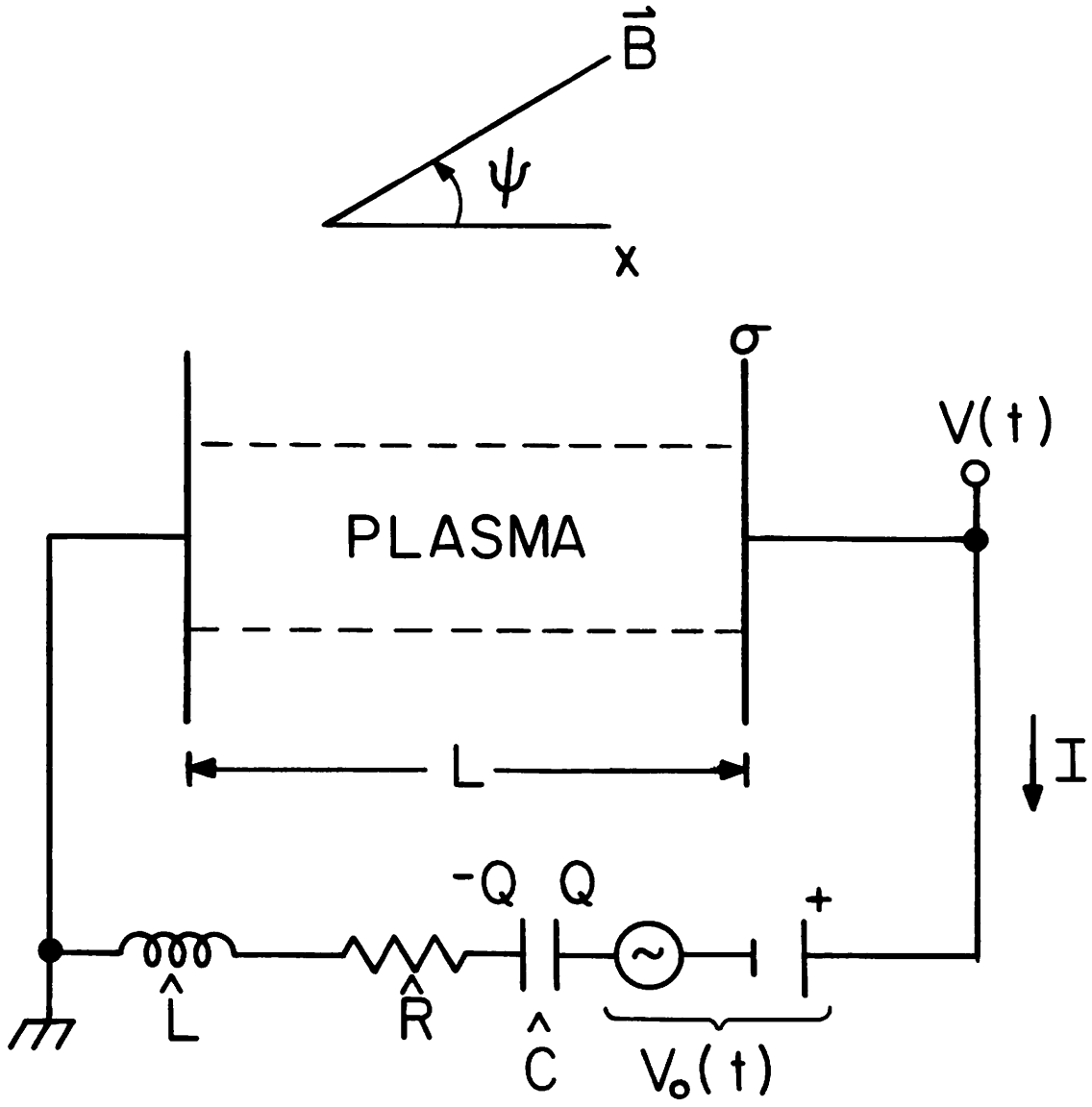


Figure 1. Diagram of circuit set-up

associated with the external circuit (Q is the charge on the capacitor) such that $I = dQ/dt$ and finally

$$\frac{d\sigma}{dt} = \sum_i q_i v_i \delta(x_i - L) - \frac{I}{A}$$

where σ is the charge on the plate at the right hand boundary of the plasma region, A is the area of that plate v_i is the velocity of the i'th particle, and L is the length of the plasma region. The factor of area of the end plate is necessary because σ is a surface charge density rather than an honest charge. This last equation determines the boundary condition for the field solve, since

$$\mathrm{E}(L) = -\frac{\sigma}{\epsilon_0}.$$

All these equations have initial conditions, which can be specified.

In addition to all these equations, the injection and initial loading of particles must be specified (see Figure 2). PDW1 is currently set up to load and inject partial Maxwellian distributions, but the loading scheme is capable of handling almost any distribution. The Maxwellian distributions injected at each end (independently) can be offset (representing a drift velocity) and cut off at any velocity greater than zero (or less than zero for the right hand wall) to represent a distribution which has already been accelerated (see Figure 3). The input parameters can become confusing because of this amount of freedom, so Appendix C contains all the integrals one must understand in order to choose the proper parameters. For many problems, a full Maxwellian distribution is desired, and the full battery of PDW1 input parameters can be tedious. The modification named PDWMAX was written for this purpose, and the differences between PDWMAX and PDW1 are described in Appendix D.

PDW1 was designed with two major programming goals: modularity and transportability. Both of these goals were occasionally compromised of necessity, but by and large they were achieved. Modularity is the breaking up of a large problem into sub-problems, each of which

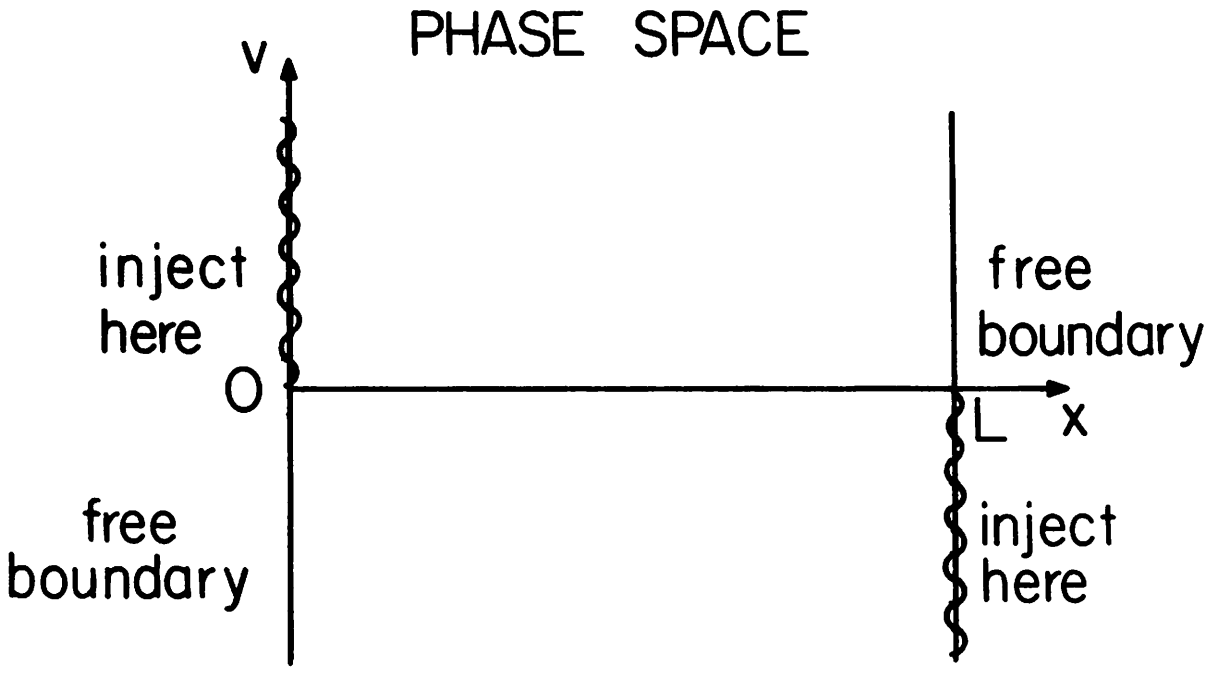


Figure 2. Phase space boundaries

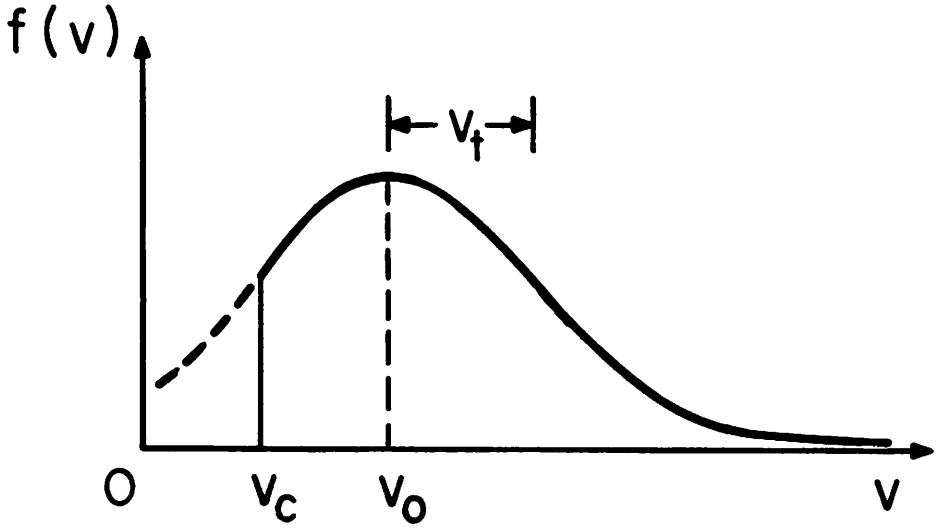


Figure 3. Injected distribution

needs to know little or nothing about the internal workings of the other sub-problems. Modular programming reduces errors, and simplifies debugging immensely. It is some measure of the success of this concept that the code was written, debugged and internally documented within a three month period. The separation of sub-problems also helps immensely in subsequent modification for slightly different problems. Transportability, in this case, means sticking as much as possible to standard FORTRAN IV, or at least to common extensions of it. This makes it possible (though perhaps not easy) to move the code to a different environment and a completely different machine, and have it work after a minimum of (straight-forward) modification. This goal was compromised in many places to simplify the input, improve the readability, and improve the speed of the code on the CRAY-1. Sometimes drastic measures were used to preserve transportability (the ZED interface was made transportable by making it easy to remove). There are also some subtle incompatibilities between systems; for example, the CRAY-1 presently preserves subroutine variables between subroutine calls, which is not true on all machines. In writing PDW1 much care was taken to use the safe alternative in such cases, and hopefully there were not too many oversights.

PDW1 owes a great deal to the earlier code ES1 and its author, Dr. Bruce Langdon. The book *Plasma Physics Via Computer Simulation* by Birdsall and Langdon is an invaluable reference in understanding how such particle codes (and ES1 in particular) work.

II Compiling and Running

All files associated with PDW1 are contained in the LIB library file PDWLIB. PDWLIB is available to users of the National Magnetic Fusion Energy Computer Center (NMFECC) on FILEM under user 1204, directory PDW. Users at other installations will need to obtain a copy either from someone on the NMFECC system, or a source listing from Prof. Birdsall. The files in PDWLIB are: PDW1 (the FORTRAN source code), BPDW (the binary library of necessary subroutines), PDWCOMP (a COSMOS control file for compiling and loading PDW1), PDWSRC (another LIB file, containing source listings for all the routines in BPDW), NFIELDS and NMOVE (non-vectorized versions of the field solve and mover routines in PDW1), and several sample input files (IPD1, IPD2, IPD3 and IPI). Of the routines in BPDW, five — LOCVARP, OOPS, LOCVAR, SETFILEZ and REPREFIX — are used by the ZED interface, and will be described in Appendix B. GAUSIN, REVERS and TIMER will be described in Section VI.

To run PDW1, it must be compiled, and loaded with BPDW. BPDW also directs the loader to the public graphics library TV80LIB, which contains all the graphics routines. NMFECC users can use PDW1 simply by pulling out the necessary files from PDWLIB with LIB, changing the parameters in PDW1 to the desired values, and running cosmos on PDWCOMP. The resulting executable file will be named XPDW1. On other installations, it will be necessary to modify the graphics and input/output routines of PDW1.

The input parameters are stored in a file separate from the main program so that the program need not be recompiled each run. Several sample input files are included in PDWLIB.

A typical input file, on the MFE systems, might look as follows:

```
box bnn run title
length=4.  nt=2000 zed=.true.  $end (species independent parameters)
j01=20.  empty=.false.  $end (parameters for species 1)
j01=10.  qm=.25 inject=.false.  $end (parameters for species 2)
2000 step from step 1 (ZED commands)
every 1 step
ese 1 1
```

The input file name should begin with an "i" (e.g., itest), but when running the code, the execute line should omit this "i" (e.g., xpdw1 test). (This format of input is due to Dr. A. B. Langdon, and has proven to be a good one over many years of use of the code ES1.)

Non-NMFECC users will most likely need to write an input routine to suit their own installation and favorite input format.

III Parameters

There are two classes of parameters which specify a given problem. These are the compile-time parameters and the input or run-time parameters. The compile-time parameters are put into the code at compile time by a precompiler, and pertain to maximum dimensions for arrays and such. Each time they are changed, the code must be recompiled. The run-time parameters are read in by the code from the input file each time it is executed. Changing run-time parameters requires only changing the input file; the code need not be recompiled.

The compile time parameters are:

NSMAX — Maximum number of species. The number of species (NSP) must be less than or equal to this, or you will get a scolding.

NGMAX — Maximum number of grid *cells* (one less than the maximum number of grid *points*). It is simplest to make this a power of two, but not at all necessary.

NVMAX — Maximum number of velocity components. May be 1, 2 or 3, (corresponding to x, xy and xyz components) depending on the number of velocity components you need to keep. It generally does not pay to make this any larger than you will need, since it represents the number of storage locations that must be reserved for each particle's velocity.

NPMAX — Maximum number of particles per species. Each species is allotted this amount of storage space. (It is fairly simple to modify the code to reserve different amounts of space for different species.)

HYMAX — Maximum number of points which can be plotted on a history graph (well, actually one less than the maximum). This is set to 400 in the standard version, and any more than about 500 is beyond the resolution of most plotters.

The run-time parameters fall into two categories: species independent and species dependent. The species independent parameters are specified once in the input file, and the species dependent parameters are specified once for each species.

The species independent parameters are:

LENGTH — length of system (default 1.)

AREA — area of ends of system (default 1.)

EPS0 — dielectric constant (default 1.)

B — magnetic field (default 0 — ignored if NV=1)

PSI — angle between magnetic field and x-direction in degrees (default 0. — ignored if NV < 3, and assumed to be 90. if NV = 2)

NSP — number of species (default NSMAX)

NG — number of grid *cells* (default NGMAX)

NT — number of timesteps in run (default HYMAX)

NV — number of velocity components (default NVMAX)

DT — timestep increment (no default)

EXTR — external resistance (default 0.)

EXTL — external inductance (default 0.)

EXTC — external capacitance (default 10.**20 — note these three values for EXTR, EXTL and EXTC represent a short-circuit)

Q0 — initial charge on right hand side of capacitor (default 0.)

I0 — initial current flowing through external circuit from right to left in the diagram (i.e. I0 = dQ/dt where Q is the charge on the right hand side of the external capacitor — default 0.)

SIGMA0 — initial surface charge on the right hand end plate (default 0.)

RHOBACK — uniform background charge density (default 0.)

BACKJ — uniform background current density (represents background current traveling from left to right *within the simulation region* due to species whose dynamics are insignificant, e.g. a beam of infinite mass ions. With EXTC=0., BACKJ can be used to represent a constant current source of - BACKJ. — default 0.)

DCBIAS — constant potential applied to external circuit (in series) so as to raise the potential of the right hand end plate if DCBIAS is positive (default 0.)

ACBIAS — amplitude of sinusoidal potential applied to external circuit (similar to DCBIAS — default 0.)

W0 — frequency (radians/second) of applied AC bias (external bias = DCBIAS + ACBIAS* SIN(W0*T) — default 0.)

IPLOT — plot frequency variable. Every IPLOT timesteps, snapshot diagnostics are plotted (default 0 — if IPLOT=0, then no plots are made)

IOUT — alphanumeric output frequency variable (default 0 — the only diagnostic of this type in the standard code is the number of particles)

ISAV — every ISAV timesteps, history diagnostics are sampled (default 0)

IHIST — every ISAV*IHIST timesteps, history plots are made (default HYMAX)

IPACK — every IPACK timesteps, particles which have left the system are removed, and their charge is deposited on the plate they have entered (default 10)

ZED — logical flag for whether ZED output files are to be produced (default .FALSE., i.e. no ZED output)

The species dependent variables are:

QM — charge to mass ratio (default -1.)

J0L, J0R — injected current densities from left and right plates (default 0.)

V0L, V0R — velocities of center of Maxwellian distributions injected from left and right boundaries (default 0.)

VYL, VYR — drift velocities in y direction of injected Maxwellian (default 0.)

VZL, VZR — drift velocities in z direction (default 0.)

VCL, VCR — low speed cutoff velocities for injected distributions — see diagram (default 0.)

VTL, VTR — thermal velocities for injected Maxwellians, where $v_t{}^2 = kT/m$ (default 0.)

FLUXL, FLUXR — number of particles to be injected per unit time from left and right ends (default 0. — note that FLUXL, FLUXR, J0L and J0R are not all independent. Only three of them need be input, but if all four are input, FLUXR is ignored.)

EMPTY — flag for whether system is initially empty, or initially loaded with the injected distribution (default .TRUE.)

INJECT — flag for whether or not particles are actually to be injected or not (default .TRUE.)

IV Common Modifications — What You Need To Know

The most common modification to the code is likely to be in the diagnostics. It is strongly recommended that diagnostic variables which are not essential to the simulation itself be computed only in the diagnostic routine OUTPUT. This will exact the smallest penalty in run time, and keep the code modular. The routines GRAPH, LABEL and HLABEL were written to simplify the production of output plots. For complete information on how these should be used, see Section VI. It will probably be easiest to just emulate one of the existing diagnostics in adding your own, though.

Another common modification will undoubtedly be changes in the input parameters. Some modifications will be simple, and others will have to include modifications to the loader Section of the code, where the normalization of the distribution function is known. The greatest pitfall will be in normalization of the position and velocity variables; *all* velocity variables (V0, VT, VC, etc.) are made dimensionless by multiplying them with $\Delta t/\Delta x$. Position variables are normalized by division by Δx. It should only be necessary to study the START routine (and its subroutines SPECIES and LOAD) in order to modify the input parameters. (For an example of such a modification, see the Appendix on PDWMAX.)

V Variables

The major variables can be divided up into two classes: problem parameters, and dynamical variables. The problem parameters are changed only during the initialization, and are all included in one of two common blocks. The dynamical variables are passed as arguments for the sake of clarity, except for tally variables which would simply clutter up the argument lists.

Only variables representing positions and velocities (e.g. X and V) are normalized; all other variables are left unnormalized. X is normalized such that X=0 at the first grid point, X=1 at the second grid point, and so on. V is normalized such that a particle with $v = \Delta x/\Delta t$ will move one grid cell per timestep. *All* variables denoting positions and velocities are so normalized, and one must be careful when using these variables to compute other diagnostics. These normalizations are purely internal; the input parameters and output diagnostics should be in consistent units.

There are two main common blocks in PDW1. The first is called PARAM, and contains most of the problem parameters which are computed during initialization. The second is called HIST, and contains all the variables associated with the history diagnostics: plot frequency variables, tally variables, and history save arrays.

The variables in PARAM and their purposes are:

NG — number of grid cells (an input parameter)

NT — number of timesteps in run (an input parameter)

DX — distance between grid points (LENGTH/NG)

DT — time in a timestep (an input parameter)

NV — number of velocity components kept (an input parameter)

NLOC(NSMAX1) — array containing the storage boundaries between different particle species. NLOC(1) is always zero, and NLOC(I) represents the offset that must be added to J to get the index in the X and V arrays corresponding to the J'th particle of the I'th species (i.e. NLOC(I) = NLOC(I-1) + maximum number of particles for species I). If this is not clear, studying some of the loop limits in any routine which updates particles should be helpful. This storage arrangement allows for efficient storage of particles when the number of particles of one species is much greater than the number of particles in another, with only minor modification of the code.

LENGTH — physical length of system (input parameter)

AREA — area of end plates (input parameter)

EPS0 — dielectric constant (input parameter)

B — magnetic field strength (input parameter)

PSI — angle between magnetic field and X-direction in degrees (B is always in the X-Z plane) (input parameter — converted to radians internally)

NSP — number of species in simulation (input parameter)

RHOBACK — background charge density (input parameter)

BACKJ — background current density (input parameter)

EXTR — resistance of external circuit (input parameter)

EXTL — inductance of external circuit (input parameter)

EXTC — capacitance of external circuit (input parameter)

DCBIAS — constant bias voltage across external circuit (input parameter)

ACBIAS — sinusoidal bias voltage across external circuit (input parameter)

W0 — frequency of AC bias voltage in radians per unit time (input parameter)

QM(NSMAX) — charge to mass ratio for each species (input parameter)

Q(NSMAX) — charge per simulation particle

M(NSMAX) — mass per simulation particle

J0(2,NSMAX) — injected current densities from the left and right ends of the simulation region for each species

V0(2,NSMAX) — drift velocities in the x direction at the left and right ends for each species (normalized)

VDY(2,NSMAX) — drift velocities in the y direction at the left and right ends for each species (normalized)

VDZ(2,NSMAX) — drift velocities in the z direction at the left and right ends for each species (normalized)

VT(2,NSMAX) — thermal velocities at the left and right ends for each species (normalized)

VC(2,NSMAX) — low speed cut-off for injected distributions at the left and right ends for each species (normalized)

ENTER(2,NSMAX) — average number of particles injected per timestep at the left and right ends for each species

INJECT — logical flag to determine whether or not injection is actually to take place (input parameter)

The contents of HIST are:

IPLOT — number of timesteps between plotting snapshot diagnostics to plot file (input parameter)

IOUT — number of timesteps between printing snapshot diagnostics to output file (input parameter)

ISAV — number of timesteps between successive points on history plots (input parameter)

IHIST — number of points plotted on a given history plot (input parameter)

IPACK — number of timesteps between successive repackings of the particle storage arrays (input parameter)

ZED — logical flag for whether or not the zed interface is being used (input parameter)

MPLOT — tally variable associated with IPLOT

MOUT — tally variable associated with IOUT

MSAV — tally variable associated with ISAV

MHIST — tally variable associated with IHIST

MPACK — tally variable associated with IPACK

HEXTQ(HYMAX1) — history save array for external charge

HEXTI(HYMAX1) — history save array for external current

HLNESE(HYMAX1) — history save array for natural logarithm of the electrostatic energy

The dynamical variables are:

X(MAXLEN) — array containing all particle positions (normalized)

V(NVMAX, MAXLEN) — array containing all particle velocities (normalized)

NP(NSMAX) — array containing the current number of particles of each species

SRHO(NGMAX1, NSMAX) — charge density broken down by species

RHO(NGMAX1) — total charge density

E(NGMAX1) — electric field

PHI(NGMAX1) — electrostatic potential

EXTQ — charge on external capacitor

EXTI — current in external circuit

SIGMAX — net charge which has been absorbed by the right hand wall (NOT including charges which have passed through the wall, but have not been removed yet)

IT — timestep number

These are all the variables present in the main program. Those few important variables which are not covered here will be covered in the detailed description of the subroutine in which they appear, in the next Section.

VI Descriptions of Routines

MAIN

The main routine does nothing but call the various subroutines and increment the time counter. It is, however, useful for understanding the flow of the program. (It may be helpful to refer to Appendix A.) First SETUP is called to create all the linkages to the outside world, then START is called to read in the input parameters, set the problem parameters, and initialize the dynamical variables. Then, FIELDS is called to calculate the initial electric field, and OUTPUT is called to plot the initial state. Now the main loop is entered. This consists of moving the particles (MOVE), advancing the circuit (CIRCUIT), shuffling particles due to boundary conditions and storage limitations (ADJUST), calculating the electric field due to the new state of the system (FIELDS), incrementing the time counter, plotting the diagnostics if necessary (OUTPUT), and going back to the beginning of the loop. After the proper number of timesteps, FINISH is called to wrap things up and print timing information. Each of these routines, along with some of the logic behind them, will be discussed.

SETUP

SETUP is fairly straight-forward, but makes many calls to FORTLIB routines, and is highly installation-dependent. Its purpose is really to isolate these non-transportable tasks into one routine, so that the code can be moved to a new installation with a minimum of trouble. The tasks which SETUP performs are (in order): getting the name of the input file, creating a dropfile, starting the timing routine, linking to the terminal, opening the input file, creating the text output file, making a header (BOXID) for the text output file and microfiche plot output, and finally setting up the plot file (FR80ID).

The timing routine (TIMER) is a Livermore product, and generates a file which must be postprocessed by a routine called TALLY. TIMER has an entry named TIMEND for stopping the timing which is called in FINISH.

START

START reads in the problem parameters, and initializes the dynamical variables. It calls two large subroutines to do this: SPECIES reads in species dependent problem parameters, and LOAD initializes the particles. START itself checks for errors in the input data, and computes default values where possible, then computes all necessary parameters from the input parameters. It then renormalizes all velocities to the internal normalization.

SPECIES simply reads in the parameters which depend on species, and returns them to START. It is quite straight forward, but a little fancy footwork is required to allow the same namelist variables to be used for all the species.

LOAD is, conversely, quite sophisticated. It is able to set up virtually any kind of decently behaved distribution function. To do this, it integrates the distribution function, which is in a function subroutine name FDIST, once to normalize it, then integrates it again, to find the particular values of the integral at which particles belong. This means that a new distribution function can be introduced without having to work out its normalization analytically (which might be quite complicated). A set of 1024 particles ordered in velocity are found in this way. These particles are then loaded uniformly into the box in bit reversed order, re-using the same 1024 velocities until the proper number of particles have been loaded. Bit-reversing derives its name from the simplest procedure for generating a series of bit reversed numbers: write a sequence of sequential integers in base two, then reverse the binary digits, and convert them back to decimal (or whatever base you are accustomed to working in). This loading scheme ensures uniform loading of phase space with minimal recurrence effects. The bit reversing routine in LOAD is named REVERS (it is used for base 3 and 5 reversing also). The y and z direction are loaded with Maxwellian distributions, since we could think of no physical reason why one would want to load anything else. These are calculated using a rational polynomial approximation to the inverse of the cumulative normal distribution function taken from the

Handbook of Mathematical Functions by Abramowitz and Stegun formula 26.2.23, which was in turn taken from *Approximations for Digital Computers* by C. Hastings (1955, Princeton University Press). The routine is named GAUSIN. The velocities are loaded in 3-reverse and 5-reverse order, respectively, to minimize correlations between the velocity components. When the thermal velocity is zero, a special loader is invoked, which introduces some noise intentionally. This is done because without this noise, the distribution may be numerically stable purely due to lack of round-off error, i.e. an instability one is looking for may never materialize for lack of an initial excitation. The random noise is generated using the standard routine RANF, which returns a random variable uniformly distributed between 0 and 1. This random variable is used to displace the initial positions of the particles slightly.

MOVE

MOVE is responsible for advancing the velocities and positions of the particles. Three different algorithms are needed for the three cases NV=1,2,3. For a simple reference on how they work, see " Electromagnetic and Relativistic Plasma Simulation Models" by Langdon and Lasinski, in *Methods of Computational Physics* , **16** (1976); in particular, see Section B on time integration of the particles. The standard version of MOVE is vectorized, which means that it is tailored to process the particles in groups of 64, which is the size of the CRAY-1 vector registers. An alternate, non-vectorized, version exists (NMOVE), but it is several times slower.

Because of the particles which may have moved outside the simulation region but have not been removed yet, some special cases must be dealt with in MOVE. This is accomplished with the function subroutine CVMGT, which is a poor man's vectorizable IF statement. CVMGT takes three arguments. They are all evaluated, and then if the third argument has a logical true value, the value of the first argument is returned. Otherwise, the value of the second argument is returned. This means that some unnecessary computing is done, but this is far outweighed by the benefits of vectorizability.

CIRCUIT

CIRCUIT is active only when the inductance of the external circuit ($\hat{L}$) is non-zero. When this is the case, the circuit (a series RLC circuit, with both AC and DC bias voltages in series with it) is advanced with a leap-frog integrator similar to the particle integrator, but including a term which would be analogous to a viscous force on the particles (the resistive term in Kirchoff's law). Specifically, the difference equation solved is:

$$\hat{L}\frac{I_{n+\frac{1}{2}} - I_{n-\frac{1}{2}}}{\Delta t} + \hat{R}\frac{I_{n+\frac{1}{2}} + I_{n-\frac{1}{2}}}{2} + \frac{Q_n}{\hat{C}} + V_{0n} = V_n.$$

This equation is accurate when $\Delta t^2/\hat{L}\hat{C} << 1$ and $\hat{R}\Delta t/\hat{L} << 1$, i.e. when the physical timescales associated with the circuit are much smaller than the time step. The method is stable as long as $\Delta t^2/\hat{L}\hat{C}$ and $\hat{R}\Delta t/\hat{L}$ are both less than 2. Other schemes exist which are stable for all parameters, but they are seldom as accurate in the regime where this method is stable. For a comparison of the algorithms for $\hat{L} = 0$ and $\hat{L} \neq 0$, see Figure 1 of Appendix A.

To compute the potential bias due to the voltage source in the external circuit (V_0), the routine PHIS is called. While this routine is quite simple in the standard version of PDW1, it is worth having as a separate routine, since it will allow more complex time-dependent behavior as a very minor modification.

More sophisticated methods of advancing the circuit are, of course, possible with minor modification to CIRCUIT. This routine should also be able to deal with much more complex circuits when suitably modified.

ADJUST

After the particles and circuit have been advanced, it is necessary to include several other things which have occurred during the timestep. Included in the standard version of PDW1 are injection of particles, and the repacking of the particle storage arrays. Many other effects can

be included, though, such as reflection at boundaries, ionization in the interior of the simulation region, and Coulomb, or other, collisions.

The selection of velocities for injected particles is handled by the function subroutine VNEXT. These velocities are calculated in basically the same way that the velocities of the initial particles were calculated in LOAD. The relevant distribution function is integrated once to normalize it, and again to produce a set of 1024 velocities, which are stored in an array, and doled out as the particles are injected. Unlike the loading algorithm, the 1024 velocities are not re-used, but a new, slightly different set are calculated. The new velocities are calculated to produce a continuous string of bit-reversed velocities, so that the size of the 1024 element storage array has no effect on the results.

FIELDS

FIELDS' ultimate purpose in life is to compute the electric field, and the potential drop across the simulation region in order to allow the integration of the particle positions and circuit over another timestep. To do this, it first integrates the charge density, then it must do one of two things depending on whether or not the inductance of the external circuit is zero. If the inductance is not zero, the circuit has been advanced, and FIELDS need only compute the charge on the right hand end wall, to obtain a Neumann boundary condition which allows a one-pass solution to Poisson's equation. If the external inductance is zero, the equation governing the external circuit is of lower order than the equation governing the particles in the plasma, and the circuit must be treated as a mixed boundary condition on the field solve. Specifically,

$$\hat{R}\frac{Q_n - Q_{n-1}}{\Delta t} + \frac{Q_n}{\hat{C}} + V_{0n} = V_n.$$

Since Q_n and V_n are linearly related, this can be solved simply using a one-step predictor-corrector. As with CIRCUIT, the routine PHIS is called to get the external potential bias (V_0) in the circuit.

OUTPUT

OUTPUT handles all diagnostic output. Diagnostic variables which are not essential to the simulation are *all* computed here, and are only computed when necessary. To accomplish this, tally variables exist to keep track of when a given type of diagnostic should be plotted. These variables are detailed in the description of the common block HIST.

To simplify the logic of the program, and to make the main routine as installation independent as possible, four routines handle all the graphs and labeling: PLTXVX, GRAPH, LABEL and HLABEL. PLTXVX makes phase space plots, and makes an educated guess as to what the best scale is. GRAPH makes an ordinary graph of any variable vs. position or time. LABEL labels snapshot graphs, i.e. graphs which picture something at a single time. HLABEL labels history plots, which extend over a range of times, and so must be differently labeled. These routines should make the introduction of new diagnostics as painless as possible.

FINISH

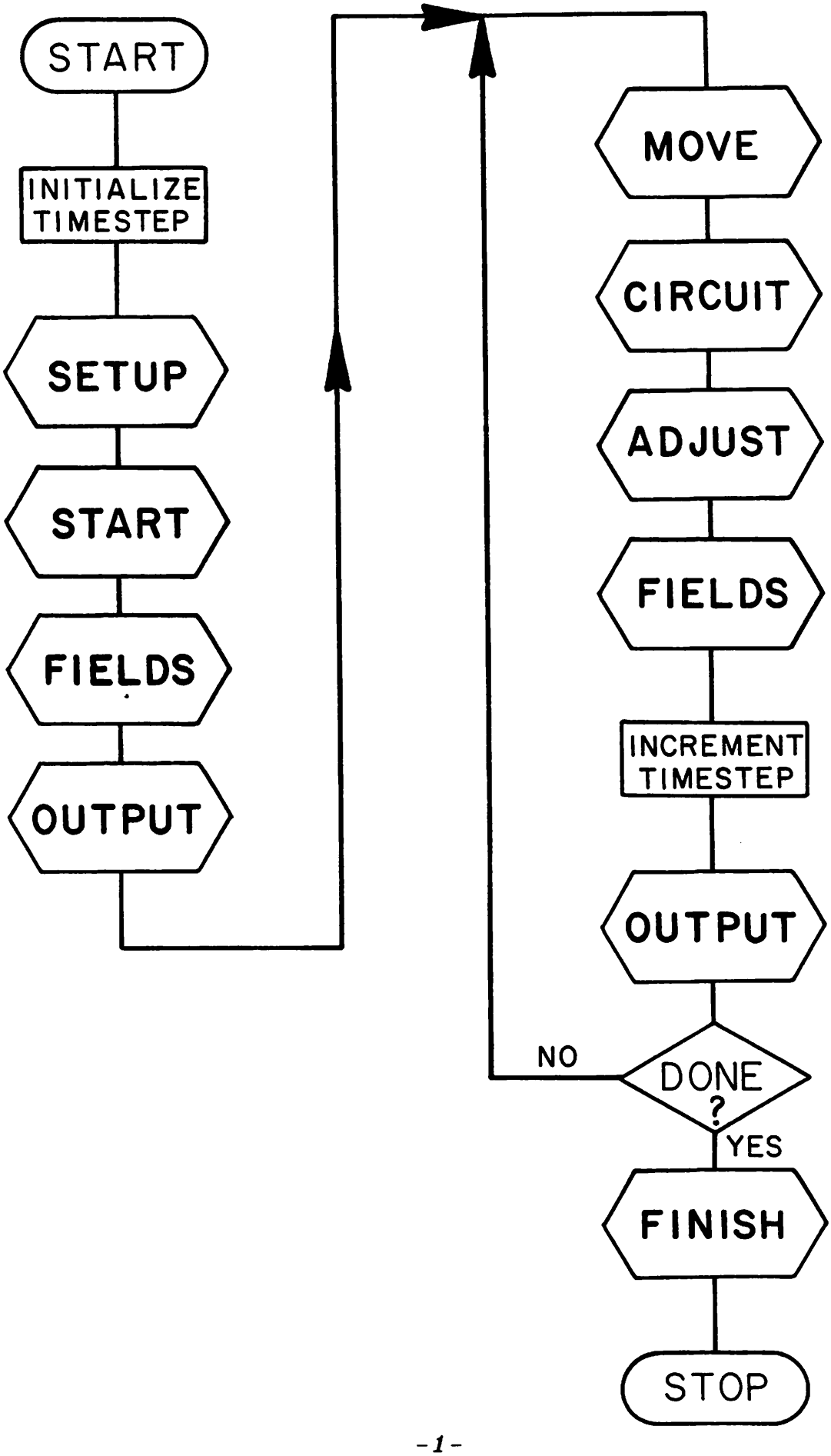
FINISH writes a summary of the total time the run has used, and calls TIMEND, which stops the collection of detailed timing information which the call to TIMER in SETUP started. Since there are no other closing operations necessary on the MFE system, FINISH then calls EXIT, which stops the program.

Acknowledgements

The author wishes to gratefully acknowledge the guidance and assistance of Prof. C. K. Birdsall, Dr. T. L. Crystal and Dr. S. Kuhn (a visitor from Innsbruck) who, together with the author, comprised the steering committee for the Plasma Device Workshop and the development of PDW1. Dr. A. B. Langdon, though not a regular member of the workshop, was particularly helpful. Also to be acknowledged for their help as members of the workshop are Dr. M. Hudson, Dr. I. Roth, V. Thomas, K.-Y. Kim, P. Gray, L. A. Schwager, and particularly N. Otani. All computation was performed on the computers at National Magnetic Fusion Energy Computer Center at Livermore. The work was supported by DOE contract number DE-AT03-76ET53064.

APPENDICES

Appendix A — Algorithm and Flow Charts



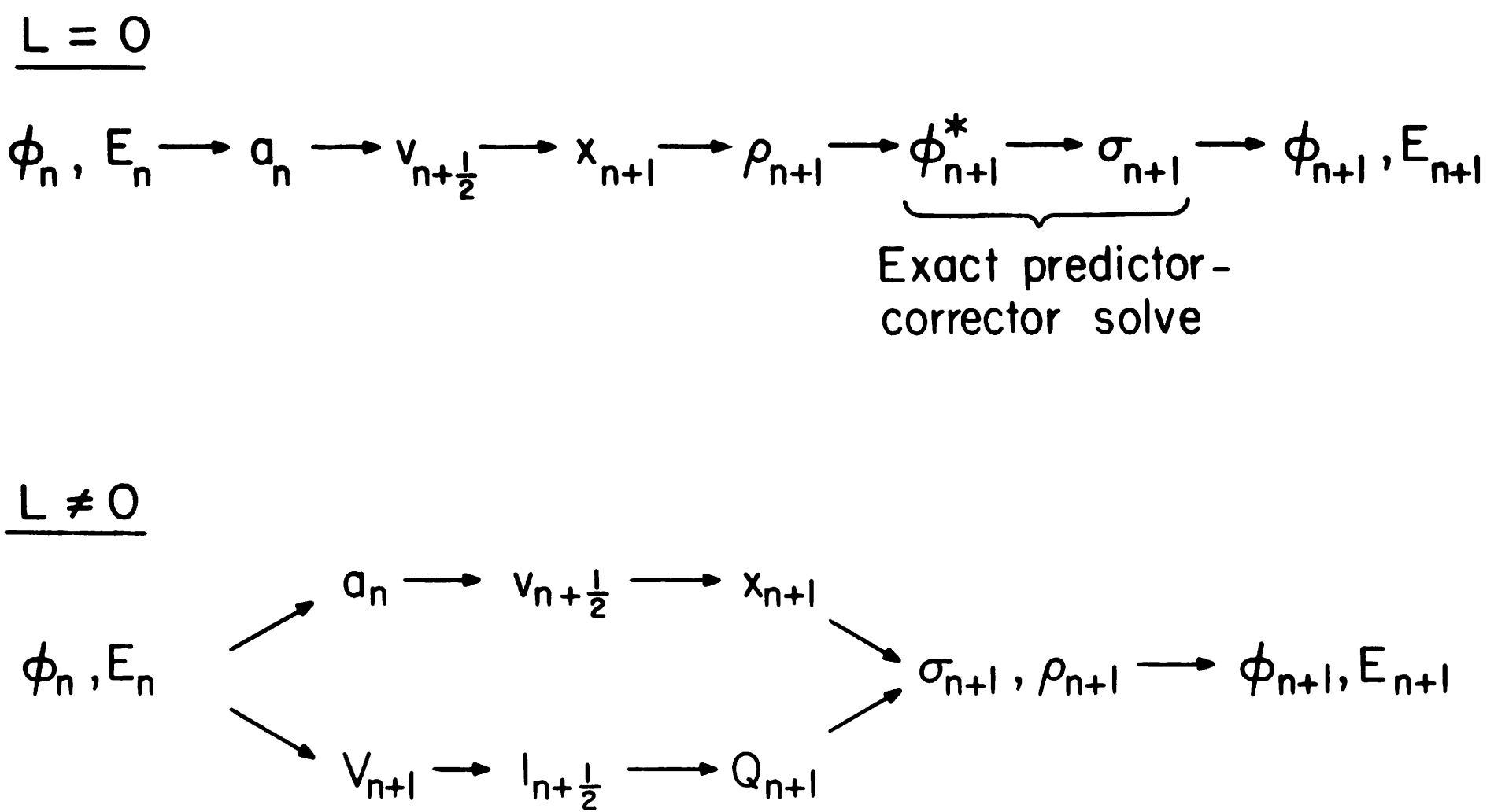
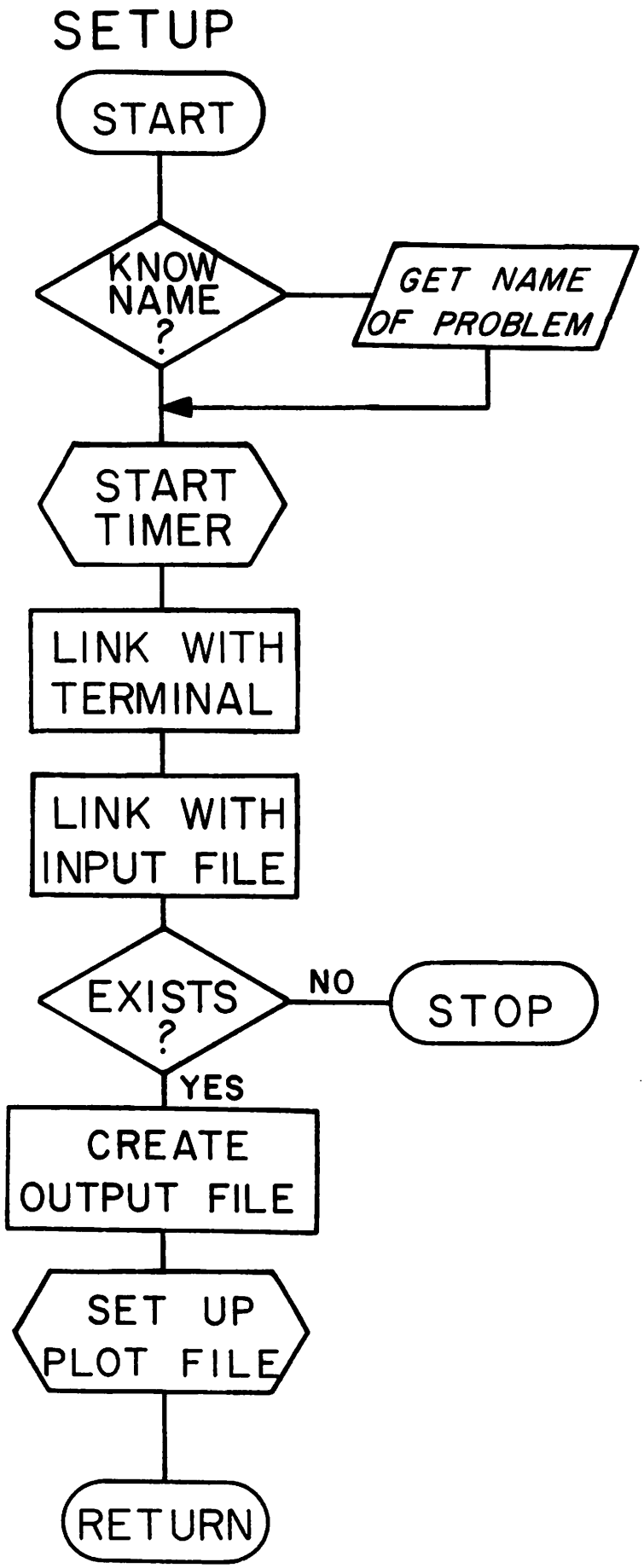
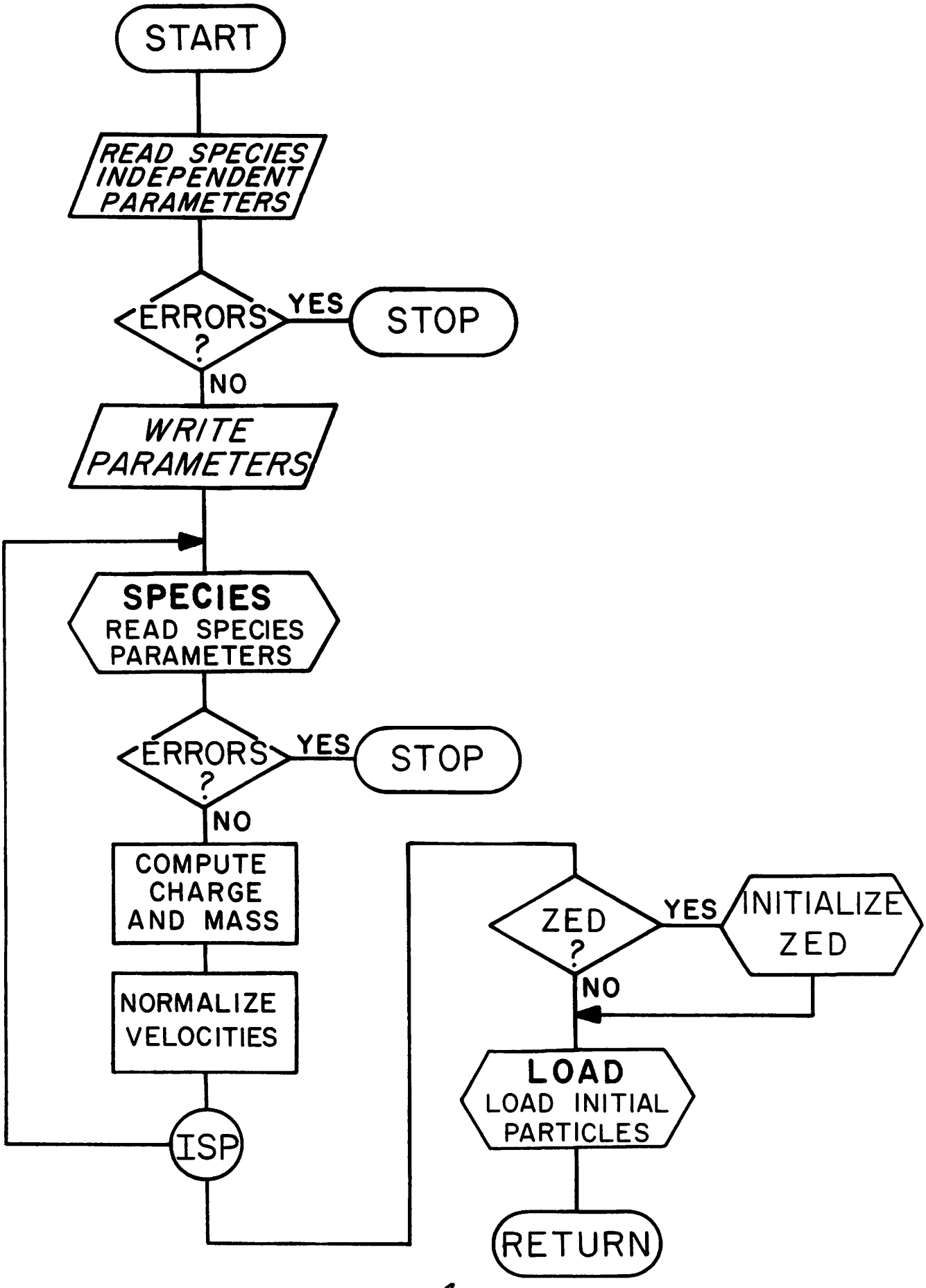
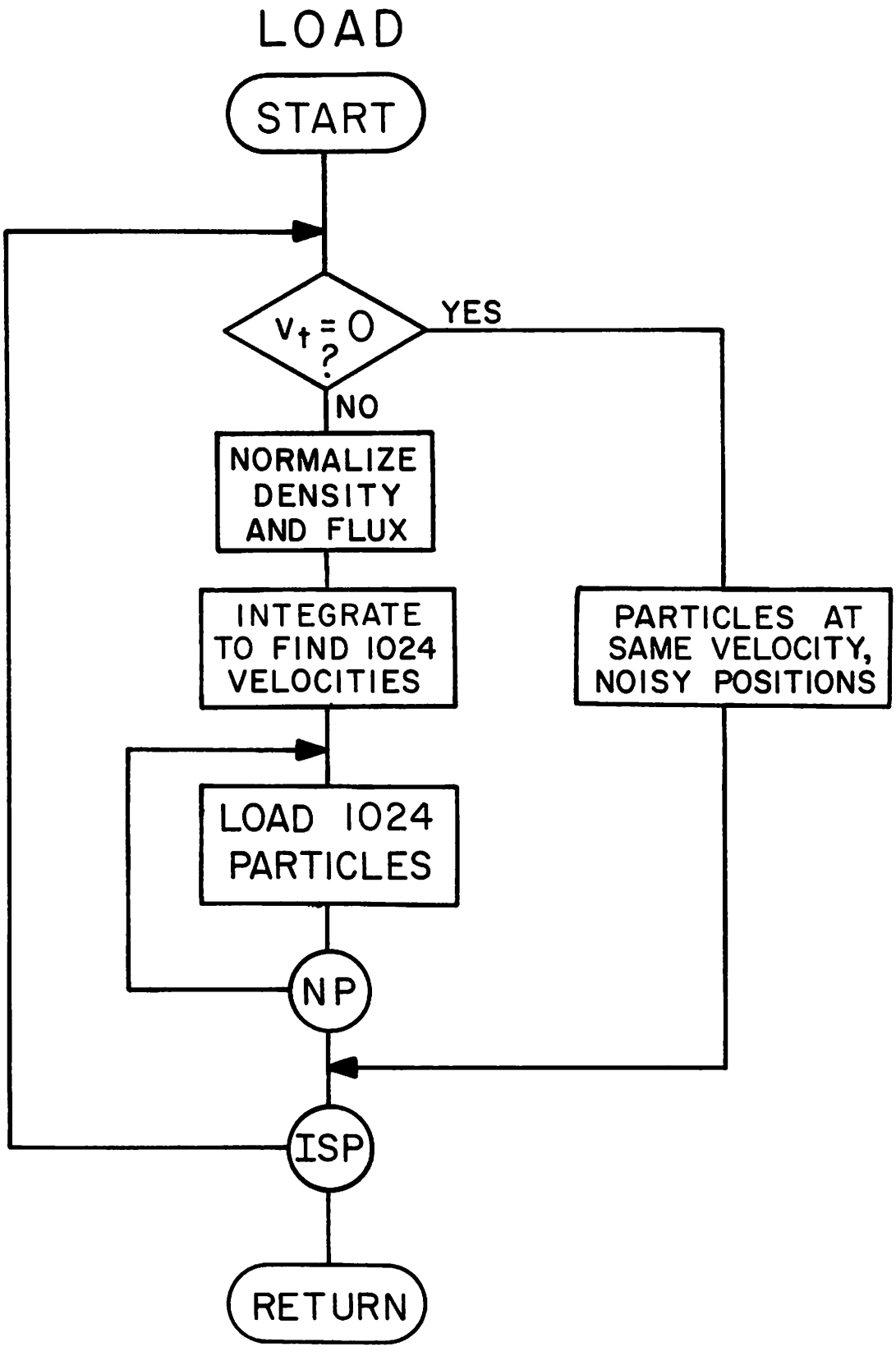


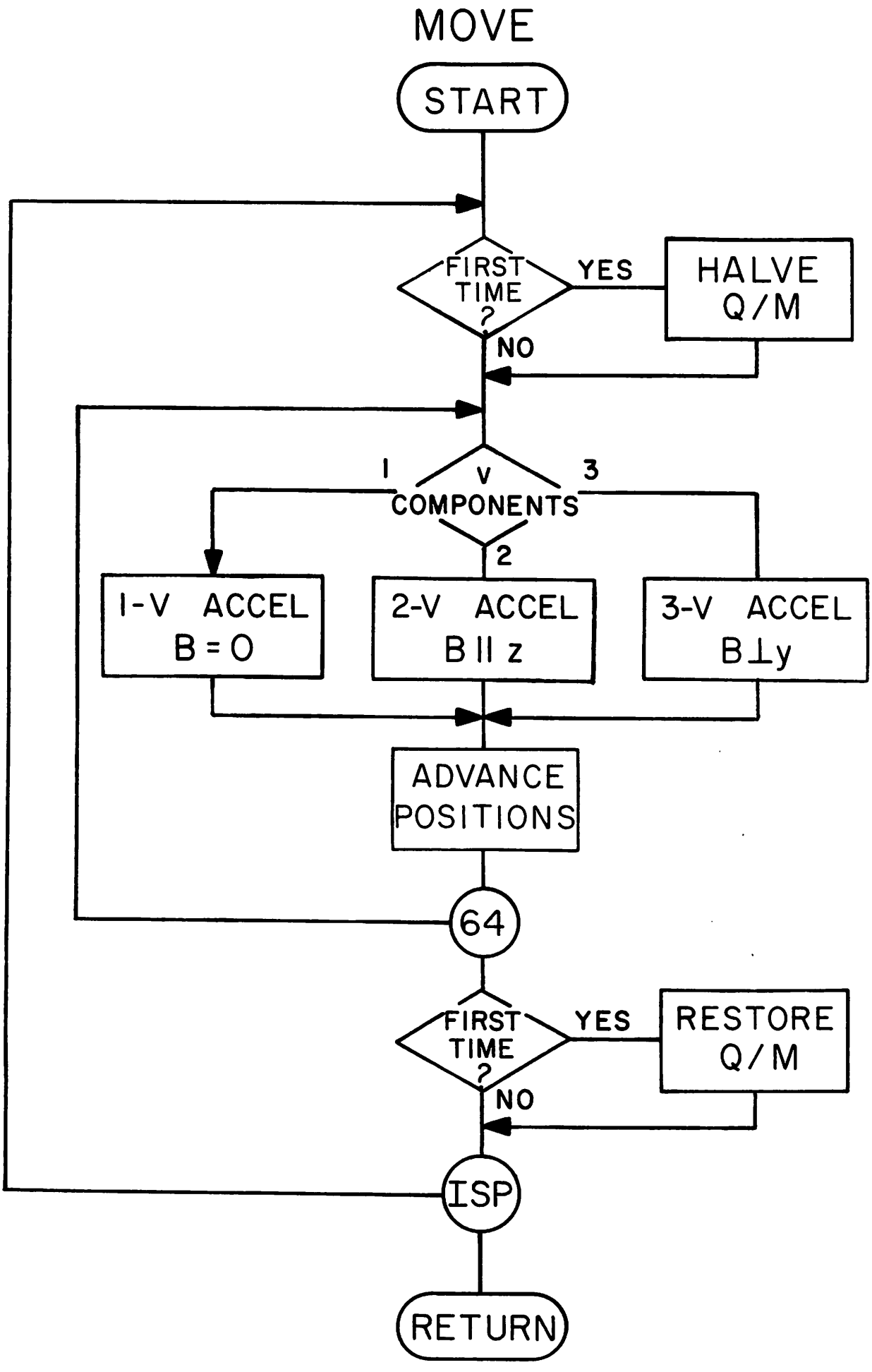
Figure 1. Algorithms for particle and circuit advancement


SETUP
START
KNOW NAME ?
GET NAME OF PROBLEM
START TIMER
LINK WITH TERMINAL
LINK WITH INPUT FILE
EXISTS ?
NO
STOP
YES
CREATE OUTPUT FILE
SET UP PLOT FILE
RETURN

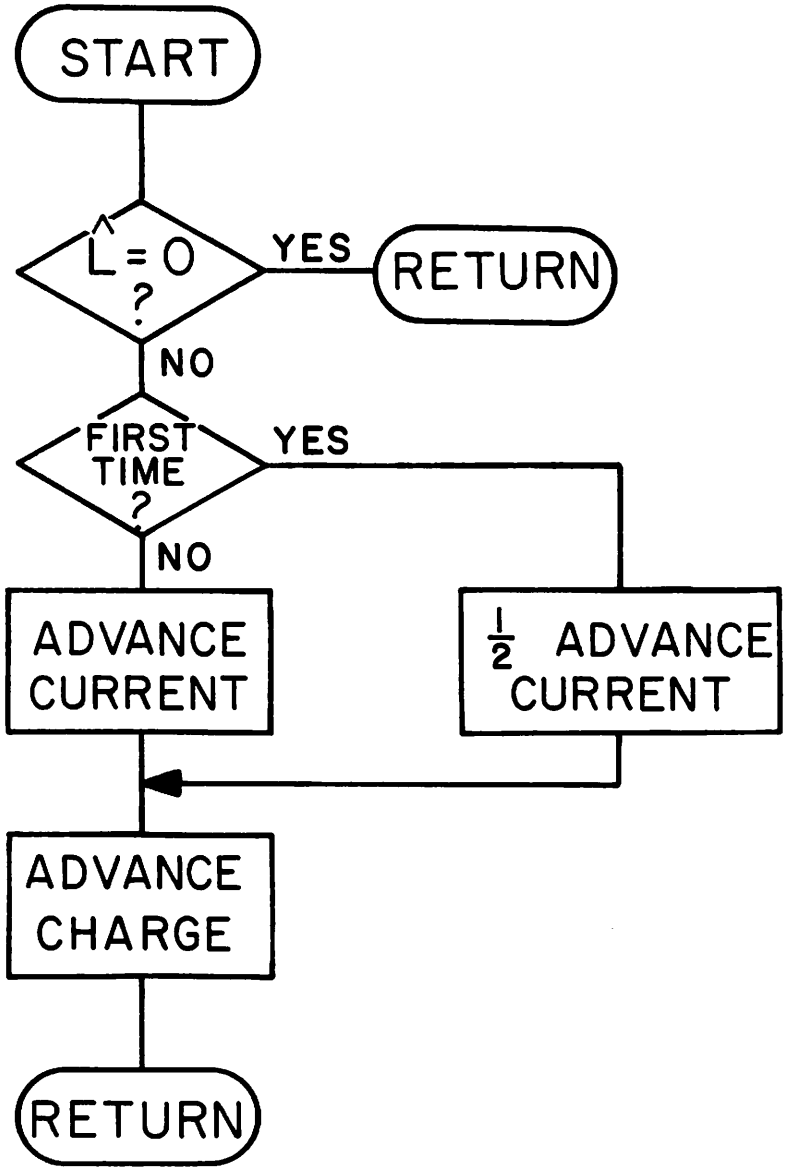
START


START
READ SPECIES INDEPENDENT PARAMETERS
ERRORS ?
YES
STOP
NO
WRITE PARAMETERS
SPECIES
READ SPECIES PARAMETERS
ERRORS ?
YES
STOP
NO
COMPUTE CHARGE AND MASS
NORMALIZE VELOCITIES
ISP
ZED ?
YES
INITIALIZE ZED
NO
LOAD
LOAD INITIAL PARTICLES
RETURN

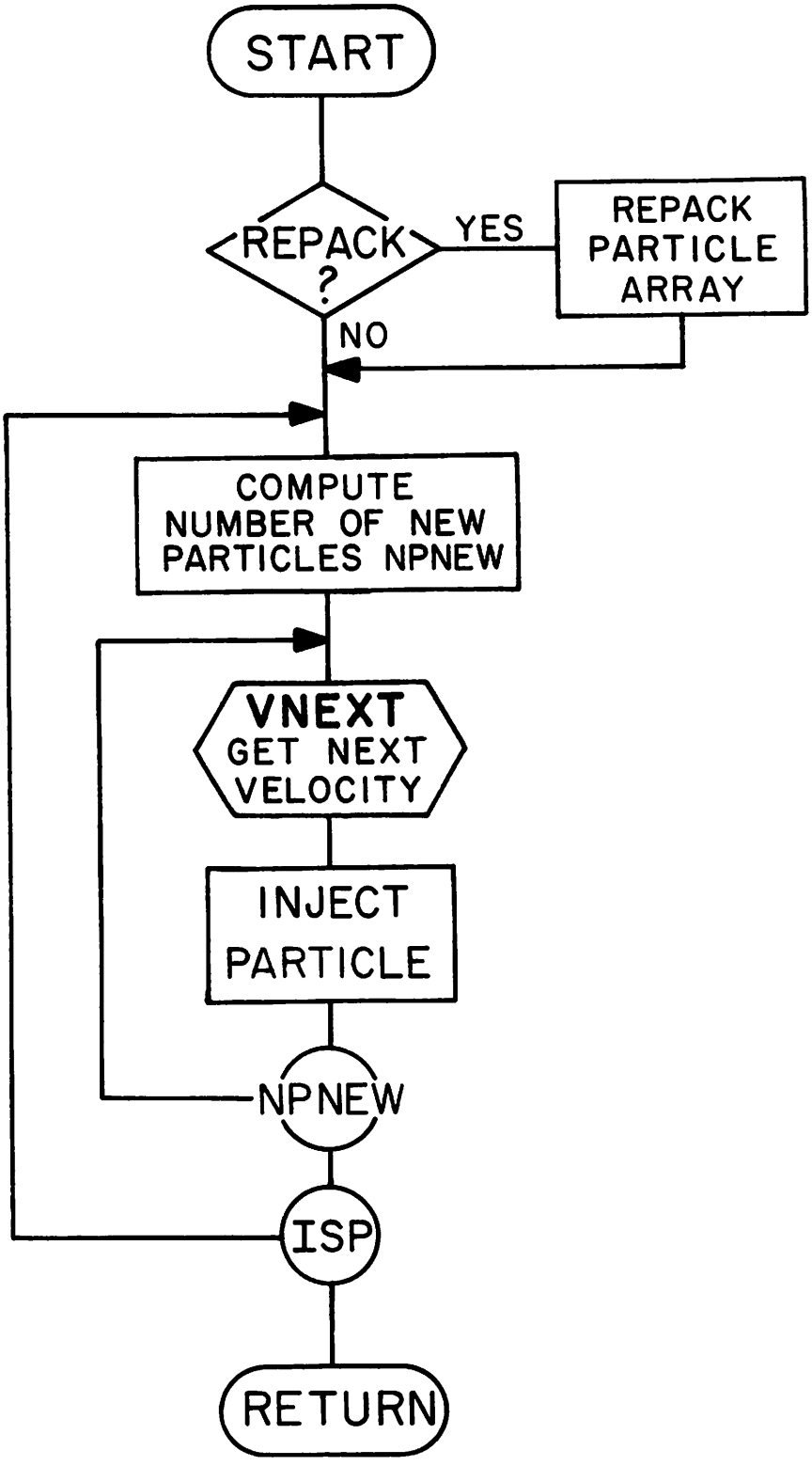

LOAD
START
$V_t = 0$?
YES
NO
NORMALIZE DENSITY AND FLUX
INTEGRATE TO FIND 1024 VELOCITIES
PARTICLES AT SAME VELOCITY, NOISY POSITIONS
LOAD 1024 PARTICLES
NP
ISP
RETURN


MOVE
START
FIRST TIME ?
YES
HALVE Q/M
NO
1
V
3
COMPONENTS
2
1-V ACCEL B = 0
2-V ACCEL B ll z
3-V ACCEL B ⊥ y
ADVANCE POSITIONS
64
FIRST TIME ?
YES
RESTORE Q/M
NO
ISP
RETURN

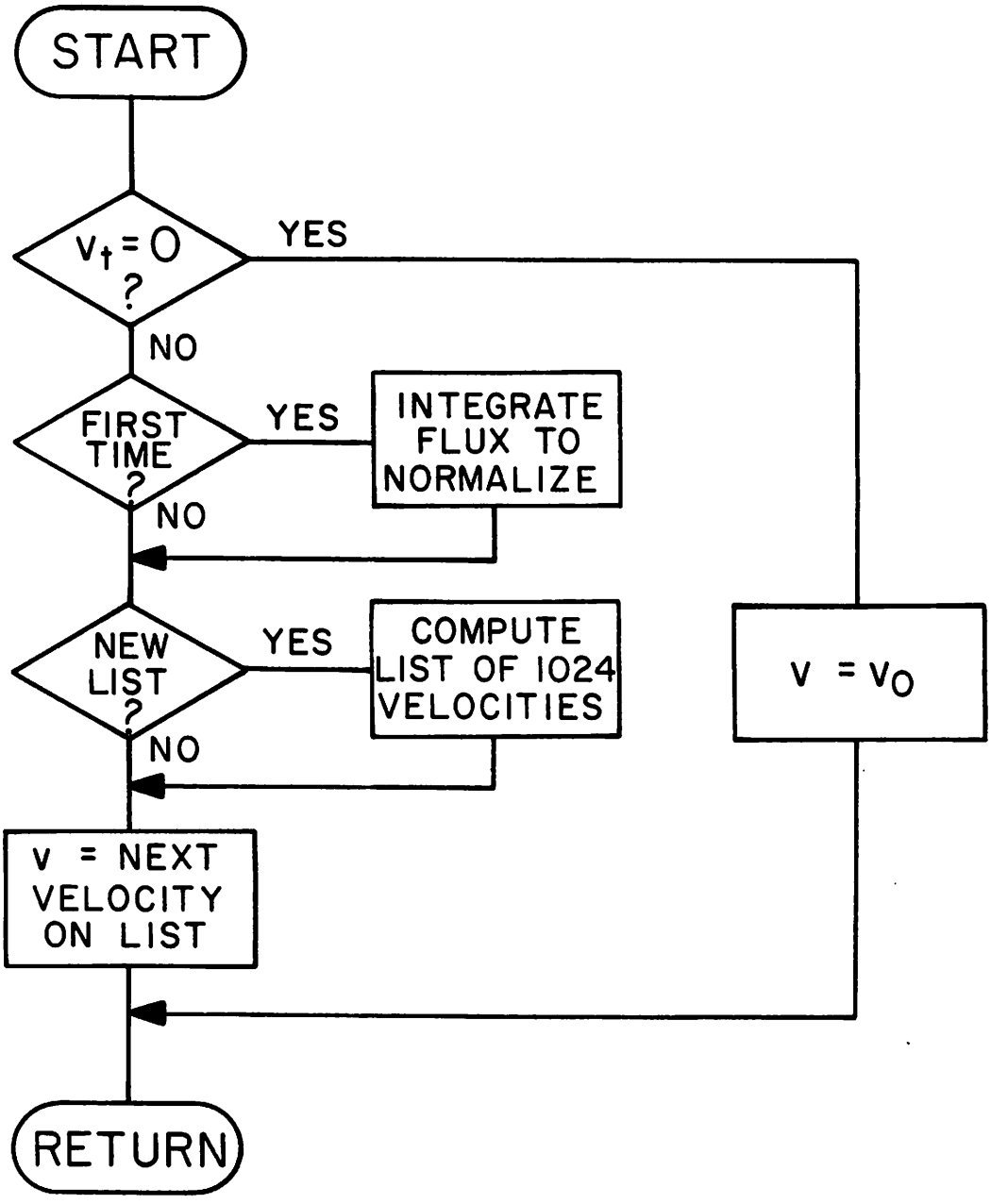
CIRCUIT


START
L = 0 ?
YES
RETURN
NO
FIRST TIME ?
YES
NO
ADVANCE CURRENT
½ ADVANCE CURRENT
ADVANCE CHARGE
RETURN

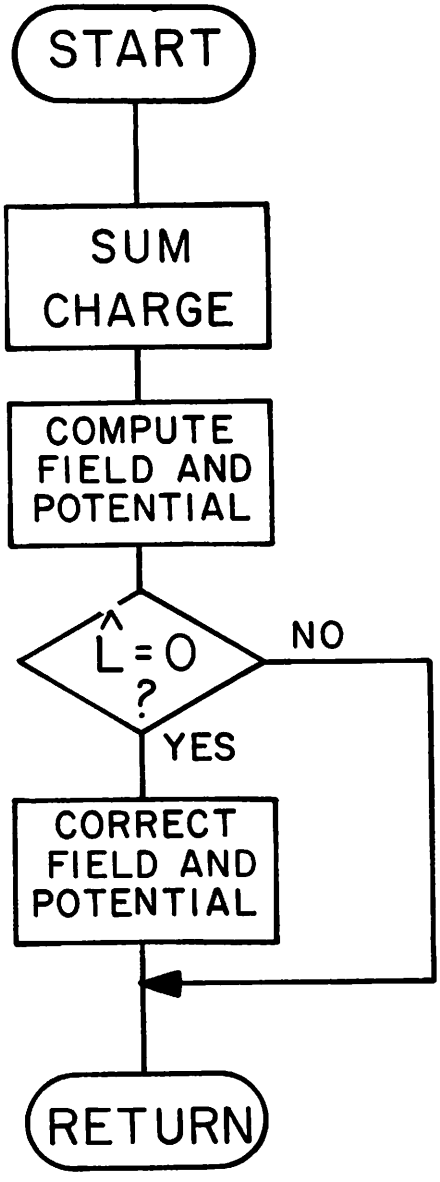
ADJUST


START
REPACK ?
YES
REPACK PARTICLE ARRAY
NO
COMPUTE NUMBER OF NEW PARTICLES NPNEW
VNEXT
GET NEXT VELOCITY
INJECT PARTICLE
NPNEW
ISP
RETURN

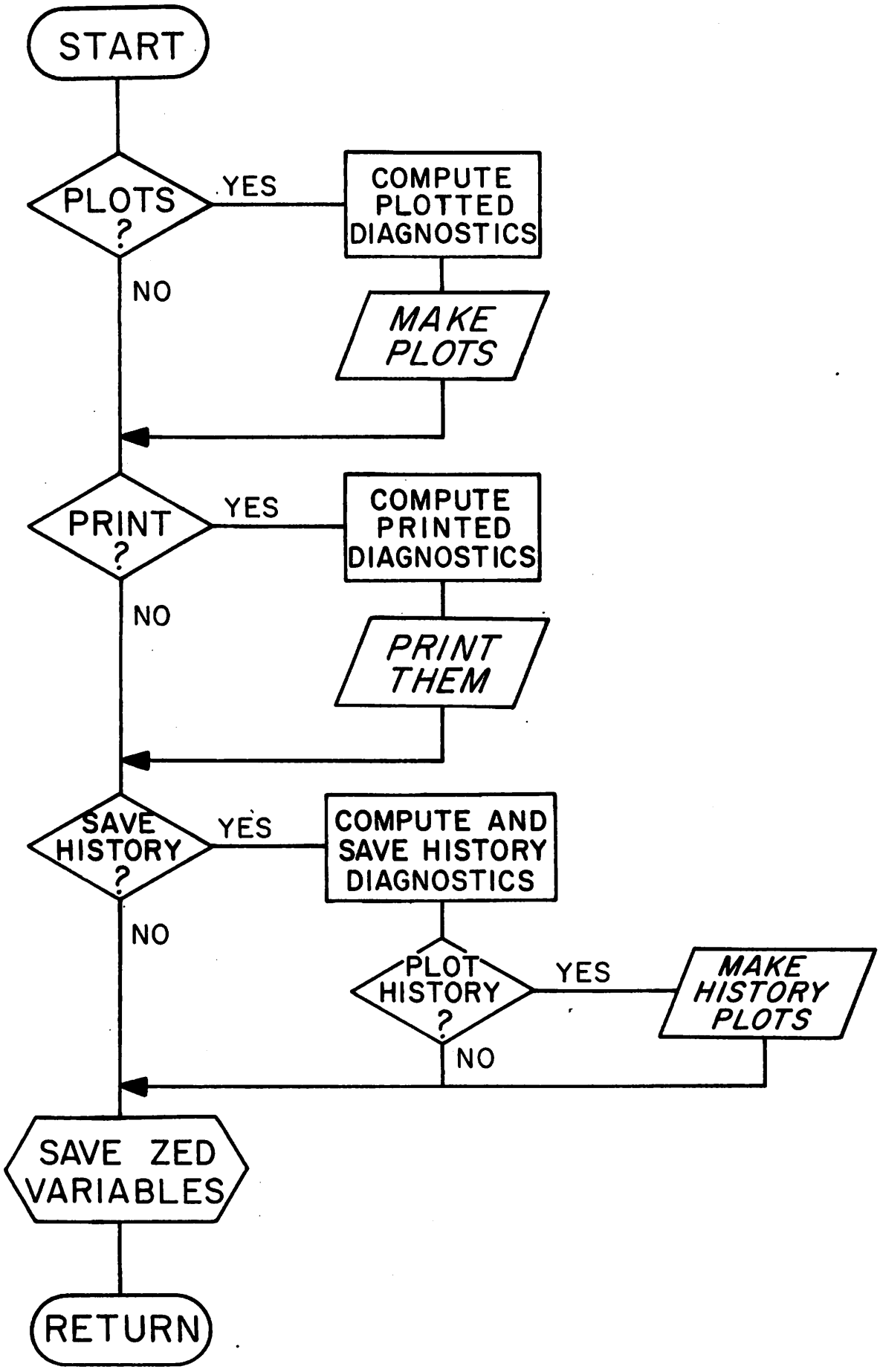
VNEXT


START
$v_t = 0$?
YES
NO
FIRST TIME ?
YES
INTEGRATE FLUX TO NORMALIZE
NO
NEW LIST ?
YES
COMPUTE LIST OF 1024 VELOCITIES
NO
$v = v_0$
v = NEXT VELOCITY ON LIST
RETURN

FIELDS


START
SUM CHARGE
COMPUTE FIELD AND POTENTIAL
$\hat{L} = 0$?
NO
YES
CORRECT FIELD AND POTENTIAL
RETURN

OUTPUT


START
PLOTS ?
YES
COMPUTE PLOTTED DIAGNOSTICS
NO
MAKE PLOTS
PRINT ?
YES
COMPUTE PRINTED DIAGNOSTICS
NO
PRINT THEM
SAVE HISTORY ?
YES
COMPUTE AND SAVE HISTORY DIAGNOSTICS
NO
PLOT HISTORY ?
YES
MAKE HISTORY PLOTS
NO
SAVE ZED VARIABLES
RETURN

Appendix B — ZED Interface

Niels F. Otani (Dr. W. M. Nevins, LLNL)

Three routines have been created to facilitate the generation of state and history files compatible with the newly available MFE-CRAY-version ZED Postprocessor. The routines are easy to use — in addition to calling the routines themselves, the user need only supply namelists containing all variables which the user may wish to write, a special array named "modetabl", and an input file at runtime. These routines are named SETFILEZ, WRFILEZ and WRFILPZ.

SETFILEZ is the initialization routine and therfore should be called before entering the main timestep loop. SETFILEZ is called with the following arguments:

```
call setfilez(iushist,iusstate,infile,modetabl,dt,ny,ntuu)
```

where

iushist — unit specifier to be associated with the history file

iusstate — unit specifier to be associated with the state file

infile — name of the input file to be read by SETFILEZ

modetabl — is an array at least 37b in length (note: 37b means 37-octal), and on input should contain the mode variable names. For example, if the ZED quantity `tv = 22b` is to be associated with the variable "phi", then the user must set `modetabl(22b) = "phi"` prior to calling SETFILEZ.

dt — the timestep size

ny — dimension of the y- (second) index. Note: ny is only used for ZED labeling conventions and only with (t=3)-type variables. See the source code for details.

ntuu — on exit, contains the last timestep required by these ZED file-writing routines

SETFILEZ reads an input file which the user uses to specify which history and mode variables are to be written. The following typical input file serves to illustrate the input file format:

```
2000 steps from step 100
var1,var2,
var3
var4, var5, var6
every 10 steps
q, 0 5 1, 1
phi, 3, 2
ex, 0 3 3, 1 3 1 &
```

The first line designates that 2000 timesteps beginning with timestep 100 will be monitored by these ZED-interface routines. If the first timestep to be read is timestep 0, the "from step 100" clause may be omitted. The next three lines specify that the (unindexed) variables var1, var2, var3, var4, var5 and var6 are to be written on one line and that the var3, var4, var5 and var6 are to be written to the history portion of the history file. Note that any number of history variables may be written on one line and that commas are ignored. Any number of lines may be used.

The line with "every" as its first word signals the modes specification portion of the file. The clause "every 10 steps" indicates that modes information will be written to the modes portion of the history file every 10 timesteps. The next three lines illustrate the format used for describing which modes are to be written. Only one mode variable may be specified per line. The range of each of two indices is specified by one or three integers. If one number appears, only modes with that value for the corresponding index will be written. If three numbers are indicated, modes within the range defined by the first two numbers (lowest index first) will be written by increments specified by the third number. If an ampersand appears at the end of a mode descriptor line, the corresponding variable will be assumed to be complex, and the imaginary part of the variable will be stored after the real part in the modes portion of the

history file. Note that for one-dimensional arrays, the second index is always 1.

The names of the history and state files created by SETFILEZ are, by default, "history" and "state". These names may be changed by accessing the common block CZFNAMES in the user program:

```
common /czfnames/ statenam, histnam
```

SETFILEZ allows new ZED files to be created in the same run by 1) closing the old history and state files, 2) assigning new history and state filenames via CZFNAMES if desired, and 3) recalling SETFILEZ. This feature is useful in codes with restart capabilities.

The routines WRFILEZ and WRFILPZ are called in the timestep loop by the user program and perform buffered writes to the ZED files according to specifications passed from SETFILEZ. WRFILPZ writes pointer-designated mode array data, while WRFILEZ writes conventional mode array data. Both routines also write conventional history (unindexed) data. WRFILPZ is specially designed for use with the author's pointer-defined arrays; most users, using conventionally defined arrays, will want to use WRFILEZ.

Actual disk writes to the history file are accompanied by corresponding updates of the state file, thus allowing the user to use the ZED postprocessor on the ZED files to monitor progress while the main code is still running.

WRFILEZ and WRFILPZ are called as follows:

```
call wrfilez(it,histnlst,modenlst,itype)
call wrfilpz(it,histnlst,modeptrs,mdim,itype)
```

where

it — current timestep number

histnlst — name of a namelist containing variables to be written into the history file at this call

modenlst — name of a namelist containing arrays to be written into the modes portion of the history file at this call

modeptrs — name of a namelist containing pointers of the mode array quantities quantities to be written into the modes portion of the history file at this call. Pointer names must be of the form p*var* where *var* is the name of the corresponding mode array.

mdim — an array of length 4. mdim must be filled with the dimensions of all arrays in modeptrs prior to calling WRFILPZ. WRFILPZ will assume all arrays have dimension (mdim(1):mdim(2), mdim(3):mdim(4)). Important: the values of mdim **must** be correctly set to insure proper writing of the data by WRFILPZ.

itype — 1 if mode arrays are real

2 if mode arrays are complex

Important: all arrays represented in a mode namelist **must** be of the same type, and itype **must** be correctly set to ensure the proper array quantities are written. If both real and complex arrays are to be written, the user simply calls WRFILPZ twice, once for each type of array.

Either or both routines may be called more than once in the timestep loop with mutually exclusive namelists, permitting the user to write different mode array types (and different mode array dimensions, in the case of WRFILPZ). This also allows the user to write different variables from different parts of the timestep loop. The same array may even be written more than once in the timestep, by using EQUIVALENCE statements and then including different names for the same array in different namelists. This feature is useful for writing an array and later its Fourier transform stored in the same array.

Simple usage of the ZED interface routines is illustrated in the following program:

```
      dimension q(0:5), phi(64,64), ex(0:63, 0:63), rho(64,64)
      namelist /hnlst/ var1, var2, var3, var4, var5, var6
      namelist /mnlst/ q, phi, ex, rho
      dimension modetabl(37b)
c     Initialize code run:
        dt = 0.2
        (Other initializations required by code)
        modetabl(21b) = "q"
        modetabl(22b) = "phi"
        modetabl(23b) = "ex"
        modetabl(24b) = "rho"
c     Read input file "inzed" and initialize ZED interface routines:
        call setfilez(8,9,"inzed",modetabl,dt,1,ntuu)
c     Main timestep loop:
        do 100 it=0,ntuu
        .
        .
        call wrfilez(it,hnlst,mnlst,1)
        .
        .
100     continue
        call exit
        end
```

This routine will write history variables var1, var2, var3, etc., and real mode variables q,

phi, ex and rho to a history file named "history" with history state information written to state file "state" according to specifications contained in input file "inzed".

ZED compatibility is provided in PDW by the six cliches CLZED1 through CLZED6 in the PDW source code. The user may remove these cliches if ZED post-processing is not required.

To produce ZED-compatible output files, the user should set zed = .true. in PDW's main namelist and append the ZED portion of the input file immediately following the last species namelist using the ZED input file format described above. No provision has been made for writing "history" variables, so the second line should be the line beginning with "every" (cf. above). The variables and arrays which may be written to the "modes" portion of the file are listed opposite the definitions of the array MODETABL in CLZED1. No "$" is required following the ZED portion of the file.

The ZED post-processor may be used either during or after the run to process the data contained in the ZED files. A summary of the use of the ZED post-processor may be found in Ref. 247. The history file will be named Hprobnam, and the state file, Sprobnam, where probnam is the input problem name given the run. ZED knows variables and arrays written to the ZED files by octal numbers. The octal number corresponding to a variable or array is the index of array element of MODETABL containing the name of the variable or array, as defined in CLZED1.

Should the user wish to write other arrays besides those defined to MODETABL in CLZED1, she/he should add the array to namelist MAINLST in CLZED1 if the array is defined in the main routine, or to namelist OUTZNLST in CLZED4 if the array is defined in subroutine OUTPUT. One of the elements of array MODETABL between MODETABL(11b) and MODETABL(37b) should be defined with the alphanumeric name of the array.

A short summary of the function of the cliches and subroutines used by PDW's ZED interface follows :

CLZED1 sets up the namelist MAINLST of variables known to the main routine which may be written, stores the location of the namelist in common block CZED, and defines the elements of the array MODETABL with the names of all variables which may be written (including those not known to the main routine).

CLZED2 obtains the input file name and calls subroutine SETZED.

CLZED3 defines the subroutine SETZED.

CLZED4 sets up the namelist OUTZNLST of variables known to subroutine OUTPUT which may be written to the ZED output files and makes the location of OUTZNLST known via common block CZED.

CLZED5 calls the subroutine WRZED.

CLZED6 defines the subroutine WRZED.

SETZED transfers the ZED portion of the input file to a temporary file, defines the names of the history and state files, and calls SETFILEZ.

WRZED locates the namelists MAINLST and OUTZNLST using CZED and the calls WRFILEZ once for each namelist.

Appendix C — Useful Integrals for PDW1 Input

Two input parameters can present difficulty: the injected charge density and the injected current. In terms of the emitted distribution function, these are:

$$\rho_{0l} = \int_{v_c}^{\infty} f_0(v)\, dv$$

and

$$j_{0l} = \int_{v_c}^{\infty} v f_0(v)\, dv.$$

In the standard version of PDW1, f_0 is Maxwellian:

$$f_0(v) = \frac{\rho_0}{\sqrt{2\pi} v_t} e^{-\frac{(v-v_0)^2}{2v_t^2}}.$$

Where ρ_0 is the charge density which would result if the distribution were a full Maxwellian. The integrals can be simplified by introducing the normal distribution functions

$$Z(t) \equiv \frac{1}{\sqrt{2\pi}} e^{\frac{t^2}{2}}$$

$$P(t) \equiv \int_{-\infty}^{t} Z(u)\, du.$$

Note that $P(t)$ is not equal to $\mathrm{erf}(t)$, but rather

$$P(t) = \frac{1}{2}[1 + \mathrm{erf}(\frac{t}{\sqrt{2}})]$$

Using these two functions, the input parameters become

$$\rho_{0l} = \rho_0 P(\frac{v_0 - v_c}{v_t})$$

and

$$j_{0l} = \rho_0 v_t Z(\frac{v_0 - v_c}{v_t}) + \rho_0 v_0 P(\frac{v_0 - v_c}{v_t}).$$

P and Z are tabulated in the *CRC Handbook of Mathematics* and in the *Handbook of Mathematical Functions* by Abramowitz and Stegun, the latter of which also has some useful numerical approximations.

Appendix D — PDWMAX

The input parameters for PDW1 are unnecessarily laborious for problems in which a full Maxwellian plasma is to be loaded and injected throughout the simulation. To simplify such problems, a minor modification of PDW1 was made, and named PDWMAX. Only the species dependent parameters were altered (plus some changes were made in LOAD). The new (simplified) species dependent parameters are:

QM — charge to mass ratio (default -1.)

WP — plasma frequency for the given species (default 0.)

V0 — drift velocity (applies to distributions injected from both ends — default 0.)

VT — thermal velocity (for both ends — default 0.)

VY, and VZ — drift velocities in Y and Z directions (for both ends — default 0.)

N0 — number of particles to be loaded initially (default 0)

INJECT — logical flag for whether or not particles will actually be injected (default .TRUE.)

The additional section in LOAD calculates the integrals necessary to know how many particles of each species to inject at each end. This takes a considerable burden off of the user. Another advantage of PDWMAX is that there are no minor non-neutrality errors when loading. This tends to happen in PDW1 because the number of particles to be loaded initially for each direction is rounded off from the exact number determined by the input parameters. Thus there may be a fraction of a charge missing or extra at each end of the simulation region. (Loading the rounded number of particles uniformly turns out to be a bad idea, since the charge of only a single particle spread out over the entire simulation is often enough to create significant potentials.) PDWMAX loads each species in one swoop, and since the initial number of particles is input as an integer, there are no errors at the end walls.

Appendix E — Example Problems

1. Pierce Diode — stable regime

Parameters:

NSMAX = 1

NGMAX = 128

NVMAX = 1

NPMAX = 2000

HYMAX = 500

Input file:

```
box bnn pierce diode I
nt=500 rhoback=4.  backj=4.  iplot=100 iout=100 isav=1 $end
j0l=4.  v0l=1.  fluxl=1024.  empty=.false $end
```

Comments:

Since $\omega_p L/v_0 = 2 < \pi$, the least stable mode is still stable.

Note the jump in charge density and electric field at the right hand wall. Why doesn't this ruin the simulation?

Try running this with an incommensurate number of particles and grid points (say 200 particles and 128 grid cells). Why does this give such poor results?

For more information on the Pierce instability, see T. L. Crystal and S. Kuhn UCB/ERL Memo (Mar 1984).

```
box b04 pierce diode I
nsp       = 1
ng        = 128
nt        = 500
dt        = 7.8125e-03
nv        = 1
length    = 1.
area      = 1.
eps0      = 1.
b         = 0.
psi       = 0.
rhoback   = 4.
backj     = 4.
extr      = 0.
extl      = 0.
extc      = 1.e+20
dcbias    = 0.
acbias    = 0.
w0        = 0.
q0        = 0.
i0        = 0.
sigma0    = 0.
iplot     = 100
iout      = 100
isav      = 1
ihist     = 500
ipack     = 10
zed       = 0
$

species 1
qm        = -1.
j0l       = 4.
j0r       = 0.
v0l       = 1.
v0r       = 0.
vcl       = 0.
vcr       = 0.
vtl       = 0.
vtr       = 0.
vyl       = 0.
vyr       = 0.
vzl       = 0.
vzr       = 0.
fluxl     = 1024.
fluxr     = 0.
empty     = 0
inject    = -1
$
timestep     0   species 1   np   1024
timestep   100   species 1   np   1025
timestep   200   species 1   np   1025
timestep   300   species 1   np   1025
timestep   400   species 1   np   1025


timestep   500   species 1   np   1025
Time used (sec): cpu=  2.57e+00 io=  2.93e+00 sys=  8.37e-02
```

NEUTRALIZING BACKGROUND

CURRENT OFFSET FOR DIAGNOSTICS

EXTERNAL SHORT CIRCUIT

BEAM VELOCITY

COLD BEAM

```
NSP     = 1
NG      = 128
NT      = 500
DT      = 7.8125E-03
NV      = 1
LENGTH  = 1.
AREA    = 1.
EPSO    = 1.
B       = 0.
PSI     = 0.
NEGBACK = 4.
BACKJ   = 4.
EXTB    = 0.
EXTL    = 0.
EXTS    = 1.E+20
DCBIAS  = 0.
ACBIAS  = 0.
WD      = 0.
ND      = 0.
ID      = 0.
SIGMAO  = 0.
IPLOT   = 100
ICUT    = 100
ISAV    = 1
IHIST   = 500
IPACK   = 10
ZED     = 0
$

SPECIES 1
QM      = -1.
JBL     = 4.
JBR     = 0.
VBL     = 1.
VBR     = 0.
VEL     = 0.
VER     = 0.
VTL     = 0.
VTR     = 0.
VYL     = 0.
VYR     = 0.
VZL     = 0.
VZR     = 0.
FLUXL   = 1024.
FLUXR   = 0.
ENPTY   = 0
INJECT  = -1
$
```

E-09
ELECTRIC FIELD
TIMESTEP = 0 TIME = 0.0000

E-07
ELECTRIC POTENTIAL
TIMESTEP = 0 TIME = 0.0000

VX VS.X SPECIES 1
TIMESTEP = 100 TIME = 0.7773

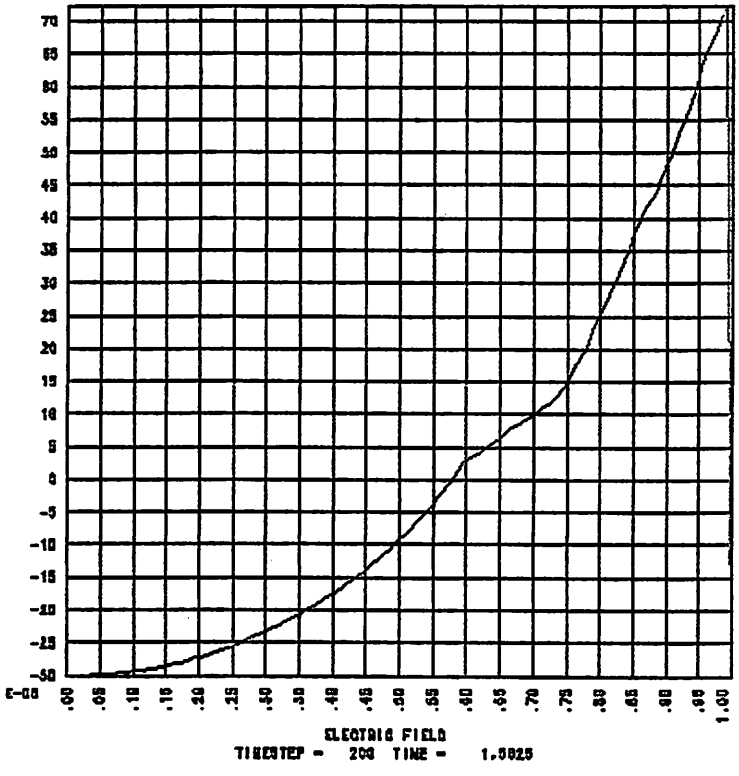
TOTAL CHARGE DENSITY
TIMESTEP = 100 TIME = 0.7812

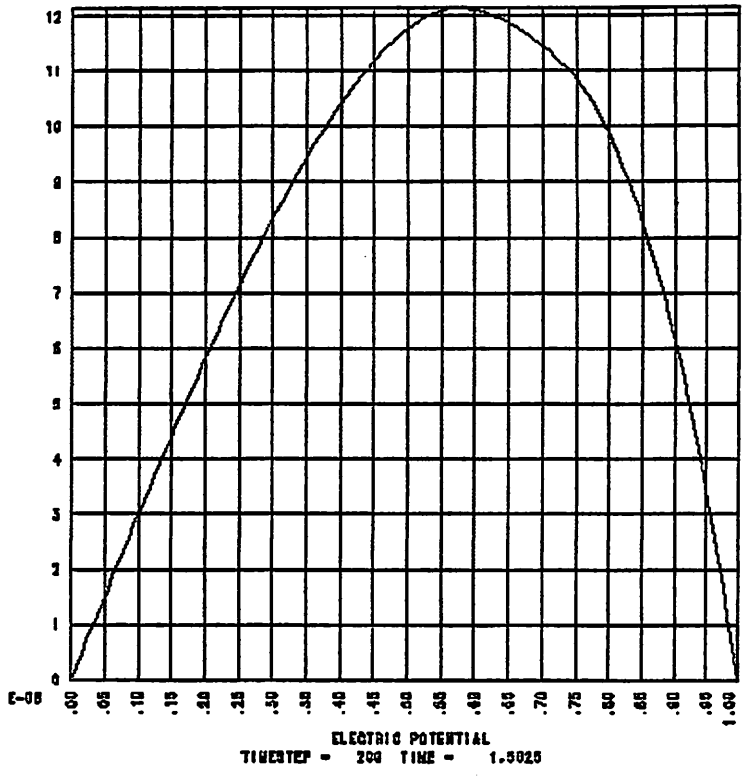
ELECTRIC FIELD
TIMESTEP = 100 TIME = 0.7512

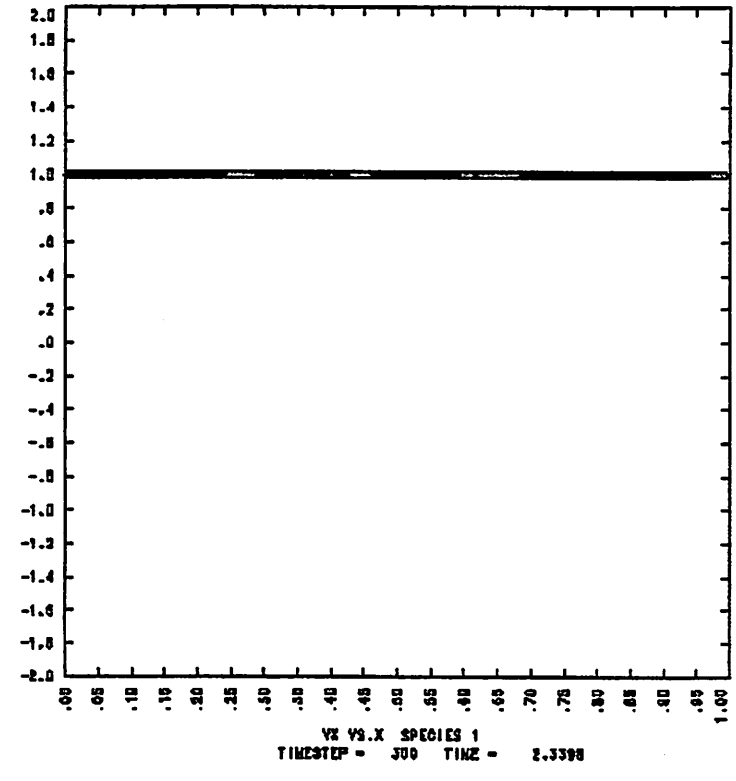
ELECTRIC POTENTIAL
TIMESTEP = 100 TIME = 0.7512

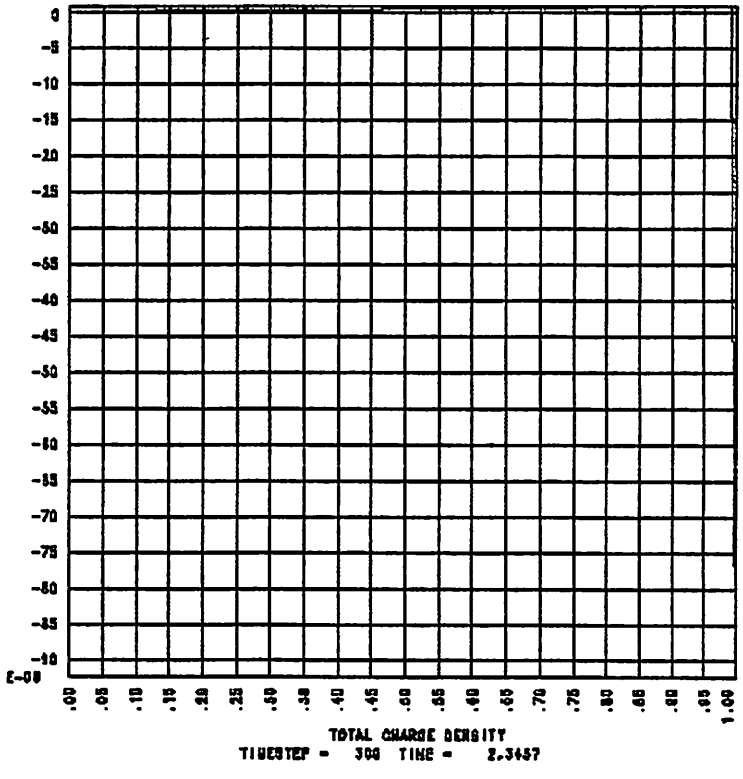
VX VS.X SPECIES 1
TIMESTEP = 200 TIME = 1.5569

TOTAL CHARGE DENSITY
TIMESTEP = 200 TIME = 1.5925


E-08
ELECTRIC FIELD
TIMESTEP = 200 TIME = 1.5025


E-08
ELECTRIC POTENTIAL
TIMESTEP = 200 TIME = 1.5025


VX VS.X SPECIES 1
TIMESTEP = 300 TIME = 2.3398


E-08
TOTAL CHARGE DENSITY
TIMESTEP = 300 TIME = 2.3457

E-08
ELECTRIC FIELD
TIMESTEP = 300 TIME = 2.3457

E-08
ELECTRIC POTENTIAL
TIMESTEP = 300 TIME = 2.3457

VX VS.X SPECIES 1
TIMESTEP = 400 TIME = 3.1211

E-08
TOTAL CHARGE DENSITY
TIMESTEP = 400 TIME = 3.1200

E-08
ELECTRIC FIELD
TIMESTEP = 400 TIME = 3.1250

E-08
ELECTRIC POTENTIAL
TIMESTEP = 400 TIME = 3.1250

VX VS.X SPECIES 1
TIMESTEP = 800 TIME = 3.0023

E-06
TOTAL CHARGE DENSITY
TIMESTEP = 800 TIME = 3.0062

E-09
ELECTRIC FIELD
TIMESTEP = 500 TIME = 3.0062

E-10
ELECTRIC POTENTIAL
TIMESTEP = 500 TIME = 3.0062

E-07
CAPACITOR CHARGE
TIMESTEPS 0 TO 500

E-05
EXTERNAL CURRENT
TIMESTEPS 0 TO 500

NATURAL LOG SO GROWTH/DECAY RATE CAN BE READ OFF

DECAY RATE AND POTENTIAL EIGENFUNCTION AGREE WITH THEORY

BOX B04 PD1 PIERCE DIODE I

TVBDLIB.MFE.CRAY - VERSION 3.0

FR80 OUTPUT... 15:18:24 08/26/83C

0031 FRAMES PLOTTED

2. Pierce Diode — first unstable regime

Parameters:

NSMAX = 1

NGMAX = 128

NVMAX = 1

NPMAX = 2000

HYMAX = 500

Input file:

```
box bnn pierce diode II
nt=500 rhoback=16.  backj=16.  iplot=100 iout=100 isav=1 $end
j0l=16.  v0l=1.  fluxl=1024.  empty=.false.  $end
```

Comments:

Since $\pi < \omega_p L/v_0 = 4 < 2\pi$, the dominant mode is a purely growing mode.

Note that a decaying oscillatory mode can also be seen early in the simulation.

It is worth running this simulation to saturation, and then inverting the initial loading noise to observe the saturated state when the initial linear mode is of the opposite sign. The result is somewhat surprising.

E-08
ELECTRIC FIELD
TIMESTEP = 400 TIME = 3.1250

E-07
ELECTRIC POTENTIAL
TIMESTEP = 400 TIME = 3.1250

VX VS.X SPECIES 1
TIMESTEP = 500 TIME = 3.9023

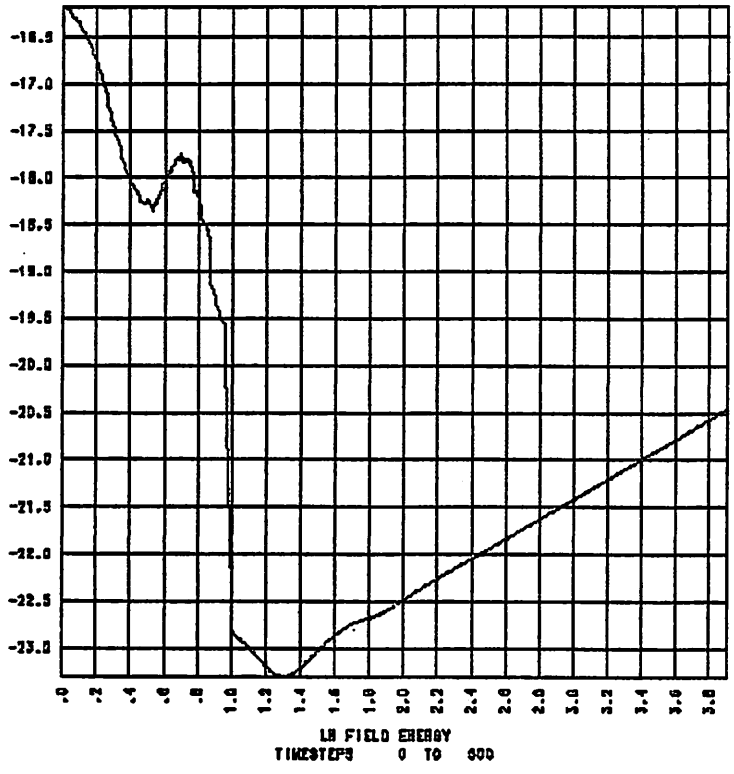
TOTAL CHARGE DENSITY
TIMESTEP = 500 TIME = 3.9062

ELECTRIC FIELD
TIMESTEP = 900 TIME = 3.0002

ELECTRIC POTENTIAL
TIMESTEP = 900 TIME = 3.0002

CAPACITOR CHARGE
TIMESTEPS 0 TO 900

EXTERNAL CURRENT
TIMESTEPS 0 TO 900



AGAIN GROWTH RATE
AGREES WITH THEORY

BOX 804 PD2 PIERCE DIODE I

TV80LIB.MFE.CRAY - VERSION 3.0

FR80 OUTPUT... 15:18:24 08/26/83C

0031 FRAMES PLOTTED

3. Pierce Diode — second unstable regime

Parameters:

NSMAX = 1

NGMAX = 128

NVMAX = 1

NPMAX = 2000

HYMAX = 500

Input file:

```
box bnn pierce diode III
nt=500 rhoback=64.  backj=64.  iplot=100 iout=100 isav=1 $end
j01=64.  v01=1.  fluxl=1024.  empty=.false.  $end
```

Comments:

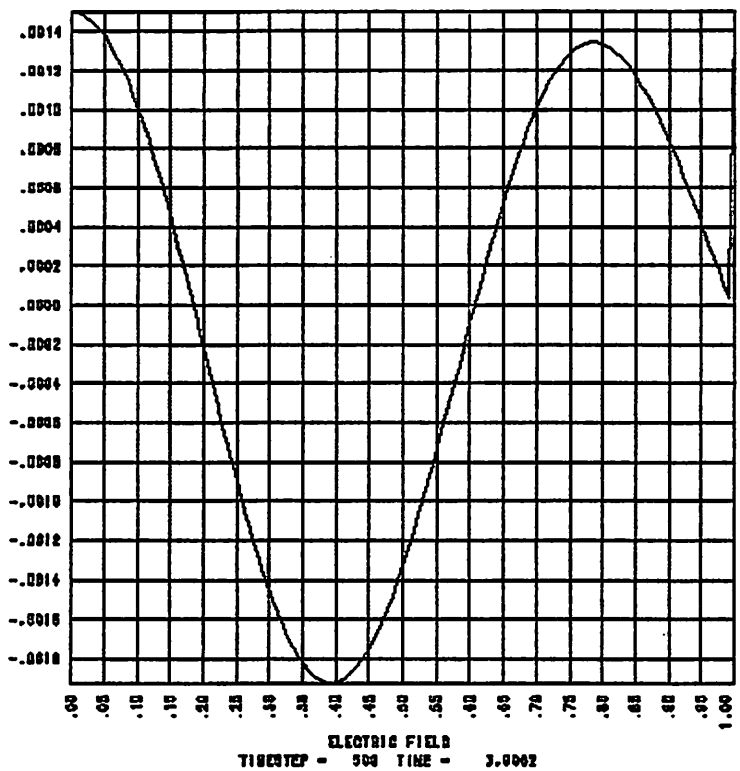
Since $2\pi < \omega_p L/v_0 = 8 < 3\pi$, the dominant mode is an oscillatory mode.

ELECTRIC FIELD
TIMESTEP = 400 TIME = 3.1290

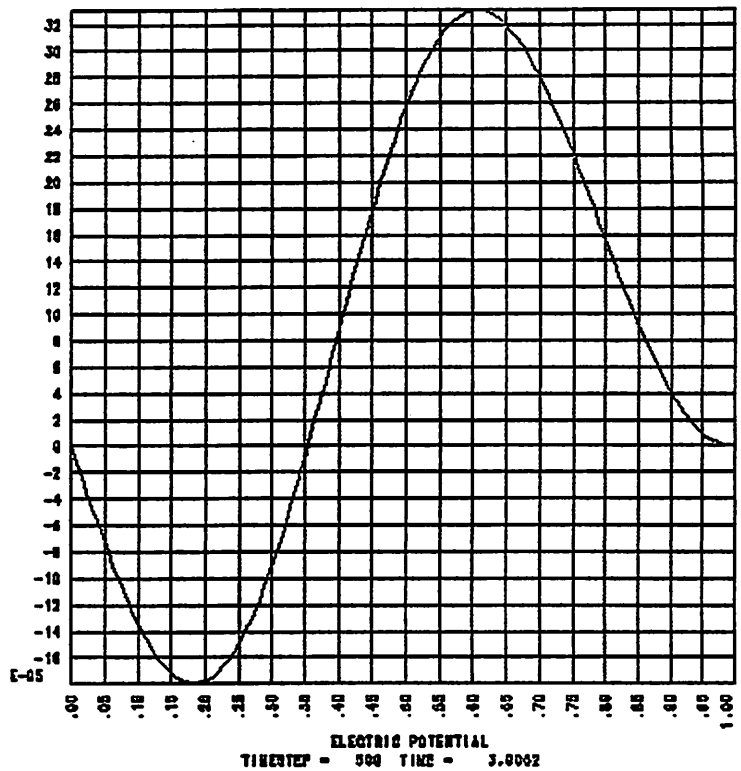
E-05
ELECTRIC POTENTIAL
TIMESTEP = 400 TIME = 3.1290

VX VS.X SPECIES 1
TIMESTEP = 399 TIME = 3.0023

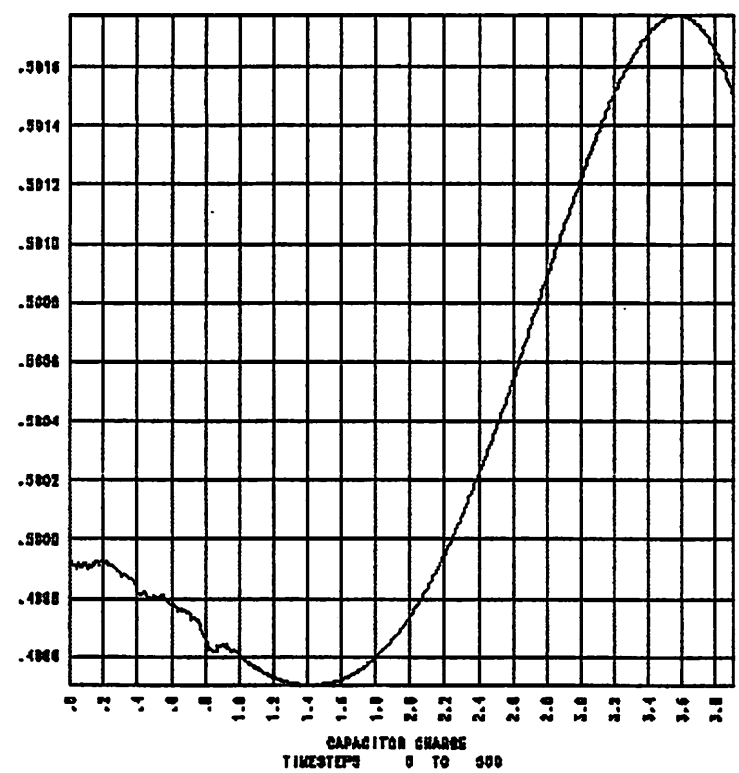
TOTAL CHARGE DENSITY
TIMESTEP = 399 TIME = 3.0062



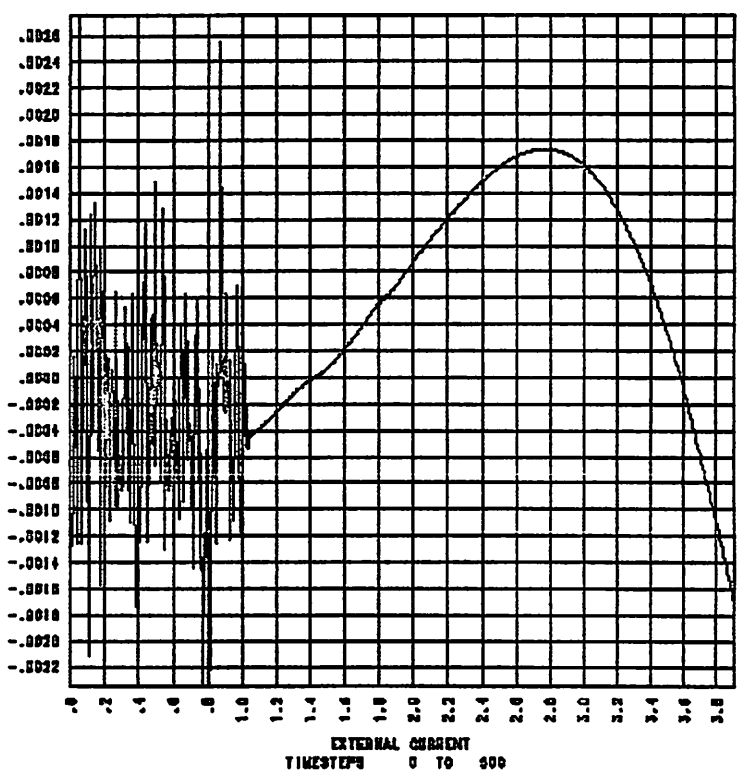
ELECTRIC FIELD
TIMESTEP = 500 TIME = 3.0002



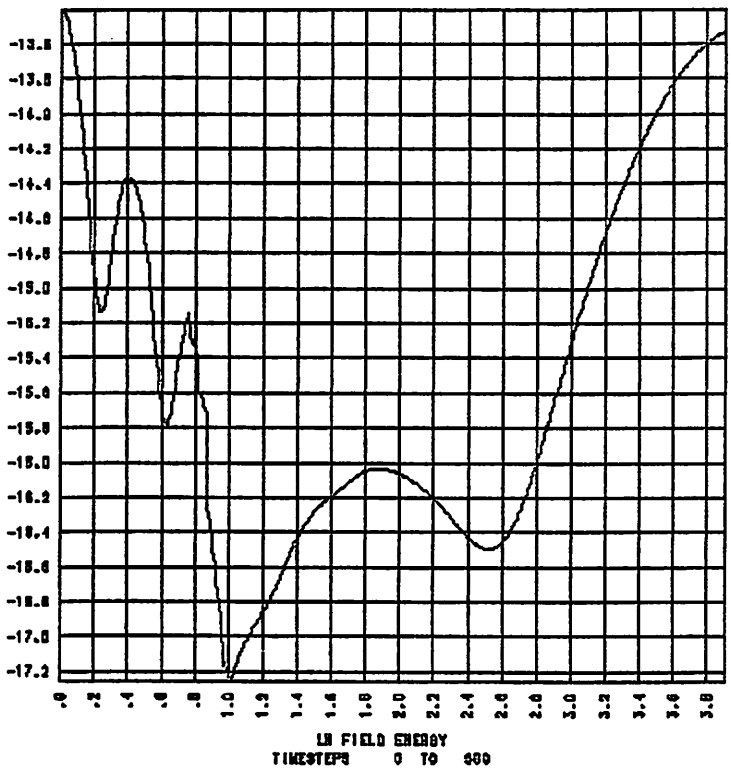
ELECTRIC POTENTIAL
TIMESTEP = 500 TIME = 3.0002



CAPACITOR CHARGE
TIMESTEPS 0 TO 500



EXTERNAL CURRENT
TIMESTEPS 0 TO 500



THIS MODE IS GROWING-OSCILLATORY, A LONGER SIMULATION REVEALS AGREEMENT WITH THEORY

BOX B04 PD3 PIERCE DIODE I

TV80LIB.MFE.CRAY - VERSION 3.0

FR80 OUTPUT... 15:20:45 08/26/83C

0031 FRAMES PLOTTED

4. Simple Plasma Injection

Parameters:

NSMAX = 1

NGMAX = 128

NVMAX = 1

NPMAX = 2000

HYMAX = 500

Input file:

```
box bnn simple plasma injection
nt=2000 length=4.  iplot=400 iout=100 isav=1 $end
j0l=40.  vtl=1.  fluxl=32768.  $end
qm=0.25 j0l=10.  vtl=0.5 fluxl=8192.  $end
```

Comments:

If you can explain *this* one, we would like to hear from you. The instability goes away if LENGTH is set to 1.

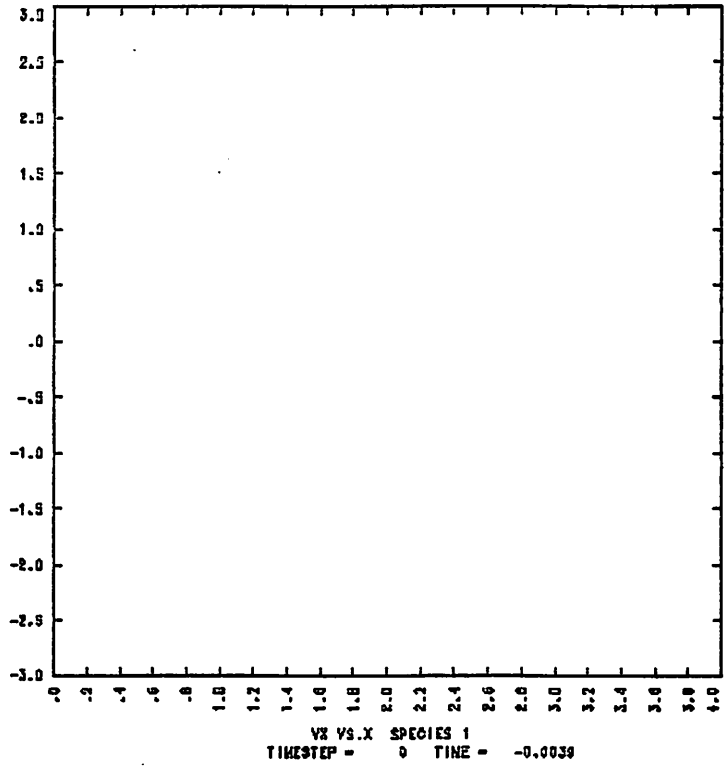
```
NSP     = 2
NG      = 512
NT      = 2000
DT      = 7.81255-03
NV      = 1
LENGTH  = 4.
AREA    = 1.
EPSO    = 1.
B       = 0.
PSI     = 0.
RNODACX = 0.
BACKJ   = 0.
EXTR    = 0.
EXTL    = 0.
EXTO    = 1.E+20
DCBIAS  = 0.
ACBIAS  = 0.
WD      = 0.
DD      = 0.
IO      = 0.
SIGMAO  = 0.
IPLOT   = 500
ICUT    = 100
ISAV    = 1
IHIST   = 400
IPACK   = 10
ZED     = 0
$

SPECIES 1
QM      = -1.
JDL     = 40.
JDR     = 0.
VDL     = 0.
VDR     = 0.
VEL     = 0.
VER     = 0.
VTL     = 1.
VTR     = 0.
VYL     = 0.
VYR     = 0.
VZL     = 0.
VZR     = 0.
FLUXL   = 32768.
FLUXR   = 0.
EMPTY   = -1
INJECT  = -1
$

SPECIES 2
QM      = 0.25
JDL     = 10.
JDR     = 0.
VEL     = 0.
VDR     = 0.
VEL     = 0.
VER     = 0.
VTL     = 0.5
VTR     = 0.
VYL     = 0.
VYR     = 0.
VZL     = 0.
VZR     = 0.
FLUXL   = 8192.
FLUXR   = 0.
EMPTY   = -1
INJECT  = -1
$
```

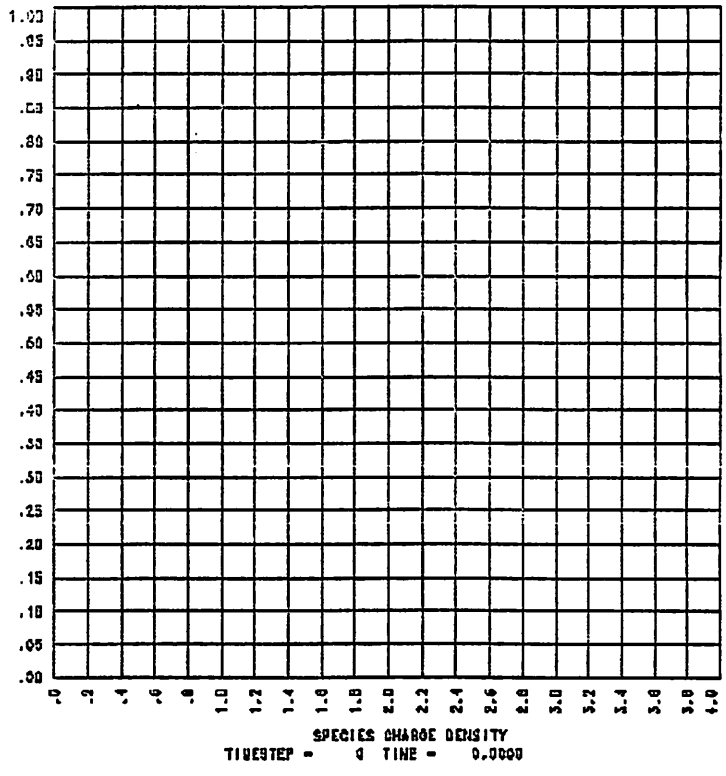
$\frac{m_i}{m_e} = 4$

$n_{i0} = \frac{n_{e0}}{2}$

(ELECTRON RICH EMISSION)

$T_i = T_e$





VX VS.X SPECIES 2
TIMESTEP = 0 TIME = -0.0030

SPECIES CHARGE DENSITY
TIMESTEP = 0 TIME = 0.0000

TOTAL CHARGE DENSITY
TIMESTEP = 0 TIME = 0.0000

ELECTRIC FIELD
TIMESTEP = 0 TIME = 0.0000

ELECTRIC POTENTIAL
TIMESTEP = 0 TIME = 0.0000

VX VS.X SPECIES 1
TIMESTEP = 400 TIME = 3.1211

SPECIES CHARGE DENSITY
TIMESTEP = 400 TIME = 3.1250

VX VS.X SPECIES 2
TIMESTEP = 400 TIME = 3.1211

SPECIES CHARGE DENSITY
TIMESTEP = 400 TIME = 3.1250

TOTAL CHARGE DENSITY
TIMESTEP = 400 TIME = 3.1250

ELECTRIC FIELD
TIMESTEP = 400 TIME = 3.1250

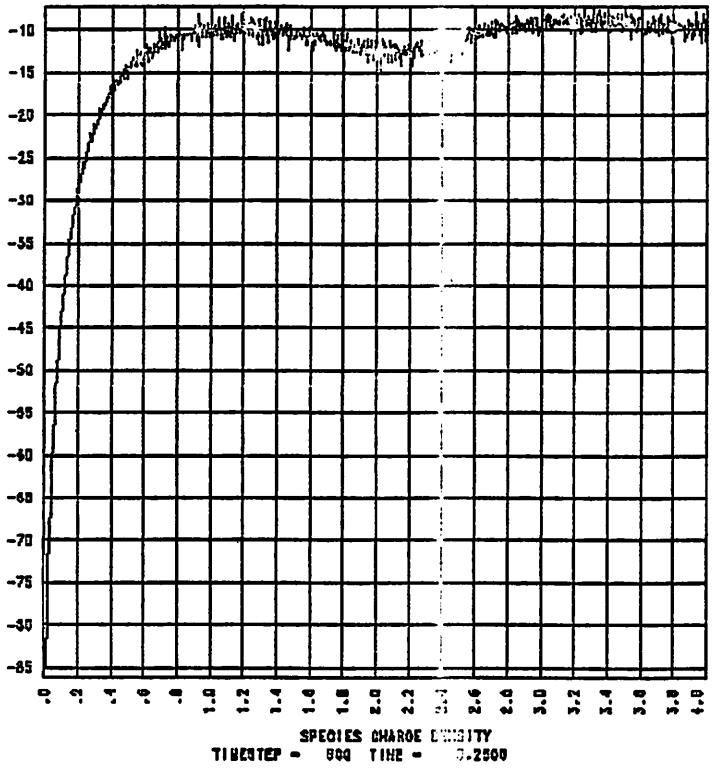
ELECTRIC POTENTIAL
TIMESTEP = 400 TIME = 3.1250

CAPACITOR CHARGE
TIMESTEPS 0 TO 400

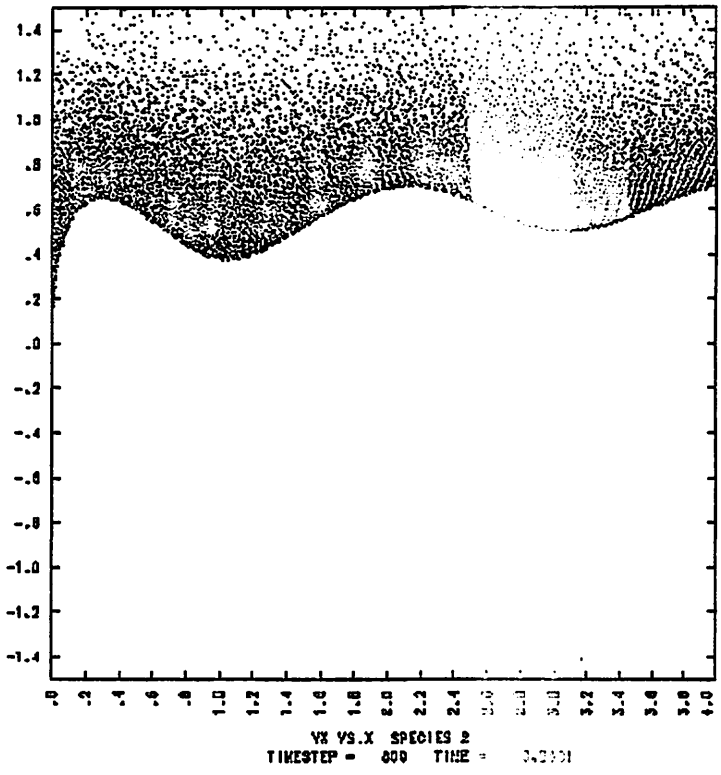
EXTERNAL CURRENT
TIMESTEPS 0 TO 400

LH FIELD ENERGY
TIMESTEPS 0 TO 400

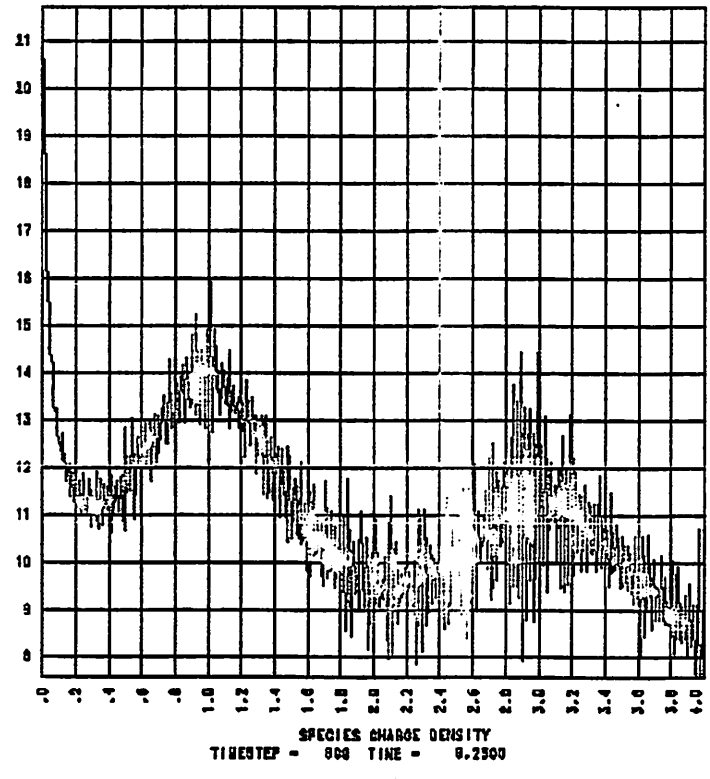
VX VS.X SPECIES 1
TIMESTEP = 400 TIME = 0.2401



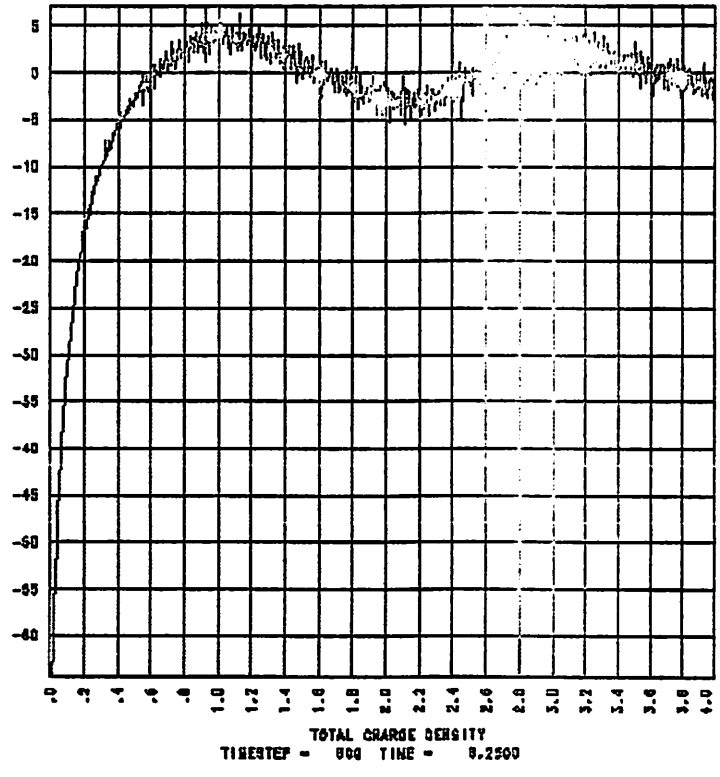
SPECIES CHARGE DENSITY
TIMESTEP = 800 TIME = 8.2500



VX VS.X SPECIES 2
TIMESTEP = 800 TIME = 8.2500



SPECIES CHARGE DENSITY
TIMESTEP = 800 TIME = 8.2500



TOTAL CHARGE DENSITY
TIMESTEP = 800 TIME = 8.2500

ELECTRIC FIELD
TIMESTEP = 800 TIME = 0.2500

ELECTRIC POTENTIAL
TIMESTEP = 800 TIME = 0.2500

CAPACITOR CHARGE
TIMESTEPS 400 TO 800

EXTERNAL CURRENT
TIMESTEPS 400 TO 800

LH FIELD ENERGY
TIMESTEPS 400 TO 800

VX VS.X SPECIES 1
TIMESTEP = 1200 TIME =

SPECIES CHARGE DENSITY
TIMESTEP = 1200 TIME = 0.3760

VX VS.X SPECIES 2
TIMESTEP = 1200 TIME = 0.3711

SPECIES CHARGE DENSITY
TIMESTEP = 1200 TIME = 0.3750

TOTAL CHARGE DENSITY
TIMESTEP = 1200 TIME = 0.3750

ELECTRIC FIELD
TIMESTEP = 1200 TIME = 0.3750

ELECTRIC POTENTIAL
TIMESTEP = 1200 TIME = 0.3750

CAPACITOR CHARGE
TIMESTEPS 800 TO 1200

EXTERNAL CURRENT
TIMESTEPS 800 TO 1200

LH FIELD ENERGY
TIMESTEPS 800 TO 1200

VX VS.X SPECIES 1
TIMESTEP = 1800 TIME = 12.4991

SPECIES CHARGE DENSITY
TIMESTEP = 1600 TIME = 12.5000

VX V9.X SPECIES 2
TIMESTEP = 1600 TIME =

SPECIES CHARGE DENSITY
TIMESTEP = 1600 TIME = 12.5000

TOTAL CHARGE DENSITY
TIMESTEP = 1600 TIME = 12.5000

ELECTRIC FIELD
TIMESTEP = 1600 TIME = 12.0000

ELECTRIC POTENTIAL
TIMESTEP = 1600 TIME = 12.0000

CAPACITOR CHARGE
TIMESTEPS 1200 TO 1600

EXTERNAL CURRENT
TIMESTEPS 1200 TO 1600

LH FIELD ENERGY
TIMESTEPS 1200 TO 1300

VX VS.X SPECIES 1
TIMESTEP = 2000 TIME = 15.0211

SPECIES CHARGE DENSITY
TIMESTEP = 2000 TIME = 15.0250

VX VS.X SPECIES 2
TIMESTEP = 2000 TIME = 15.0211

SPECIES CHARGE DENSITY
TIMESTEP = 2000 TIME = 15.02.0

TOTAL CHARGE DENSITY
TIMESTEP = 2000 TIME = 15.0200

ELECTRIC FIELD
TIMESTEP = 2000 TIME = 15.0200

ELECTRIC POTENTIAL
TIMESTEP = 2000 TIME = 15.0200

CAPACITOR CHARGE
TIMESTEPS 1600 TO 2000

EXTERNAL CURRENT
TIMESTEPS 1600 TO 2000

LH FIELD ENERGY
TIMESTEPS 1600 TO 2000