A METHOD FOR SIMULATION OF VLASOV SYSTEMS WITH DISCONTINUOUS DISTRIBUTION FUNCTIONS

and

GASBAG USER'S MANUAL

by

William S. Lawson

Memorandum No. UCB/ERL M84/24

March 1984

Research sponsored by U.S. Dept. of Energy
Contract DE-AT03-76ET53064; computing done
at National Magnetic Fusion Energy Computer Center
Lawrence Livermore National Laboratory, Livermore
California

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A Method for Simulation of Vlasov Systems with Discontinuous Distribution Functions

William S. Lawson

Electronics Research Laboratory, University of California, Berkeley
Berkeley, Ca. 94720

ABSTRACT

A method is presented for simulating non-periodic (or periodic) Vlasov systems with discontinuous distribution functions. The method is a hybrid of a standard Vlasov method and the Waterbag method; the discontinuity is integrated separately from the distribution function. Sample runs and particle simulation comparison runs are included.

Introduction

Most simulations of plasma systems have, until recently, assumed periodic boundary conditions. While this is satisfactory for some problems, there is a growing interest in the simulation of problems having non-periodic boundary conditions. Particle simulation methods have been adapted to these problems [1], but particle methods are inherently noisy. Finite difference grid methods for simulating the Vlasov equation, on the other hand, are not noisy, but have been thought to be unsuitable for non-periodic problems because of the discontinuities in the Vlasov distribution function which are inherent in the non-periodic boundary conditions. Presented here is a method of surmounting the difficulties inherent in using a finite difference grid method to simulate a non-periodic Vlasov system. The method is not as flexible as a particle method, but the accuracy is comparable, and the particle shot noise is eliminated.

Problem Definition and Approach

The problem to be solved is the integration of the one-dimensional Vlasov-Poisson system in time over a spatial domain of length L, and an infinite velocity domain. Specifically, to solve

$$\frac{\partial f_s}{\partial t} + v\frac{\partial f_s}{\partial x} + \frac{q_s E(x,t)}{m_s}\frac{\partial f_s}{\partial v} = 0 \tag{1}$$

and

$$\frac{\partial^2 \phi}{\partial x^2} = -\frac{1}{\epsilon_0}\sum_s \int_{-\infty}^{\infty} f_s(x,v,t)\,dv \tag{2}$$

where

$$E = -\frac{\partial \phi(x,t)}{\partial x}$$

with appropriate boundary conditions on ϕ. (Note that the charge of each species is absorbed into the distribution function.) For simulation purposes, the velocity domain is cut off at $\pm v_{max}$. The boundary conditions on the distribution function used are that the incoming distributions at $x = 0$, $v > 0$, and $x = L$, $v < 0$, are time-independent (or at worst be slightly perturbed around some time-independent distributions). The potential difference between $x = 0$ and $x = L$ is also taken to be time-independent (or slightly perturbed). The reasons for these rather strict boundary conditions will be explained shortly.

To simplify the explanation of the method, only one species is considered here, and the incoming distribution function at $x = L$ is taken to be zero (i.e. no particles coming from the $x = L$ side). The main simulation difficulty is numerical, and arises because the distributions entering from the two ends are independent of each other, and will therefore almost always give rise to a discontinuity in $f(x,v)$ where they meet. In Figure 1, for example (a particle simulation of an electron diode), the distribution function jumps from a large value above a certain characteristic, to zero below it. If a finite difference method were used on this distribution function directly, the discontinuity would wreak havoc. Depending on the method used, the discontinuity would either be diffused out over many grid cells, or cause overshoots resulting in large negative values for the distribution function. This difficulty with the discontinuity is solved by extending the distribution function smoothly over the entire x-v domain, and following the position of the discontinuity separately. Since the distribution function is now smooth, ordinary finite difference methods can be used. The extended part of the distribution function is then ignored in calculating the charge density, and so does not affect the physics.

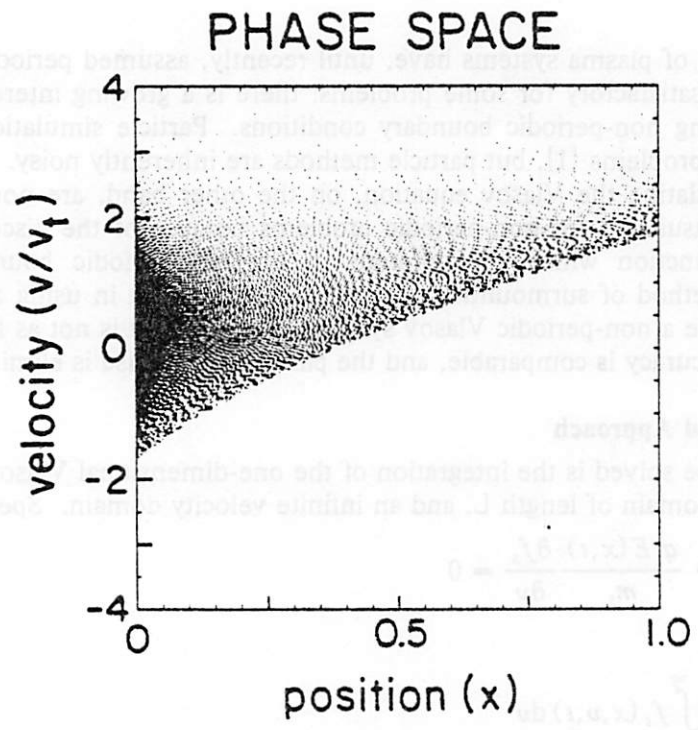

Figure 1. Phase space plot of electrons in a single species diode with space-charge limited current. Distribution function jumps suddenly to zero.

To ensure that no large phase space gradients form in this extended distribution function — defeating the purpose of the extension — the boundary condition on the artificial part of the distribution function at $x = L$ must be chosen carefully. The simplest boundary conditions are a fixed voltage difference ϕ between $x = 0$ and $x = L$, and fixed incoming distribution functions, with the additional constraint that

$$f(L,-v) = f(0,\sqrt{v^2 - 2q\phi/m}) \quad \text{for } v^2 \geq 2q\phi/m \tag{3}$$

The incoming distributions must, of course, be smooth. These conditions guarantee that the extended distribution will be continuous everywhere for the steady state. One must then hope that the time dependent case is not too badly behaved. (It will be seen that some bad behavior can be tolerated.) For the specific boundary conditions chosen for the sample implementation (half-Maxwellian at both ends) condition (3) can be written

$$f(L,-v) = f(0,v)\exp(-q\phi(L)/kT). \tag{4}$$

The position of the discontinuity is followed by a string of test particles, as in the Water-bag model [2]. When two test particles drift too far apart, a new one is created between them. The entire scheme is diagrammed in Figure 2.

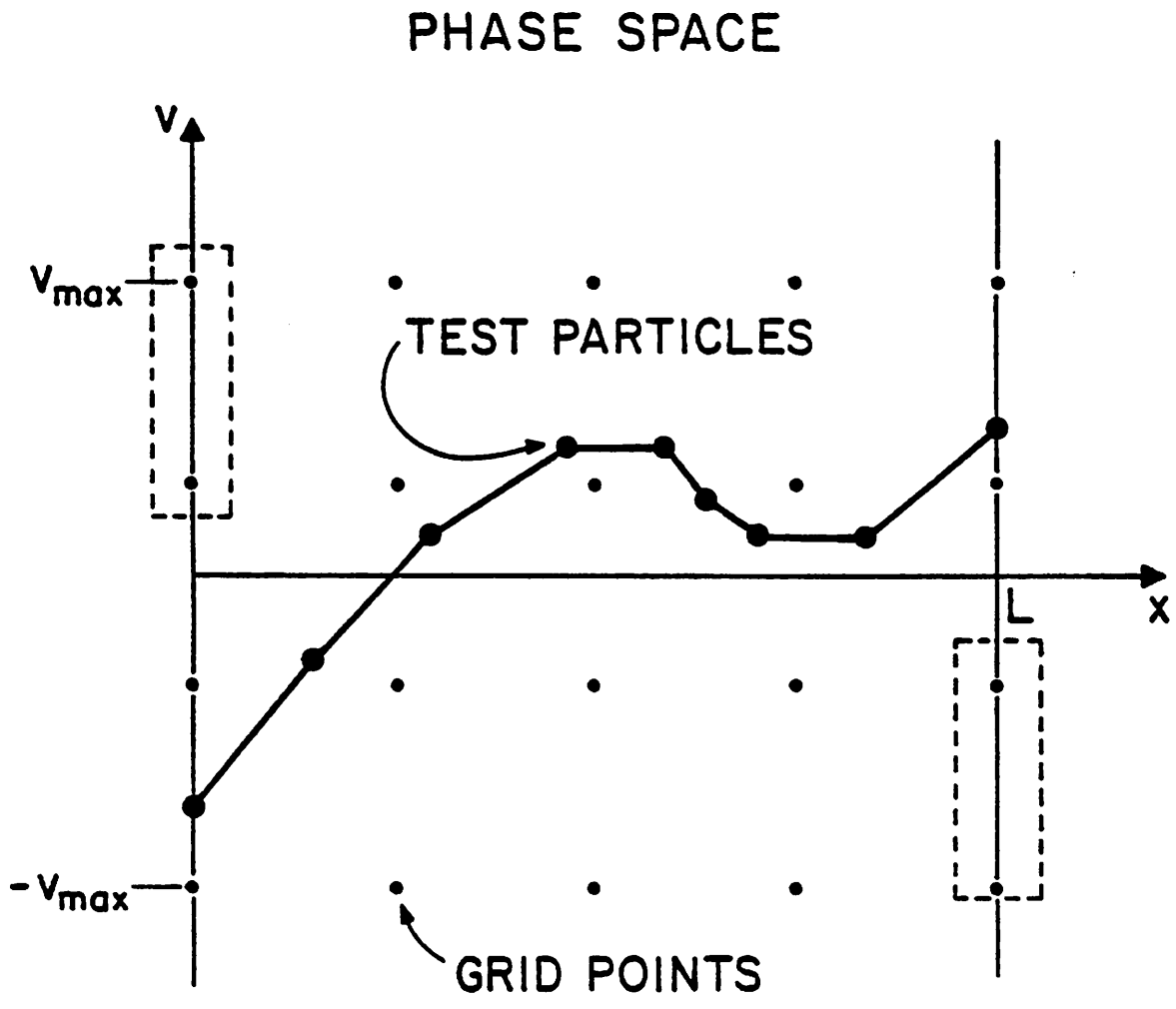

Figure 2. Grid layout and test particle scheme. Incoming distribution function is specified at grid points in dashed boxes.

Experience with the sample code has shown some pitfalls and limitations which seem obvious in retrospect. For instance, in many situations, despite choosing good boundary conditions, large gradients in phase space develop as time progresses. These cannot be dealt with in the same way as the test particles deal with the primary discontinuity, so they will give rise to numerical diffusion, and thus be a source of inaccuracy. There appears to be little that can be done about this. Another problem arises when one wants to vary the voltage across the domain in time. In order to prevent large gradients in phase space of the kind just mentioned, it is necessary to vary the incoming boundary conditions at $x = L$, but how to vary them so as to minimize the gradients in phase space is unclear. Some of these gradients (ripples for instance) may be diffused away harmlessly, but others may cause unacceptable inaccuracy.

Sample Implementation

The Vlasov equation integrator used in this implementation of the method is that used by Cheng and Knorr [3]. It is completely analogous to the leap-frog particle mover:

$$f^{p+1/2}(x,v) = f^{p}\left(x, v - \frac{qE^{p}(x)}{m}\Delta t\right) \tag{5}$$

$$f^{p+1}(x,v) = f^{p+1/2}(x - v, v\Delta t) \tag{6}$$

where $f(x,v)$ is the Vlasov distribution function, p is the time index, and x, v, and E are the normalized position, velocity, and electric field respectively. This scheme yields second order accuracy in time, and reduces the integration to two interpolations, which can be performed by any of several methods.

Many interpolation methods were tried, and it was discovered (empirically) that some numerical diffusion was necessary for stability. The best methods are those which have diffusion only when large phase-space gradients occur. When there is insufficient numerical diffusion, these large gradients apparently interact with the x-v grid to produce a numerical instability. Large phase space gradients also cause accuracy problems because of the inaccuracy of the test particle integrator. A minor shift in the position of the test particles near such a gradient can result in a major error when the charge density is computed. Introducing some numerical diffusion at steep gradients solves both these problems with a minimum error in other quantities, such as current and kinetic energy.

The interpolation schemes finally settled on are two, three, and four point *off-center* interpolations, and are shown below. They are first, second, and third order accurate respectively.

$$f_l^{p+1} = \begin{cases} f_l^p - u\left(f_{l+1}^p - f_l^p\right) & \text{if } u < 0 \\ f_l^p - u\left(f_l^p - f_{l-1}^p\right) & \text{if } u \geqslant 0 \end{cases} \tag{7}$$

$$f_l^{p+1} = \begin{cases} f_l^p - u\left(2f_{l+1}^p - \frac{3}{2}f_l^p - \frac{1}{2}f_{l+2}^p\right) + \frac{u^2}{2}\left(f_{l+2}^p - 2f_{l+1}^p + f_l^p\right) & \text{if } u < 0 \\ f_l^p - u\left(\frac{3}{2}f_l^p - 2f_{l-1}^p + \frac{1}{2}f_{l-2}^p\right) + \frac{u^2}{2}\left(f_l^p - 2f_{l-1}^p + f_{l-2}^p\right) & \text{if } u \geqslant 0 \end{cases} \tag{8}$$

$$f_l^{p+1} = \begin{cases} f_l^p - u\left(f_{l+1}^p - \frac{1}{2}f_l^p - \frac{1}{3}f_{l-1}^p - \frac{1}{6}f_{l+2}^p\right) + \frac{u^2}{2}\left(f_{l+1}^p - 2f_l^p + f_{l-1}^p\right) \\ \qquad - \frac{u^3}{6}\left(f_{l+2}^p - 3f_{l+1}^p + 3f_l^p - f_{l-1}^p\right) & \text{if } u < 0 \\ f_l^p - u\left(\frac{1}{3}f_{l+1}^p + \frac{1}{2}f_l^p - f_{l-1}^p + \frac{1}{6}f_{l-2}^p\right) + \frac{u^2}{2}\left(f_{l+1}^p - 2f_l^p + f_{l-1}^p\right) \\ \qquad - \frac{u^3}{6}\left(f_{l+1}^p - 3f_l^p + 3f_{l-1}^p - f_{l-2}^p\right) & \text{if } u \geqslant 0 \end{cases} \tag{9}$$

where $u = v\Delta t/\Delta x$. These formulae are derived simply by fitting straight lines, parabolas, and cubic curves through the given points. In practice, the three point scheme seems to be most economical.

It is also worth mentioning the methods which were tried which did not work. Derivative methods (linear interpolations with the slope of the line determined by approximating the derivative) were tried for two, three, and four point approximations, and were found to be unstable. This is not surprising, since they are strictly unstable in the periodic or infinite grid case. Also tried were a centered three point interpolation, and a cubic spline interpolation. The reasons for the failure of these methods are less clear. These methods both failed because odd-even modes developed in regions where the distribution function was becoming very steep. The failure of the cubic spline is probably due to too little diffusion. The odd-even mode appeared at exactly the place where diffusion should have been occurring to remove detail too small for the grid to resolve. Since the four point interpolation method should be more accurate than the three point centered method, the failure of the three point centered method is probably not due to insufficient diffusion, but to its symmetry in a fundamentally asymmetric problem. When the method is used, information can flow both ways with equal ease on the numerical grid, whereas the solution being modeled allows information to flow in only one direction. This is admittedly an insufficient explanation of the failure of the three point centered method, but it is a useful hypothesis for future investigation.

The test particle integrator is the usual leap-frog integrator:

$$v^{p+\frac{1}{2}} = v^{p-\frac{1}{2}} + \frac{qE^p(x)}{m}\Delta t \tag{10}$$

$$x^{p+1} = x^p + v^{p+\frac{1}{2}}\Delta t \tag{11}$$

It is this particle integrator which dictates the use of the splitting scheme for the Vlasov integrator. The test particles must stay in step with the distribution function, or else significant systematic errors will result. A different Vlasov integrator could be used, but it might be difficult to decide what particle integrator would be appropriate. This combination of particle and Vlasov integrators is simple, fast, and second order accurate in time.

As time passes, the test particles on the discontinuity will generally drift apart, and new ones must be created to maintain detail. The method used for positioning these new particles in the sample implementation is linear bisection between those particles which have drifted too far apart. A more sophisticated method could be used, such as fitting an ellipse through a set of four particles, but the time and complexity required make this unattractive; care must be taken to make sure that a change of scale will not produce different results, since the x and v axes have different units. It should also be possible to find some curvature criterion for introducing new points which would introduce new particles where they would be most needed.

The charge integrator, which integrates the distribution function in v from the boundary of test particles to v_{max}, uses the trapezoidal rule. The lower limit of integration is taken to be the crossing of the line of test particles with any given x grid position. While the maximum spacing of the test particles could be any distance, the charge integration routine is simplified if the particles are never more than one x-grid cell apart; so, in the sample code, the maximum allowed distance between particles is precisely the grid spacing.

Because of the non-periodic boundary conditions and the limit in velocity, the boundary conditions deserve some comment. At the outgoing boundaries ($x = 0$, $v < 0$ and $x = L$, $v > 0$) there are no problems, since all the interpolation methods are off-center in the upstream direction; however, at the incoming boundaries ($x = 0$, $v > 0$ and $x = L$, $v < 0$), an interpolation method cannot be used where it would need points outside the domain. Therefore, a less accurate interpolation method must be used at these boundaries. The effect of this can easily be seen in the jump in the ballistic current diagnostic at the left hand edge in the sample runs (Figure 3). This effect is certainly removable if it is a problem. An extra column of grid points, either used as a guard cell or placed between the first two columns of grid points, would probably smooth out the jump.

The velocity limit at $v = \pm v_{max}$ presents an entirely different problem, and has no definitive solution. In the sample code, interpolations which require grid points outside the velocity domain assume an exponential fall-off. This method is apparently quite successful, and all but guarantees that the distribution function remains positive. It breaks down when any significant dynamics reaches the high velocity boundary, but in this case the problem must be rerun with a larger v_{max} anyway.

Sample Runs

The results of a sample run with a single particle species are shown in Figure 3. The domain is initially vacuum, and the ends of the domain are shorted. The single species (electrons) is injected from the left end starting at $t = 0$, and they rapidly fill the simulation domain. $\omega_p \Delta t$, which is a function of position, is much less than 1 throughout the simulation. The diagnostics shown are contour plots of phase space (the heavy line is the discontinuity marked by the test particles — the distribution below this line is non-physical), charge density, and ballistic current density. This simulated diode rapidly comes to a steady state with no overshoot, and no oscillation of the potential minimum, in contrast to particle simulations of similar problems [1,4,5]. This indicates that the oscillations observed in particle simulations are noise driven oscillations, and not an instability.

The steady state solution for this problem can be accurately calculated numerically [4]. The steady-state current, according to theory, should be 9.116 throughout the simulation domain, so the current diagnostic represents a very sensitive test of the accuracy of the method. As can be seen from the final current density plot in Figure 3, the simulation agrees with the theory to within 0.4%. The steady state electrostatic potential (not shown) agrees even more closely than the current density.

The plots in Figure 3 also point out the necessity for some numerical diffusion. The exact Vlasov solution of this problem contains a wall in the distribution function at the discontinuity (not to be confused with the discontinuity itself), which would go to zero thickness without ever declining in height. Naturally a finite grid simulation method cannot cope with this kind of detail, and any interpolation method which does not have some diffusion when gradients in phase space become large must lead to a numerical instability. (Cubic spline interpolation is unstable, presumably for this reason.) Furthermore, this wall becomes physically uninteresting beyond a certain point, as the Vlasov equation is spatially averaged to begin with, so only a certain scale of detail is relevant. The diffusion allows the wall to fade away with a minimum of inaccuracy.

While the grid cannot support any detail smaller than the grid spacing, the string of test particles can. Thus, even though filamentation within the gridded distribution function is smoothed out, the filamentation of the discontinuity is still observable. This property could conceivably prove to be useful.

The speed of this method is comparable to the speed of particle methods if one considers grid points to be roughly equivalent to particles. The test run shown was run on a 129×128 grid for 1600 time steps. The total time for the run was 30 seconds of CRAY time. The time for moving and maintaining the test particles on the discontinuity is about equal to the time required to advance the distribution function when there are 64 grid values of velocity.

Comparison runs were made using the newly developed particle code PDW1 (PDW is an acronym for Plasma Device Workshop, a workshop at U.C. Berkeley during the spring quarter of 1983 for which the code was written). The x-grid has 128 grid cells, and the number of particles in the particle simulation is roughly three quarters of the number of grid points in the Vlasov simulation. PDW1 is fully vectorized (running at roughly 2 μsec/particle/timestep), yet took over 100 seconds for this comparison run. The results of the particle run are shown in Figure 4, and are in excellent agreement with the Vlasov simulation. The inaccuracy of the Vlasov simulation is, however, smaller than the noise level of the particle scheme, despite using a quiet loading scheme (bit-reversing) for the particle run. This is a major advantage in tests of linearized theory in which one wants to look at small perturbations of a steady state.

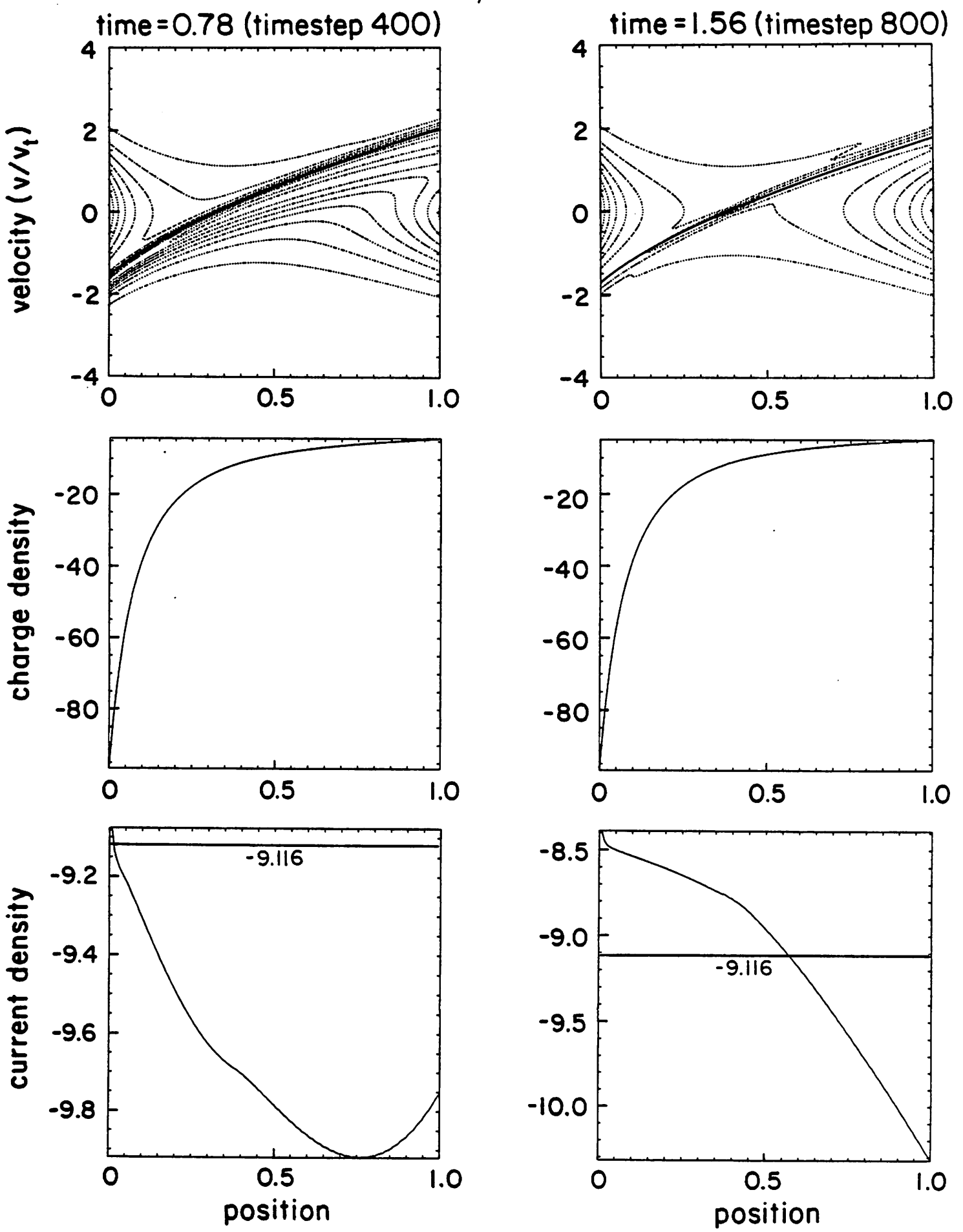

Figure 3a. Results of a sample run.
Steady State current should be a constant 9.116
according to Langmuir's theory.

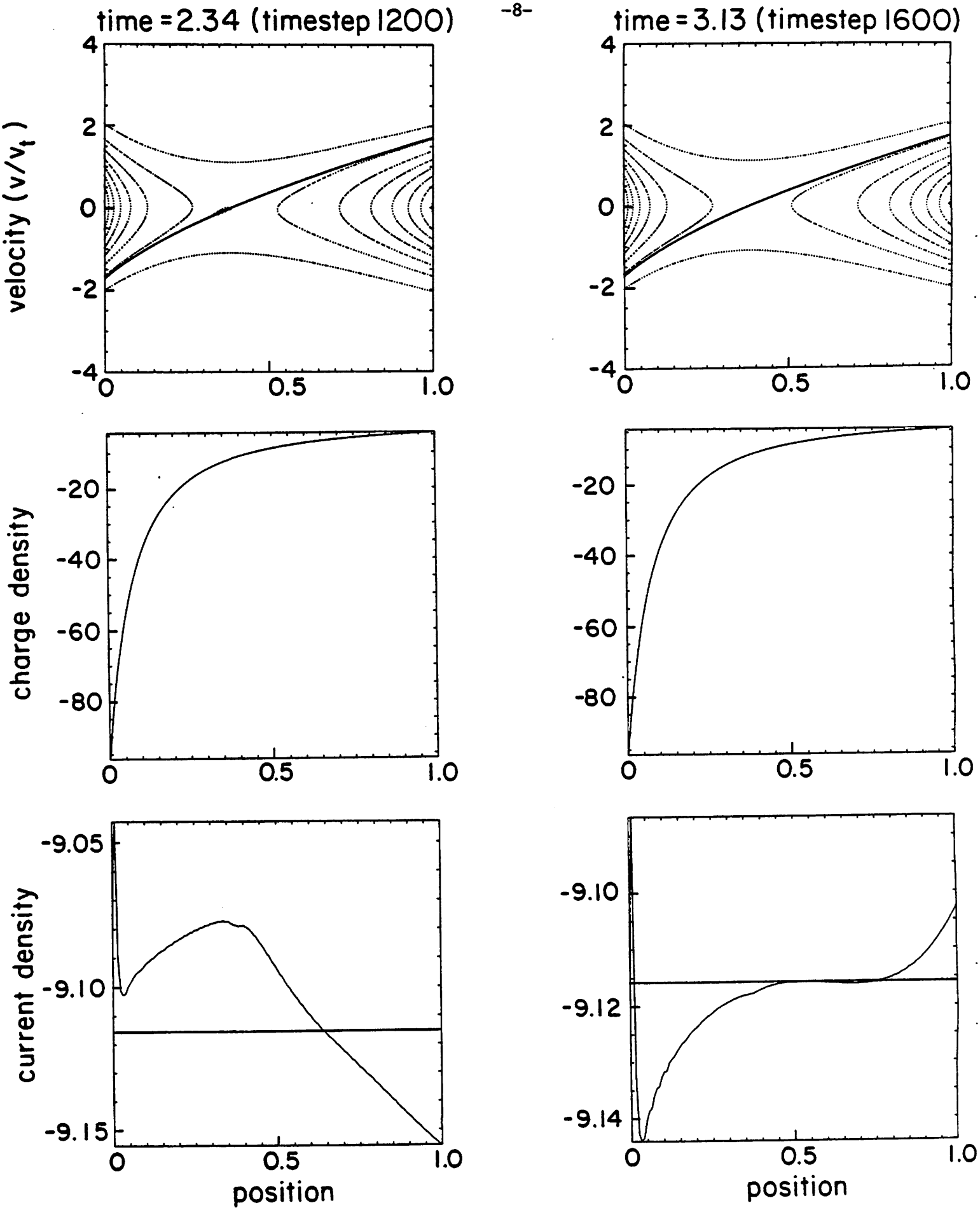

Figure 3b. Results of a sample run
(continued)

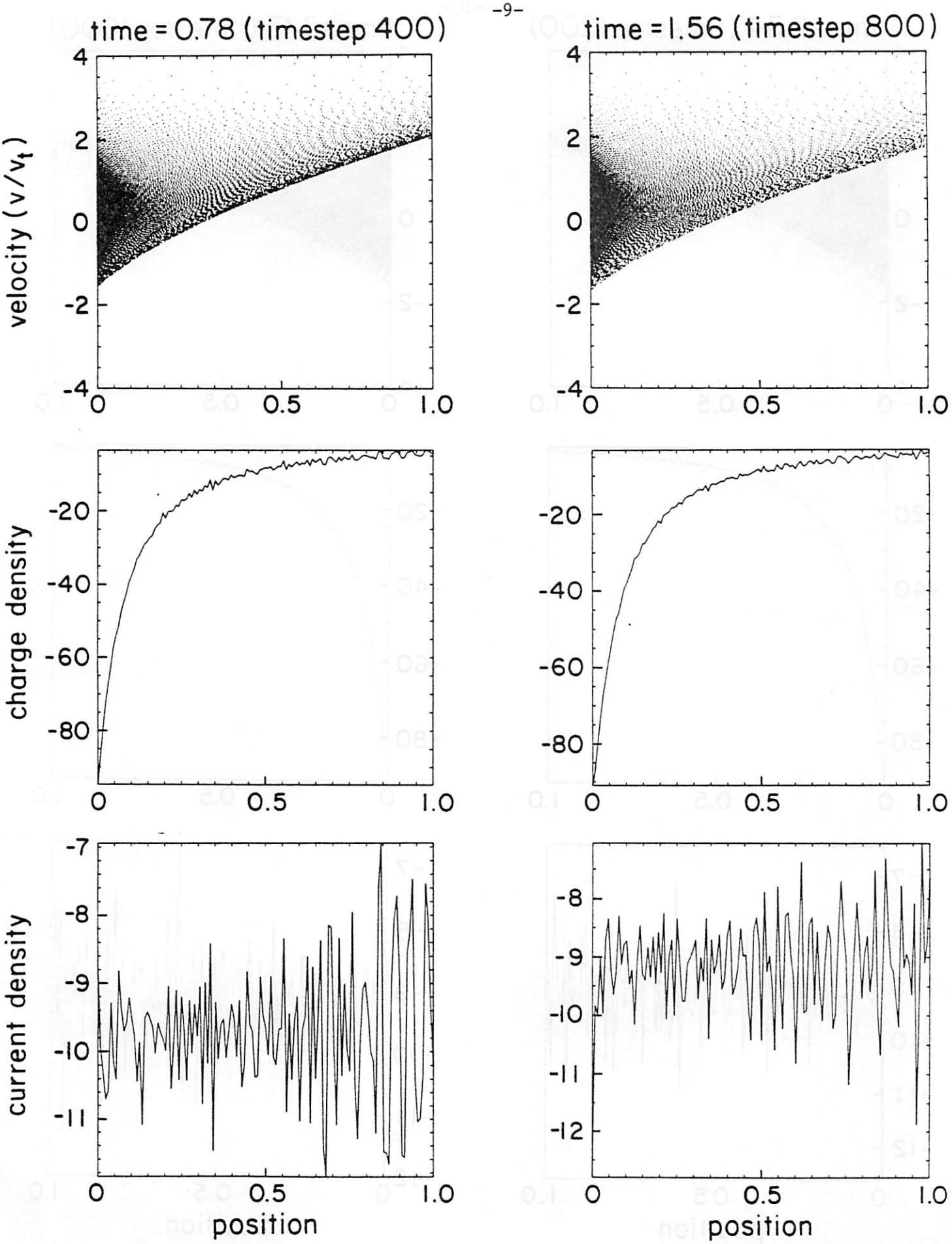

Figure 4a. Particle code run of problem in figure 3, for comparison.
(Note that the noise level of the final current density is
much higher than the inaccuracy of the GASBAG result.)

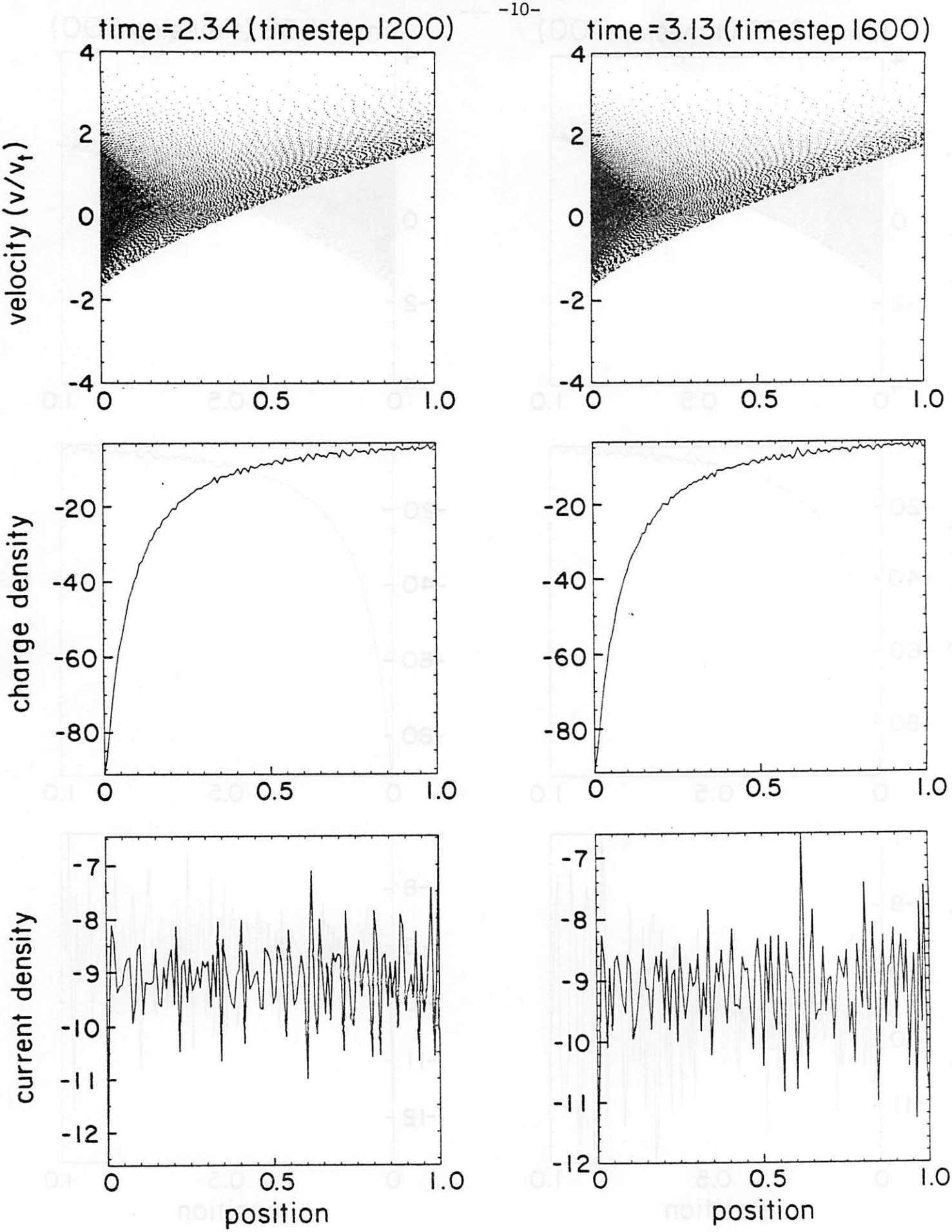

Figure 4b. Particle code run
(continued)

Figure 5 is a comparison of the total kinetic energy histories of the partiacle and Vlasov runs. Here the particle noise is drastically reduced (since all of the particles are summed over), and a systematic deviation can be seen. The particle code is probably most accurate here, with the inaccuracy of the Vlasov method stemming from the numerical diffusion near the potential minimum.

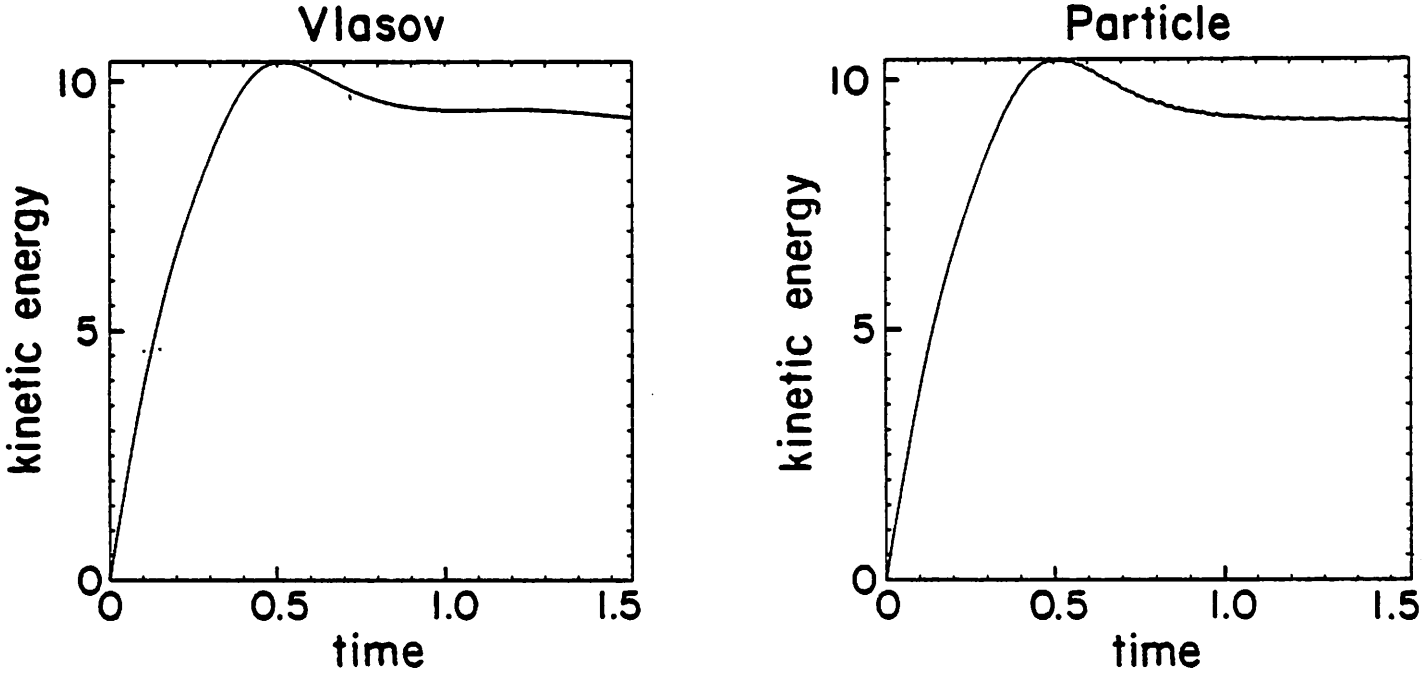

Figure 5. Comparison of kinetic energy histories for both GASBAG and PDW1. Note the 3% difference between times .7 and 1.5.

Some runs have been made with two species, both with the Vlasov code, and the particle code. Figure 6 shows a pair of such runs for a thermal plasma which enters from the left hand wall into a vacuum. The mass ratio is $m_i/m_e = 4$, and the external circuit is a short. Both simulations show an instability which starts with a wavelength roughly the length of the simulation, and then shrinks accordion-style to the final state shown with roughly three wavelengths in the simulation. At this point, the distribution breaks, forming long filaments around holes in phase space. Particle simulations run for longer times show that these filaments become very convolved; however, only a relatively small number of particles are involved, and they have little effect on the electric field. The agreement between the two codes is not perfect, and many more grid points were needed to get the Vlasov result (513 × 128), but the chief virtue of the Vlasov method is the low noise, and not high accuracy. If low noise is a must, then it is easier to use a fine mesh Vlasov code to get it than to use a huge number of particles.

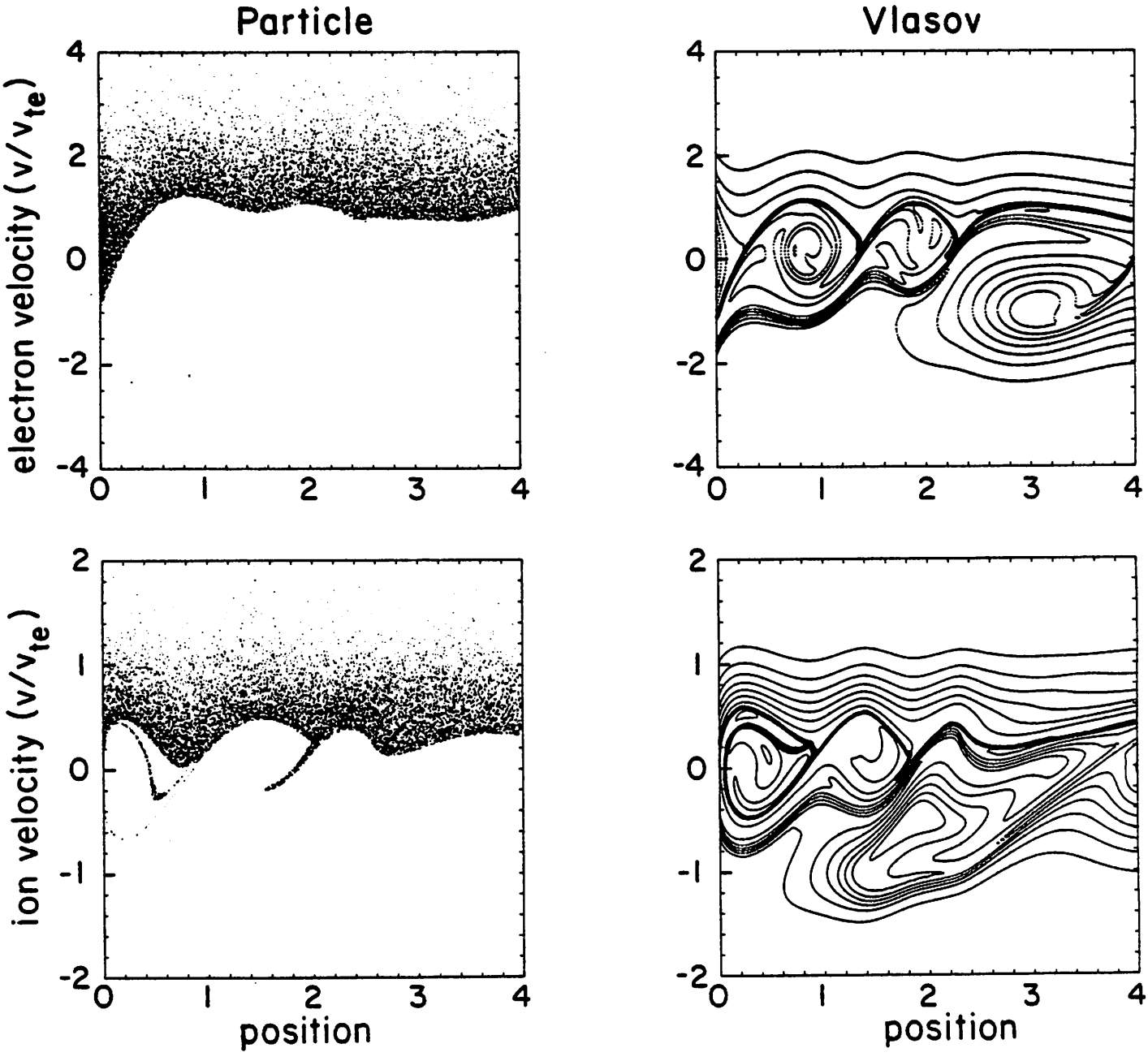

Figure 6. Phase space plots of a plasma after expanding into a box from one side, for both Vlasov and particle methods.

Summary

A method has been developed which allows the use of finite difference grid techniques to follow the evolution of non-periodic Vlasov systems having jump discontinuities. The method is numerically stable, fast, and is free of the noise problems associated with particle simulation methods. Sample runs were shown to be in excellent agreement with both theoretical results, and independent particle simulation results. The method is somewhat less flexible than particle methods with respect to boundary conditions (it requires boundary conditions which are at most slightly perturbed about some steady state), but future work may find ways of relaxing this requirement. Most importantly, it allows some problems (e.g. linearized and perturbation problems) to be solved which would be washed out by noise in a particle simulation.

The sample code, with documentation, is available from the author. It is designed to be run on the National Magnetic Fusion Energy Computer Center CRAY computers, but should be adaptable to any high speed computer.

Acknowledgments

I would like to thank Prof. C. K. Birdsall and Dr. T. L. Crystal for their valuable advice and support. This work was supported by U. S. Department of Energy contract DE-AT03-76ET53064, and the computing was performed on the CRAY computers at the National Magnetic Fusion Energy Computer Center.

References

[1] Stéphane Rousset, "Time-Dependent Child-Langmuir Diode Simulation", ERL Report No. UCB/ERL M83/39, 11 July 1983

[2] H. L. Berk and K. V. Roberts, *Phys. Fluids* **10**(1967), 1595
K. V. Roberts and H. L. Berk, *Phys. Rev. Letters* **19**(1967), 297

[3] C. Z. Cheng and Georg Knorr, *J. Comp. Phys.* **22**(1976), 330-351

[4] P. K. Tien and J. Moshman, *J. Appl. Phys.* **27**(1956), 1067

[5] C. K. Birdsall and W. B. Bridges "Electron Dynamics of Diode Regions", chapter 6, Academic Press, New York, 1966

[6] I. Langmuir, *Phys. Rev.* **21**(1923), 419

GASBAG User's Manual

William S. Lawson

Electronics Research Laboratory
University of California

ABSTRACT

A guide to use and modification of the GASBAG 1D Vlasov plasma simulation code. Also included are descriptions of the variables and subroutines for those who find it necessary to learn the entire code.

The GASBAG code simulates plasma problems with non-periodic boundary conditions using a finite grid approximation to the Vlasov equation. To do this, it introduces a string of particles to separate the part of the distribution which originated at the left wall from that which originated at the right wall, since this boundary will, in general, represent a discontinuity in the Vlasov distribution function.

GASBAG User's Manual

William S. Lawson

Electronics Research Laboratory
University of California

1. Introduction

This guide is intended to be the sole external documentation for the GASBAG Vlasov plasma simulation code. Section 2 is a guide for the casual user who needs only enough information to run the code. Section 3 is for those who intend to use the code as a research tool, and will therefore want to make modifications such as new diagnostics, and new routines for loading or perturbations.

The GASBAG code was written to be run on the National Magnetic Fusion Energy Computer Center CRAY computers. These are vector machines, and the code makes extensive use of this vector capability, as well as of some software unique to the installation. Anyone wishing to modify the code for another installation will find the information in section 4 useful.

It is assumed that the reader understands the method used by the code. Using the code without understanding the method is likely to produce bad results. For example, if the string of test particles finds its way too far up the tail of the Vlasov distribution, a small error in the position of the test particles will produce a large error in the charge density, current, and kinetic energy. A paper on the method is being prepared for submission to the Journal of Computational Physics.

2. Compiling and Running GASBAG

2.1. Input Parameters

There are five basic program parameters which can only be changed by recompiling the program. These parameters are:

NXMAX number of grid cells — must be a power of 2, preferably greater than or equal to 64 to get the most out of the CRAY vector capability

NVMAX number of velocity grid points — must also be a power of 2

NSMAX maximum number of charged species (not including background) — it generally doesn't pay to make this any larger than absolutely necessary

NPMAX maximum number of boundary points — for two or more species this must be a fairly large number, but the cost in memory is not great, so it pays to make it large

MAXHIS maximum number of points to be saved for history plots — any more than 400 is probably beyond the resolution of any plotter.

The input parameters are of two varieties: species independent and species dependent. The species independent input parameters are:

NX number of x grid cells (default NXMAX)

NV number of v grid points (not cells!) — NX and NV must be powers of 2 (default NVMAX)

L length of system (default 1)

V voltage applied to x=L side of system (default 0)

NT	number of timesteps in run
DT	length of timestep (optional — program will compute this for maximum effect if omitted)
VMAX	maximum velocity over all species (default 4.)
EPSI	dielectric constant (default 1)
NSP	number of species (default NSMAX)
IM	number code for interpolation method to be used: 1 = linear, 2 = quadratic (default), 3 = cubic
QBACK	logical flag for existence of an immobile background species (default .false.)
IFN	frequency (in timesteps) with which contour plots of the distribution function are made — as with all diagnostic frequencies, a value of zero (which is default) is a flag to omit the diagnostic
IRHO	frequency with which charge density plots are made
IJ	frequency with which current density plots are made
IE	frequency with which electric field plots are made
IPHI	frequency with which electric potential plots are made
IED	frequency with which energy density plots are made
ISAV	frequency with which history variables are saved in history array
IHST	number of points to be saved in history array before making history plots (this frequency alone has a default of MAXHIS — setting ISAV = 0 will delete the history plots)

The species dependent variables are:

I0	emission current (before any reflection) at x =0 (the only emitter) — this variable is not optional
QM	charge to mass ratio (default -1.)
VT	thermal velocity (default 1.)
VMAX	maximum velocity (optional — default = 4*VT)
EMPTY	logical flag for whether the region should be empty or filled with a half-Maxwellian at t = 0 (default .TRUE.)
IP	perturbation method: 0 = no perturbation (default), 1 = perturb all velocities at x = 0, 2 perturb only one velocity at x = 0
DF0	fraction by which distribution function is perturbed
W0	frequency of perturbation (radians per unit time — not timestep)
J0	velocity grid points above v = 0 at which perturbation is to apply

The only relevant species variables for a background species are I0, QM and VT.

2.2. Compiling on MFE computers

The COSMOS control file JGASBAG takes care of all precompiling, compiling and loading on the CRAYs. For proper compilation, JGASBAG requires the source code GASBAG, the TV80 library, and its own BGASBAG library. (BGASBAG directs the loader to tv80lib, so it is not mentioned explicitly in JGASBAG.) It will produce the usual array of object, list and map files, and an executable file named XGASBAG.

The input file for XGASBAG has a simple format on the MFE CRAYs. The first line is a header. The first eight columns of the header should be the box ID of the user. The next sixteen columns should be some comments describing the run. The rest of the file should be namelist assignments terminating each namelist with a "$end". For example:

box bnn single species
V = 2. NT = 1600 IFN = 40 ISAV = 4 $end
I0 = 40. EMPTY = .FALSE. $end

where bnn should be the box number of the user.

Note that the species independent parameters form one namelist input (ended with a "$end") and the species dependent parameters for each species will form other namelist inputs.

2.3. Compiling Elsewhere

Unless the computer installation has access to the TV80 graphics library, the plotting routines will have to be rewritten. The structure of the program should make this as painless as possible. All necessary changes should be in subroutines SETUP, SNAP, GRAPH, and FINISH.

In addition, the system calls in SETUP and FINISH must be changed. The only computational subroutine which must be supplied by a library is the fast Fourier transform subroutine CPFT. It should not be hard to find a suitable substitute for this routine (almost any good book on numerical methods contains such a routine).

The precompiler statements in GASBAG may represent another hurdle. If there is no precompiler on the installation to be used, the source code taken from the MFECC should be precompiled. This will be much more of a nuisance to alter, but it will be better than nothing.

Finally, there are several common but non-ANSI FORTRAN statements in the code. In particular, the NAMELIST statement and the structured IF statement. The NAMELIST statements are limited to the input routines START and SPECIES. The structured IF statements are limited to SETUP, SNAP, and GRAPH. These can be converted to standard FORTRAN with little trouble, though exacting a price in clarity.

3. Common Alterations

3.1. Changing Defaults

The default parameters are set in subroutines START and SPECIES. In START, the defaults are all set in the DATA statement at the start of the routine. In SPECIES, the defaults are set directly in the first few executable statements. Any of these can be changed without ill effect except VMAX in SPECIES. VMAX should always be initialized to zero as a flag that it has not been read in as input. To change the default value of VMAX, the constants in two statements must be changed: the statement

```
IF(VMAX.EQ.0)VMAX = AMIN1(4*VT,VVMAX)
```

And the statement

```
IF(VMAX.LE.4*VT)GOTO 12.
```

Many of the input parameters in START are computed if they are not input, so changing the defaults may introduce unexpected side effects.

3.2. New Diagnostics

Introduction of new diagnostics requires knowing the normalizations of the internal variables, and some straight forward changes in SNAP. The following table is a complete list of normalizations which should make the figuring of any new diagnostics relatively easy.

Variable	Description	Normalization
X	Length	Δx
V	Velocity	Δv
U	Velocity	$\frac{\Delta x}{\Delta t}$
F	Distribution Function	$\frac{\epsilon_0}{\Delta v \Delta t^2}$
RHO	Charge Density	$\frac{\epsilon_0}{\Delta t^2}$
E	Electric Field	$\frac{\Delta x}{\Delta t^2}$
PHI	Potential	$\frac{\Delta x^2}{\Delta t^2}$
SXJ	Species Current Density	$\frac{\Delta v}{\Delta t^2}\epsilon_0$
XJ	Summed Current Density	$\frac{\epsilon_0}{\Delta t^2}$
SKE	Species Kinetic Energy Density	$\frac{\Delta x \Delta v}{2\Delta t^3}\epsilon_0$
XKE	Summed Kinetic Energy Density	$\frac{\Delta x}{2\Delta t^3}\epsilon_0$
TSKE	Total Species Kinetic Energy	$\frac{\Delta x^2 \Delta v}{2\Delta t^3}\epsilon_0$
TKE	Total Kinetic Energy	$\frac{\Delta x^2}{2\Delta t^3}\epsilon_0$
ESE	Electrostatic Energy Density	$\frac{\Delta x^2}{2\epsilon_0 \Delta t^4}$
TESE	Total Electrostatic Energy	$\frac{\Delta x^3}{2\epsilon_0 \Delta t^4}$

The most efficient algorithm for computing a new diagnostic must be decided by the programmer, although it may be possible to simply emulate an existing diagnostic. The following must be changed, however: (1) the new diagnostic must be dimensioned in MAIN, SNAP, and where ever it is computed; (2) a plot frequency variable must be added to the common block EXTERN and a corresponding tally variable added to common block HIST (unless the new diagnostic is a history variable, in which case it must be added to the common block HIST); and, (3) a plotting section must be added to SNAP, along with a logic flag, unless the new diagnostic is a history variable. As a matter of style and clarity, the logical flow of the program should be altered as little as possible.

3.3. New Routines

Many interesting problems will require wholly new routines. In particular, new methods of perturbing the problem, and non-Maxwellian input distributions are likely possibilities. The former can be achieved by modifying PERTURB, and the latter can be achieved by altering the section of START which loads the boundary conditions. In general, though, the programmer will be forced to learn a good deal about the detailed operation of the code. The code is broken up into fairly logical and independent subroutines to try to minimize the effort required for such modifications.

4. Program Details

4.1. Common Blocks and Dynamical Variables

There are three important common blocks: INTERN, EXTERN, and HIST. INTERN contains the parameters which are of only internal importance. Virtually all subroutines use these variables, and once the program has been initialized, they do not change. The variables which are not input variables are:

DU velocity increment for units in which DV = DX*DU/DT

AQM adjusted q/m ratio to simplify the integrators

BC the incoming boundary conditions before any perturbations

BACK background charge density to simulate stationary ions if desired

BACKJ background current density (affects only the diagnostics)

EXTERN contains variables needed only in dealing with the outside world. Thus EXTERN appears only in the input and output routines, and its variables do not change after being set in the initializing routines. The variables are (again omitting the input variables):

DX x-grid spacing

DV v-grid spacing (in external units)

HIST is the only common block to contain dynamical variables. These variables are used only in START and SNAP. They consist of tally variables for counting the intervals between the different diagnostic plots, and arrays for saving history variables. The tally variables' names correspond to the input control variables with an M replacing the I at the beginning of the variable names. The history variables are (in order) Total Species Kinetic Energy, Total Kinetic Energy (summed over species), Total ElectroStatic Energy, Total Energy, and Ballistic current at the non-emitting wall.

The important dynamical variables are:

F Vlasov distribution function

PHI electrostatic potential

E electric field

RHO charge density (summed over species)

XJ current density (summed over species)

SRHO single species charge density

SXJ single species current density

SKE single species kinetic energy density

X0 test particle x position

V0 test particle v position

NP number of test particles

PBC perturbed boundary conditions

4.2. Subroutines

SETUP

This routine is the most installation dependent. First, it gets the name of the input file, either from the execute line, or by prompting for it. Then it reads in the first line of the input file, which contains the output header information. It also sets up the I/O linkages with the computer terminal and the plotting files.

START and SPECIES

START first reads in the input parameters which are species independent. Then it calls SPECIES to read in the species dependent parameters, and computes all internal parameters. Then the unperturbed boundary conditions are computed, and the initial distribution is loaded. The initial distribution is Maxwellian, and assumes a uniform electric field; this is a good approximation for an initially empty box, but not so good for an initially full box.

SUMQ and SUMV

In these routines the distribution function is summed over the velocities greater than the velocities of the test particles (the trapezoidal rule is used). SUMQ starts at the left hand edge, and follows the string of test particles. When a new particle is on the other side of a grid point from the previous particle, SUMV is called to sum the charge at that position. The charge is added if the new particle is to the right of the old one, and subtracted if it is to the left.

SUMV sums the charge at a single x-grid point. It also computes the ballistic current and kinetic energy, which slows down the code somewhat. (SUMV could be rewritten to calculate the current and energy only when they are needed, but some care must be taken to be sure that the vectorizability of the code is not lost.)

POISSON

POISSON solves Poisson's equation using a fast Fourier transform. A sparse matrix inversion method would perhaps be slightly faster, and marginally more understandable. The Fourier transform technique is a holdover from earlier periodic codes. It allowed an easy way to filter out high wave numbers, and gave a spatial mode analysis as a by-product; however, in a non-periodic code, such filtering and modal analysis is *not valid*, so there is really nothing to be gained in using a Fourier transform.

SNAP and GRAPH

These are the diagnostic routines. GRAPH exists merely to isolate some of the clutter involved in making a plot, and to make SNAP a little less installation dependent. SNAP also adjusts the ballistic current and kinetic energy for the time offset between position and velocity whenever the current or kinetic energy are to be plotted.

PERTURB

This routine perturbs the incoming boundary conditions when necessary, as per the description in the input parameters.

MOVEF

MOVEF is the integrator for the distribution function. It is long, and fairly repetitive. Structured statements (if .. then .. else, and such) were deliberately avoided to make the code easier to transport to a FORTRAN which lacks such statements. This exacts a price in clarity, but not a high one (the routine would probably be confusing even with structured statements). The best way to understand the routine is probably just to study it; the algorithm is straight forward, but lengthy. First the edges at $x = 0$ and $x = L$ are advanced half a timestep in velocity, and the boundary conditions are applied. Then the edges are advanced the last half timestep while the rest of the distribution is advanced a full timestep. Then, finally, the entire distribution is advanced a full timestep in position.

MOVED

This is the integrator for the test particles. Not only does it advance the particles using the leap-frog method, it also adds new particles when necessary, and repacks the arrays of particle positions and velocities. A significant saving in time might result from carefully choosing the frequency with which the arrays are added to and repacked and the spacing between

particles. Here, the simplest alternative was chosen: the arrays are added to and repacked each timestep, and new particles are introduced whenever two particles become separated by more than a grid cell spacing.

FINISH

This routine exists only for providing timing information. It must be either removed or totally rewritten on another installation.