ES2 USER'S MANUAL – VERSION 1

by

Kim Theilhaber

Memorandum No. UCB/ERL M87/23

11 May 1987

COVER PAGE

ES2 USER'S MANUAL – VERSION 1

by

Kim Theilhaber

Memorandum No. UCB/ERL M87/23

11 May 1987

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

ES2 USER'S MANUAL – VERSION 1

by

Kim Theilhaber

Memorandum No. UCB/ERL M87/23

11 May 1987

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Abstract

The two-dimensional, electrostatic plasma simulation code "ES2" is described. This includes a synopsis of the physical basis of the model and a summary of the numerical schemes used to implement it. This is followed by two examples of applications, one showing the formation of unmagnetized double layers, the other illustrating the formation of vortices in a configuration perpendicular to the magnetic field. Finally, the main section of the manual is presented: a copy of the "on-line" manual, in which a detailed description of the various options and parameters entering into ES2 is given.

Contents

ES2 User's Manual - version 1.
Kim Theilhaber, May 1st 1987.

1 Introduction.

The particle simulation code ES2 has been designed as a two-dimensional generalization of the one-dimensional, bounded electrostatic code PDW1[1], and is inspired by many features of this one-dimensional code. It consists of a bounded simulation region (see Fig.(1)), containing the plasma particles (electrons and ions), with periodic boundary conditions on the particles and fields in one direction (y), and finite bounding surfaces in the other, perpendicular direction (x). At the bounding surfaces, which can represent walls or plates in a physical device, or merely open boundaries in the simulation of an infinite plasma, the code allows for the emission and absorption of particles. It also allows for the volume creation of electron-ion pairs, thereby simulating a distributed ionization mechanism. The code models the outside world by an external circuit, so as to permit the flow of charge from one boundary to the other, subject to the desired circuit characteristics. The plasma simulation, while incorporating only electrostatic self-forces, also allows for an externally imposed magnetic field ($\mathbf{B}_0$ in the figure), which can be set at an arbitrary angle in the xy plane, or made parallel to the z-axis.

ES2 has been specifically designed for extending the areas of research undertaken with PDW1, in particular the study of the physics and dynamics of the sheathes and non-neutral structures which arise in plasma-wall interactions. In the MFE environment, it can run on both Cray-1's (C and D Machines), on the Cray X-MP (E Machine), and on the Cray-2 (B Machine). In what follows, we shall first give a synopsis of the normalizations introduced in the physical model. We shall then summarize the numerical schemes used in the code, and the various options available. Following this, we give two examples which illustrate two-dimensional plasma dynamics, first in an unmagnetized plasma, then in a configuration perpendicular to the magnetic field. In an Appendix, we present what is really the main section of this Memorandum: the "on line" version of the user's manual for ES2, with detailed specifications of all the options available.

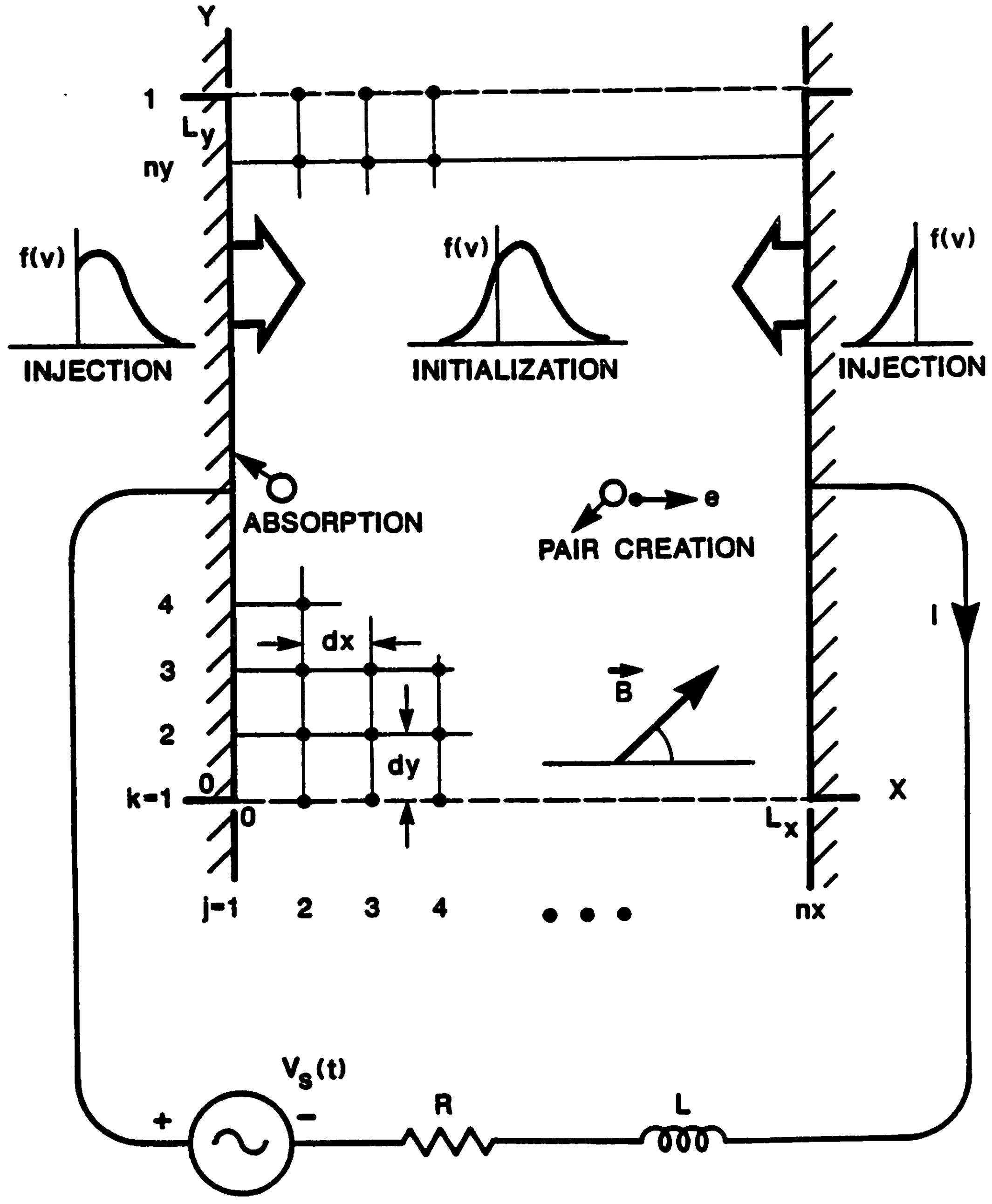

Figure 1: Geometry of the simulation region for the ES2 code.

2 Physical Normalizations.

In the electrostatic approximation assumed by ES2, the plasma is entirely described by the coupled Lorentz and Poisson equations:

$$\frac{d}{dt}\mathbf{v}_s = \frac{q_s}{m_s}(\mathbf{E} + \mathbf{v}_s \times \mathbf{B}), \tag{1}$$

$$\nabla^2 \phi = -\frac{e}{\epsilon_0}(n_i - n_e), \tag{2}$$

$$\mathbf{E} = -\nabla \phi, \tag{3}$$

where s is the species index, $s = e, i$, and where n_e and n_i denote the particle densities (number per unit volume). We assume singly-charged ions, so that $q_i = -q_e = e$.

In our simulation model, we choose $e/m_e = 1$, $\epsilon_0 = 1$ and we define n_i and n_e as the number of particles per unit area. Thus, the equations of motion become:

$$\frac{d}{dt}\mathbf{v}_e = -(\mathbf{E} + \mathbf{v}_e \times \mathbf{B}), \tag{4}$$

$$\frac{d}{dt}\mathbf{v}_i = \frac{m_e}{m_i}(\mathbf{E} + \mathbf{v}_i \times \mathbf{B}), \tag{5}$$

$$\nabla^2 \phi = -e(n_i - n_e), \tag{6}$$

$$\mathbf{E} = -\nabla\phi, \tag{7}$$

For instance, let us consider a homogeneous simulation plasma where electrons and ions have an equilibrium density of $\overline{n}$ (per unit area), and where the electrons are initialized with thermal velocity v_{te}, the ions with thermal velocity v_{ti}, with $v_{ti} \approx (m_e/m_i)^{1/2} v_{te}$. Then the plasma and cyclotron frequencies are simply:

$$\begin{aligned}
\omega_{pe} &= (e\overline{n})^{1/2}, & (8)\\
\omega_{pi} &= \left(\frac{m_e}{m_i}\right)^{1/2} \omega_{pe}, & (9)\\
\omega_{ce} &= B, & (10)\\
\omega_{ci} &= \frac{m_e}{m_i} B, & (11)
\end{aligned}$$

If we then choose $v_{te} = 1$ and assume $\overline{n}$ is given, then by setting $e = 1/\overline{n}$ we insure that $\omega_{pe} = 1$ and hence that $\lambda_{de} = 1$ as well. In other words, $\overline{n}$ automatically becomes the number of particles per Debye square.

3 Numerical Schemes.

We shall briefly describe the numerical schemes use in ES2.

The field solver is of a mixed type, similar to the one described by Birdsall and Langdon in [2]: Poisson's equation is solved by using a Fourier representation of the equation in the y direction, that is by transforming from y to k_y using an FFT (Fast Fourier Transform), and by solving a finite-difference scheme in the x direction for each of the Fourier components. A final inversion, from k_y to y space using the FFT, completes the solution process. The finite-difference equation for the $k_y = 0$ Fourier component requires a special treatment, as it must account for the surface charge that accumulates on the boundary surfaces. This treatment is in fact nearly identical to the formulation of Poisson's equation in PDW1[1]. For the higher Fourier components, the boundary conditions are zero, and the solution of the finite difference equations is slightly simpler[2].

The numerical particles are "square" in shape, and use a straightforward area-weighting for the gather/scatter operations, as described in [3]. When the particles hit a boundary they are absorbed, and their charge is instantaneously dispersed as a uniform surface charge on the conductor's surface. Similarly, when a particle is injected from a surface, it leaves a distributed hole of opposite charge behind, and the hole's charge contribution is equally accounted for. A repacking algorithm periodically discards the coordinates and velocities of particles which have been absorbed by a surface, and thus keeps the total number of particles in the system at any one time within reasonable bounds of size.

The external circuit uses an algorithm similar to the one described in [1]. It can model a voltage source, a resistor and an inductor in series, or more simply a short or an open circuit across the plasma region.

4 Options.

Various options exist for setting up the physical conditions of a simulation. The magnetic field can be chosen at an arbitrary angle to the xy axes, or be made parallel to the z axis. While in the present version of ES2 the magnetic field is constant, the algorithm which moves the particles can be modified for a non-uniform magnetic field, while conserving vectorizeability. The program can be initialized with the simulation region full or empty of particles, with prescribed stationary or drifting Maxwellians. One can separately specify the injection rates of each species at each bounding surface, and the rate of volume creation of electron-ion pairs within the simulation region. The distribution functions of the emitted particles are Maxwellians, with specified temperatures and drifts. One available option assumes that the simulation region bathes in a "plasma reservoir", of given density, temperature and drift velocities; the injection rate is then automatically adjusted in accordance with the reservoir parameters.

At the present time, ES2 incorporates diagnostics in the form of scatter plots, graphs of the distribution functions, field plots of the potential and electric fields (cross-sections and contours), plots of Fourier components of the fields, and time histories of energies, circuit quantities, etc. A more specialized diagnostic provides time-averaged snapshots of grid quantities.

5 Example 1: Double Layer Formation.

As a first example of a run of the ES2 code, we consider the two dimensional simulation of instability induced by a drifting population of Maxwellian electrons, in the presence of a cold ($T_i \ll T_e$) population of ions. This is the regime for the so-called ion acoustic instability, predicted by linear plasma theory. Through particle simulations, we can hope to gain insight in the nonlinear dynamics of the process[4,5].

The data file for a run which illustrated the formation of a double layer is shown in Table 1. The system dimensions, for the configuration illustrated in Fig.(1), are $L_x = 32$, $L_y = 32$, with a grid spacing $\Delta x = 1.0$ and $\Delta y = 1.0$. Thus, the numerical mesh has $N_x \times N_y = 33 \times 32 = 1056$ points. The system is initially filled with equal populations of electrons and ions (about 5000 of each), with densities $n_e = n_i = 5$ per unit area, electron charge $e = 0.2$ and mass ratio $m_i/m_e = 100$. The thermal velocities are $v_{te} = 1.0$ and $v_{ti} = 0.01$, corresponding to "temperatures" $T_e = v_{te}^2 m_e = 0.2$ and $T_i = 0.02$, so that $T_i \ll T_e$. Furthermore, the electrons are initially set drifting with a velocity $v_{de} = 1.0$. With these normalizations, we find that $\omega_{pe} = 1$, $\omega_{pi} = 0.1$, $c_s = (T_e/m_i)^{1/2} = 0.1$, $\lambda_{de} = 1$. In particular, we have $L_x = L_y = 32\lambda_{de}$. Finally, in this run we chose the plasma to be unmagnetized, with $B = 0$.

The injection rates are set to those that would obtain if the simulation region bathed in a reservoir of plasma, with exactly the same parameters as those outlined above. Thus, in the absence of collisional and collective effects, the system should be in a steady state, with injection at the boundaries exactly replenishing the particles lost by absorption at these same boundaries. In fact, this is not the case: first, because even when $v_{de} = 0$, there is a slow but measurable heating of the ions due to numerical collisonality, and second, when $v_{de} \neq 0$, turbulence develops in the plasma.

Finally, the external circuit is set to a short-circuit condition: we require that $\phi(0, y) = \phi(L_x, y) = 0$ at all times.

In Fig.(2) we display the ion phase space ($v_x$vs x) at $t = 140$ ($\omega_{pi}t \approx 14$). This scatter plot shows *all* ions in the system. A large ion vortex has formed about $x = 6$, with a velocity trapping width of order $\Delta v_{tr} \approx 0.08 \gg v_{ti} = 0.01$. The ions in the vortex are trapped in a large-amplitude wave, which propagates in the direction of electron drift with a velocity comparable to the ion acoustic velocity $c_s = 0.1$. The two dimensional potential structure of this wave is shown in Fig.(3): this is a perspective view of the potential $\phi(x, y)$, averaged over a time interval $T = 1.5\omega_{pi}^{-1}$ so as to suppress the high-frequency ($\omega \approx \omega_{pe} = 1$) fluctuations of the fields. Fig.(3) displays three features which are expected in this situation[4,5]. The first is a broad potential dip, which traps the ions observed Fig.(2). The second is a positive potential step, downstream from the ion vortex, which is formed by the asymmetric reflection of electrons by the potential dip. The third feature is a low-frequency ($\omega \leq \omega_{pi}$) turbulence, which eventually dominates the "coherent" features of the potential profile that we have just just discussed.

TABLE 1: Data file for the double-layer run.

```
*/ =============================================
*/ ==       data180  for  gdd180             ==
*/ ==          apr 29 1987                   ==
*/ ==   double-layer problem                 ==
*/ =============================================
*select box=b14,account=818sbl,printlog=none
*file name=datav

 irun= 13
 n1d= 0  nsym= 0
 dx= 1.0  dy= 1.0
 dt= 0.2
 ntau= 600  kpack= 20

$end ============ external circuit ============
 nshort= 1
 ncirc= 1
 rext= 0.0  blext= 100.0
 vssmax= 0.0

$end ============ particles and fields ========
 fmime= 100.0  ebar= 0.2  rwe= 1.0
 theta= 0.0
 nperp= 0
 vmax= 5.0  vthe= 1.0  vthi= 0.01
 vdein= 1.0  vdell= 1.0  vderr= 1.0
 vdiin= 0.0  vdill= 0.0  vdirr= 0.0
$end
 npe= 10240  npi= 10240
 nee= 1  nii= 1  nback= 0
$end ........... injection parameters .................
 ninj= 1
 snejl= 0.0  snejr= 0.0  snijl= 0.0  snijr= 0.0
 nres= 1
 fnrel= 5.0  fnrer= 5.0  fnril= 5.0  fnrir= 5.0
 snei= 0.0  nbit= 0  ngau= 1
$end ============== data for all plotting =====================
 numl= 10
 numx= 1  numy= 1
 vme= 5.0  vmi= 0.1  xwind= 4.0
 kjump= 450
```

```
$end ........ jsnap(1)= 0 sets skips of "kjump" for snapshots ...
 jsnap(1)= 500 750 1000 1250 1500 1750
$end ............. data for time-average snapshots ..............
 naver= 1  nphiav= 1  kskip= 100  jkav= 100
$end ........ javhi(1)= 0 sets auto. skips of kskip .............
 javhi(1)= 550 800 1050 1300 1550 1800 1995
$end ......... nophi=1 for output to binary file ...
 nophi= 1  noden= 0  nopart= 0
 npout= 1575  ihop= 13

$end ============ switches for all outputs ========
 ndens= 1  nfilt= 3
$end ............ scatter plots ...................
 nxye= 1  nxvxe= 1  nxvye= 0  nxvze= 0  nvxvye= 0
 nxyi= 1  nxvxi= 1  nxvyi= 0  nxvzi= 0  nvxvyi= 0
$end ............ distribution functions ..........
 nfex= 1  nfey= 0  nfez= 0
 nfix= 1  nfiy= 0  nfiz= 0
$end ............ fields ..........................
 nex= 1  ney= 0  nexf= 0  neyf= 0  neplot= 0
$end ............ test particle ...................
 ntest= 0
$end ===================================================


*/ ===============================================
*/ ==     rec will contain the execution record   ==
*/ ===============================================
*file name=rec

*
```

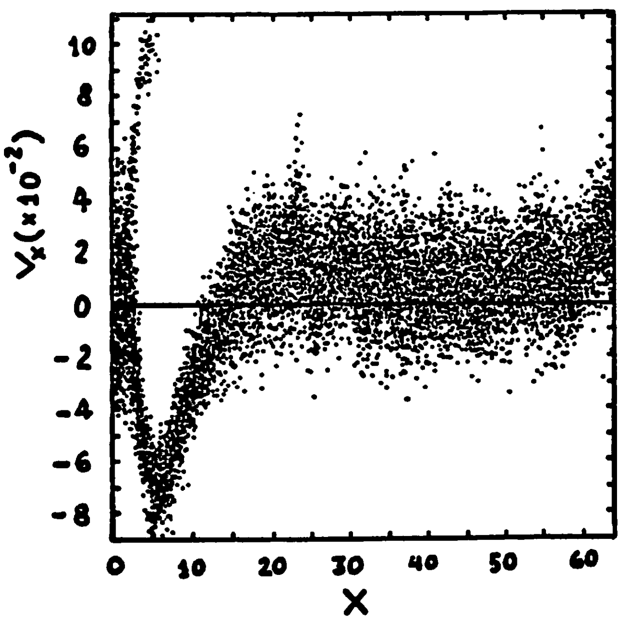

Figure 2: Scatter-plot of ion phase space x-v_x, taken at $\omega_{pe}t = 100$ in the double-layer run.

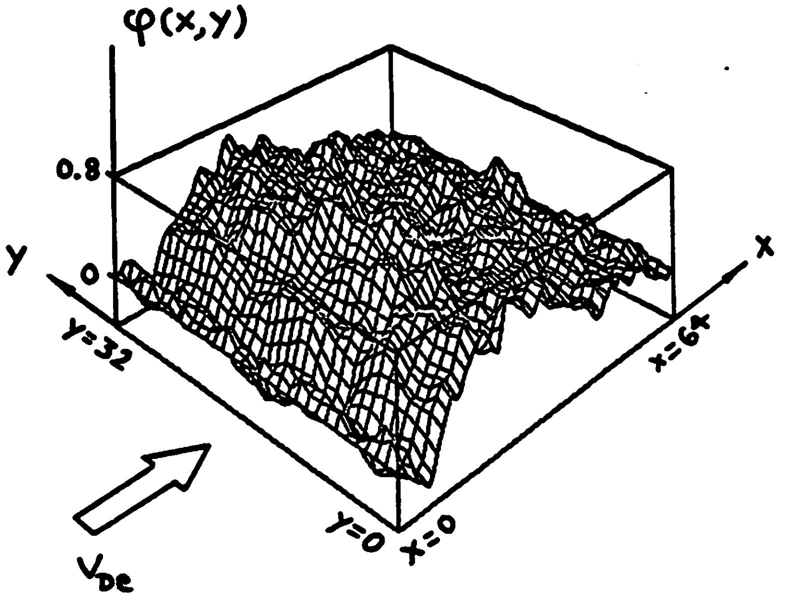

Figure 3: Electrostatic potential at $\omega_{pe}t = 100$ in the double-layer run, time-averaged over 1.5 ω_{pi}^{-1}.

6 Example 2: Vortex Formation in a Crossed-Field Sheath.

In this example, we consider a system bounded by two conducting and absorbing walls which is initially filled with equal numbers of electrons and ions. As we shall see, the presence of the boundaries induces edge instabilities which saturate into long-lived vortices. The parameters for this example are given in Table 2. In the normalized units of ES2, the mass ratio is $m_i/m_e = 40$ and the magnetic field is $B_0 = 1$, with an initial particle density of $\overline{n} = 5$ particles per unit area. Because ES2 assumes $m_e = e$, and $\epsilon_0 = 1$, this implies that $\omega_{ce} = 1$, $\omega_{pe} = 0.3$, $\omega_{pi} = 0.047$, and $\omega_{pi} = 0.025$. We have initially Maxwellian electrons and ions, with $T_e = T_i$, and the thermal velocities are $v_{te} = 1$, $v_{ti} = 0.158$. From these we obtain the length scales $\rho_e = 1$, $\lambda_{de} = \lambda_{di} = 3.3$, and $\rho_i = 6.32$, so that we have the ordering $\rho_e \ll \lambda_{de} = \lambda di \ll \rho_i$. The frequency ratios are then: $\omega_{pe}/\omega_{ce} = 0.3$, $\omega_{pi}/\omega_{ci} = 1.96$. The initial number of ions and electrons in the system is for each species $N = 40960$, and the system size is $L_x = 64 \sim 10\rho_i$, and $L_y = 128 \sim 20\rho_i$.

In Figs.(4a,b,c), we show an overview of the time-evolution of the electrostatic potentials in the system, by displaying three contour plots of $\phi(x, y)$, at times $\omega_{ci}t = 5$, 55 and 305. Fig.(4a) shows the essentially y-uniform sheath which forms at the beginning of the evolution of the system, after one rotation of the ions, $\omega_{ci}t \sim 2\pi$. This sheath is due to an initial loss of ions which have impacted into the walls. This results in a layer of depletion of positive charge over a depth in x of about $2\rho_i \sim 13$, and a corresponding potential drop from the wall into the plasma. This is in sharp contrast to the situation in the unmagnetized sheath, which is dominated by electron flow to the walls, and in which there is a potential rise from the wall into the plasma. In Fig.(4a), the total potential drop from the wall to the "valley" floor is $e\Delta\phi/T_i = 1.6$.

The sheathes which are shown in Fig.(4a) contain strongly nonuniform electric fields $E_x(x)$ pointing inward from the walls. These fields in turn give rise to highly sheared $\mathbf{E} \times \mathbf{B}$ drifts of electrons and ions in the y direction. As sheared fluid flow is in general unstable, it might be expected that the initial structure shown in Fig.(4a) will not persist, with the flow vulnerable to the Kelvin-Helmholtz instability. This is precisely what is observed in the subsequent evolution of the potential: y-dependent ripples develop in the potential contours, and are both amplified and convected by the $\mathbf{E}\times\mathbf{B}$ flow, over a time-scale of order $\omega_{ci}t \sim 20$. A first saturated state of the instability is shown in Fig.(4b). This state consists of a set of well-defined circular potential contours which drift parallel to the walls, in accordance with the local $\mathbf{E} \times \mathbf{B}$ drift (downwards on the left, upward on the right). These contours correspond to sizeable perturbations of the potential, with $\delta\phi/T_i \sim 0.5$. To the extent that the particle motion is determined by the $\mathbf{E}\times\mathbf{B}$ drift, the electrostatic potential is in fact a stream function for the particles, and the circular structures are quite simply large-amplitude vortices, in close analogy to the evolution of the Kelvin-Helmholtz instability in fluid flow.

The state shown in Fig.(4b) is only an approximate steady-state. Over a longer time scale, of the four vortices shown in Fig.(4b) only two larger vortices with $e\delta\phi/T_i \sim 1$ survive by $\omega_{ci}t = 305$, as shown in Fig.(4c). The process which leads in time from Fig.(4b) to Fig.(4c) occurs as follows. The vortices induce a steady ambipolar transport of particles from the inner plasma to the walls which gradually depletes the density profiles. Simultaneously the density profiles change by broadening and by taking on a triangular shape, and as the profiles broaden in x, the vortices grow in extent, in both x and y. This leads to a competition between the vortices filing parallel to each wall, the outcome being that in each line of vortices, the largest vortex, which is also the fastest, overtakes and absorbs the others in what might be considered completely inelastic collisions. On the other hand, the counterstreaming vortices have only highly "elastic" encounters, and periodically overlap with no subsequent effect. The final state shown in Fig.(4c) thus consists of two large and apparently stable counterstreaming vortices, which continue to transport particles outward. The density profile in this configuration is approximately triangular and quasineutral ($n_e(x) \approx n_i(x)$), with the exception of small trapped-electron populations in the center of each vortex.

TABLE 2: Data file for the crossed-field sheath run.

```
*/ ===============================================
*/ ==       data180  for  gdd180              ==
*/ ==          apr 29 1987                    ==
*/ ==     crossed-field problem.              ==
*/ ===============================================
*select box=b14,account=818sbl,printlog=none
*file name=datav

 irun= 27
 n1d= 0  nsym= 0
 dx= 1.0  dy= 1.0
 dt= 1.0
 ntau= 4995  kpack= 20

$end ============ external circuit ============
 nshort= 0
 ncirc= 0
 rext= 0.0  blext= 100.0
 vssmax= 0.0

$end ============ particles and fields ========
 fmime= 40.0  ebar= 0.018  rwe= 1.0
 theta= 0.0
 nperp= 1
 vmax= 5.0  vthe= 1.0  vthi= 0.1581139
 vdein= 0.0  vdell= 0.0  vderr= 0.0
 vdiin= 0.0  vdill= 0.0  vdirr= 0.0
$end
 npe= 100  npi= 100
 nee= 1  nii= 1  nback= 0
$end ........... injection parameters ..................
 ninj= 0
 snejl= 0.0  snejr= 0.0  snijl= 0.0  snijr= 0.0
 nres= 0
 fnrel= 0.0  fnrer= 0.0  fnril= 0.0  fnrir= 0.0
 snei= 2.51  nbit= 0  ngau= 1
$end ============== data for all plotting =====================
 numl= 20
 numx= 1  numy= 1
 vme= 5.0  vmi= 1.0  xwind= 4.0
 kjump= 450
```

```
$end ........ jsnap(1)= 0 sets skips of "kjump" for snapshots ...
 jsnap(1)= 5000
$end ............. data for time-average snapshots .............
 naver= 0  nphiav= 1  kskip= 16  jkav= 10
$end ........ javhi(1)= 0 sets auto. skips of kskip ............
 javhi(1)= 100000
$end ......... nophi=1 for output to binary file ...
 nophi= 0  noden= 0  nopart= 0
 npout= 1575  ihop= 13

$end ============ switches for all outputs ========
 ndens= 1  nfilt= 3
$end ............ scatter plots ..................
 nxye= 1  nxvxe= 0  nxvye= 0  nxvze= 0  nvxvye= 0
 nxyi= 1  nxvxi= 0  nxvyi= 0  nxvzi= 0  nvxvyi= 0
$end ............ distribution functions ..........
 nfex= 1  nfey= 0  nfez= 0
 nfix= 1  nfiy= 0  nfiz= 0
$end ............ fields .........................
 nex= 1  ney= 0  nexf= 0  neyf= 0  neplot= 0
$end ............ test particle ..................
 ntest= 0
$end ================================================

*/ ==============================================
*/ ==    rec will contain the execution record  ==
*/ ==============================================
*file name=rec

*
```

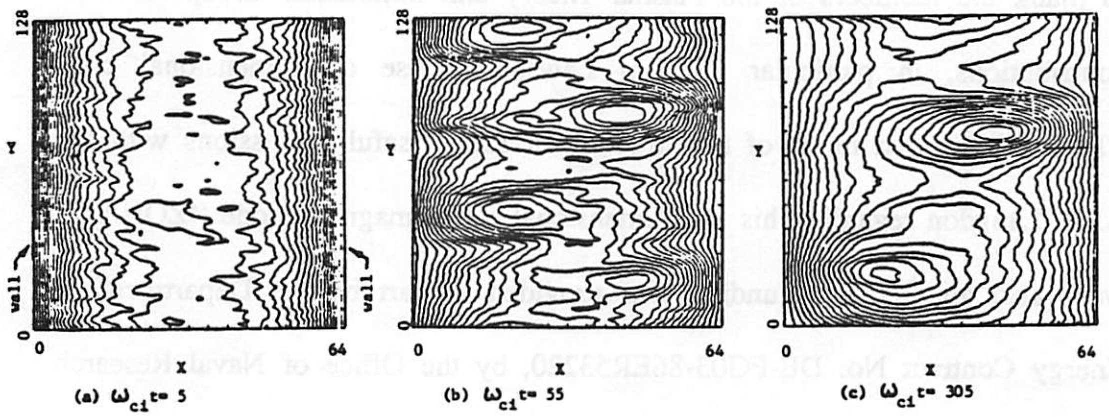

Figure 4: Equipotential contours at three different times in the crossed-field configuration; (a) $\omega_{ci}t = 5$, (b) $\omega_{ci}t = 55$, (c) $\omega_{ci}t = 305$.

Acknowledgments

The author wishes to thank Professor C. K. Birdsall for his essential role in initiating work on bounded plasma simulations, and for his participation in the creation of the two-dimensional bounded code "ES2". The author also wishes to thank the members of the Plasma Theory and Simulation Group for their contributions, in particular William Lawson, whose one-dimensional code "PDW1" inspired much of the structure of ES2. Useful discussions with Dr. A. B. Langdon regarding his two-dimensional electromagnetic code "ZOHAR" were also appreciated. Funding was provided in part by the Department of Energy Contract No. DE-FG03-86ER53220, by the Office of Naval Research Contract No. N00014-85-K-0809, and by a gift of the Varian Corporation.

References

[1] *PDW1 User's Manual*, Wm. Lawson, Electronics Research Laboratory Memorandum No. UCB/ERL M84/37, 27 April 1984.

[2] *Plasma Physics via Computer Simulation*, C.K. Birdsall and A.B. Langdon, McGraw-Hill, 1985, p. 318.

[3] *Idem*, p. 312.

[4] J.S. DeGroot et al., Phys. Rev. Letters, 38, 22 (1977) 1283.

[5] C. Barnes, M.K. Hudson, W. Lotko, Phys. Fluids, 20, 4 (1985) 1055.

7 Appendix: Detailed "On-Line" Manual.

```
% ============================================================
% =                 MANUAL FOR ES2 USAGE.                    =
% =                                                          =
% =                   by K. Theilhaber.                      =
% =                     UC Berkeley.                         =
% =                                                          =
% =               Version 12, may 05 1987.                   =
% ============================================================
% =                                                          =
% =       This manual can be obtained with:                  =
% =                                                          =
% =           filem read 1214 .twod6 manua.12                =
% =                                                          =
% =       It describes how to use the current version        =
% =       of ES2, as contained in the files:                 =
% =                                                          =
% =               gdd180       data180                       =
% =                                                          =
% =       These are also in the directory  .twod6.           =
% =                                                          =
% ============================================================
```

```
1. Introduction:
   ------------

        The code "ES2" is a two-dimensional, electrostatic, bounded
plasma simulation program, patterned on the one-dimensional
bounded code "PDW1".  ES2 evolves an ensemble of
electrons and ions, in a region bounded in one
direction (x) and periodic in the other (y),
with two spatial coordinates and three velocity components for
each of the particles. The particles are moved according to
their self-consistent electrostatic fields. Provision is also made
for an externally imposed electric field, and for an external
magnetic field. The latter can point in an arbitrary direction
in the x-y plane, or parallel to the z-direction.

        In what follows, I shall describe how to
use ES2 in the NMFE environment, that is on
the CRAY-1's "C", "D", on the CRAY-XMP
"E" machine, and on the CRAY-2 "B" machine . I will try to account
```

for variations in different implementations, by noting them both in the body of the following text, and in updates of the present manual. For instance, a recent implementation of ES2 has a spatially varying magnetic field as well. This has not yet been adapted to the version described here.

An essential feature of ES2 resides in the implementation of the boundary conditions. These allow for absorption and injection of particles from the bounding surfaces. Coupled to the plasma simulation is an external circuit, which permits the flow of charges from one boundary to the other. For a fuller description of the numerical methods used in ES2, there is a write-up in the Plasma Theory and Simulation Group Quarterly Progress Report, 3rd and 4th Quarters 1985.

An additional feature of ES2 is that it allows for volume creation of electron-ion pairs. This provides another physical mechanism for replenishing the number of particles, which are otherwise lost to the plane boundaries, in many of the physical configurations that we have studied.

In what follows, I shall describe the input parameters to the code. These include: (i) the physical parameters (thermal velocities, densities, etc),
(ii) the numerical parameters (mesh size, time step, etc),
(iii) the parameters and switches for the diagnostics (frequency of snapshots, which kinds desired, etc).

2. Running the Code:

In the following, we shall refer to the current version of ES2, version "180". The source files for ES2 consist of:

```
gdd180   --------->  source file.

data180  --------->  data file.
```

and I should also mention the source file for a graphics post-processing program:

```
                phiin.25  --------->  post-processor.
```

This post-processor produces 3d and contour plots of the potential and of other two-dimensional variables.

```
..............................................................
.     These files, in their current versions, are available from
.
.     filem; do:                                               .
.                                                              .
.          filem read .twod5  gdd180  data180  phiin.25        .
.                                                              .
..............................................................
```

I shall describe in the next section how to adjust parameters in both "gdd180" and "data180". Assuming this has already been done, the operating procedure is as follows: First compile "gdd180"; this produces an executable file "xgdd":

```
          cosmos gdd180  --------->   xgdd
```

On the C and D machine I have been using the CIVIC and CFT compilers, while on the B machine only the CIVIC compiler is available. Switching between the two compilers involves simply choosing one of the following lines of job control language:

```
            *cft i=evob,b=bevob

            *civic i=evob,b=bevob
```

these lines occuring at the very end of the source code "gdd180".

The next step is to submit "data180" to cosmos; this produces two files, "datav", the file accessed by "xgdd" as the data file, and "rec", a file which is a log of the execution of the program (many subroutines, as they are called, print something like "entry in ...." in "rec", error messages also appear here):

```
                                         datav:  data file.
      cosmos data180   --------->
                                           rec:  log file.
```

The next step is to submit "xgdd", by typing:

```
                xgdd / t v
```

After some initializations, the code will request the value of the
parameter "nhalt". There are two possibilities at this point:

(i) type simply the dollar sign $, followed by a carriage return; the program will then execute assuming nhalt=1, that is, after execution it will stop in such a manner as to allow a restart.

(ii) or type:

```
nhalt=0$
```

(followed by a carriage return). The program will then execute in such a way as to terminate permanently after execution.

Upon execution, xgdd produces several kinds of files:

```
                              +xgdda :    dropfile.

      xgdd  ------------>     f3.... :    graphics files.

                              ophi.. :    binary output file,
                                          contains potential
                                          snapshots (its creation
                                          depends on the switch
```

"nophi" -- see below).

opar.. :	binary output file, contains snapshot of a subset of particle positions (its creation depends on the switch "nopar" -- see below).
oden.. :	binary output file, contains snapshot of y-averaged particle densities (its creation depends on the switch "noden" -- see below).

The use of the output files "ophi", "opar" and "oden" will be discussed below. The graphics files are named "f3aaaa..", "f3bbbb..", etc, corresponding to the first submission, and subsequent restarts of the code. Similarly, the binary files are named:

```
          ophi01, ophi02, etc,  ....
```

Provision is made for up to 20 restarts, after which ES2 will cycle back to the letters "aaaa" for the graphics files, and "01" for the binary files.

An important feature of the code is its restart feature. If nhalt=1, the code ends on a so-called "HALTGO" statement, for which the drop file can be re-submitted. To restart a run, do:

```
            +xgdda / t v
```

This new submission will create new graphics files f3..... , but will assume that the previously defined auxilliary files "datav" and "rec" are present. For each restart, the graphics produced during the run have a header in the first frame which specifies the number of the restart ("part n"). This helps keeping track of all the restarts.

A good procedure to follow is to save the drop file after each successful run (under some temporary name), before resubmitting it for yet another execution. This procedure can provide a considerable amount of security, when one is doing a long computer run.

Upon re-submission, the user is first prompted for a new value of the number of time steps "ntau" (the default is just the previous value), and a new value of "nhalt" (0 or 1). In a second input line, the user is prompted for the value of "ng":

```
nhalt= 0,  not a restartable run
       1,  (default) a restartable run.

   ng= 0,  quit and exit
       1,  (default) resume
       2,  resume, but provide first graphics
           of the initial state.
```

In all cases, the syntax to enter variables (say xxx, yyy) is:

```
    xxx= 2.0 yyy= 1.5$    or:    xxx= 2.0$     or:     $
```

where the $ sign alone signifies the adoption of the default values.

Finally, we note that upon a restart, all counters for snapshots, etc, are reset to zero.

3. Parameters in the Source File:

Parameters in the source file "gdd180" are controlled by the parameter statements in the following CLICHE, which occurs near line 30 of the code.

```
   ...............................................
   .     cliche param                              .
   .      parameter(nx=33,ny=128)                  .
   .      parameter(nmax=10000)                    .
   .      parameter(npemax=21000,npimax=21000)     .
```

```
.      parameter(npsto=1800)                    .
.      parameter(numsec=15)                     .
.      parameter(nv1=100)                       .
.      parameter(jssmax=50)                     .
.      parameter(numsct=12000)                  .
.     endcliche                                 .
.................................................
```

These are defined as:

```
   nx=  Number of grid points in x.  The grid point j=1
        (x=0) corresponds to the left wall, the grid point
        j=nx (x=lx) to the right wall.  Thus the total system
        length is  lx= (nx - 1) * dx .

   ny=  Number of grid points in y.  The grid point
        k=1 is at y=0, and the full extent
        of the system in y is  ly= ny * dy
        (there is a "ghost" grid cell at k=ny+1).
        NOTA BENE:  "ny" must be a power of 2.
                    -------------------------

 nmax=  Maximum number of time steps that can be run in any
        single submission (an indefinite number of time steps
        can be run by resubmitting the program).

npemax= Maximum number of electrons that can be accomodated.

npimax= Maximum number of ions that can be accomodated.

 npsto= Maximum number of particle positions which can be
        buffered-out to the binary file "opar"; i.e:
        with nopar= 1 (see below), up to  npsto  pairs
        (xe(i),ye(i)) and  npsto  pairs (xi(i),yi(i)) can
        be outputted when desired.


numsec= Maximum number of cross-sections of the fields and
        distribution functions.

   nv1= Number of points used in the calculation of the
        distribution functions f(v).
   jssmax=  Maximum number of times for taking snapshots
            that can be specified in any one submission.

   numsct=  Maximum number of particles plotted in a scatter plot.
    ------------------------------------------------------------------
```

In the above, the parameters "numsec", "nv1", "jssmax", "numsct" are used only for the diagnostics, and the user will probably change them only on rare occasions.

4. Parameters in the Data File:

Below is a full listing of variable entries in the data file:

```
.........................................................
...         data180                                   ...
.........................................................

 irun= 403
 n1d= 0  nsym= 0
 dx= 1.0  dy= 1.0
 dt= 0.2
 ntau= 300  kpack= 20

$end ============ external circuit ============
 nshort= 0
 ncirc= 0
 rext= 0.0  blext= 100.0
 vssmax= 0.0

$end ============ particles and fields ========
 fmime= 40.0  ebar= 0.018  rwe= 1.0
 theta= 0.0
 nperp= 1
 vmax= 5.0  vthe= 1.0  vthi= 0.1581139
 vdein= 0.0  vdell= 0.0  vderr= 0.0
 vdiin= 0.0  vdill= 0.0  vdirr= 0.0
```

```
$end
 npe= 20480  npi= 20480
 nee= 1  nii= 1  nback= 0
$end ........... injection parameters .................
 ninj= 0
 snejl= 0.0  snejr= 0.0  snijl= 0.0  snijr= 0.0
 nres= 1
 fnrel= 5.0  fnrer= 5.0  fnril= 5.0  fnrir= 5.0
 snei= 2.51  nbit= 1  ngau= 0

$end ============== data for all plotting =====================
 numl= 20
 numx= 1   numy= 1
 vme= 5.0  vmi= 1.0  xwind= 4.0
 kjump= 450
$end ........ jsnap(1)= 0 sets skips of "kjump" for snapshots ...
 jsnap(1)= 2500  0
$end ............. data for time-average snapshots .............
 naver= 1  nphiav= 1  kskip= 450  jkav= 100
$end ........ javhi(1)= 0 sets auto. skips of kskip ............
 javhi(1)= 2500   4990
$end ......... nophi=1 for output to binary file ...
 nophi= 1  noden= 1  nopar= 1

$end ============ switches for all outputs ========
 ndens= 1  nfilt= 3
$end ............ scatter plots ...................
 nxye= 1  nxvxe= 1  nxvye= 1  nxvze= 1  nvxvye= 1
 nxyi= 1  nxvxi= 1  nxvyi= 1  nxvzi= 1  nvxvyi= 1
$end ............ distribution functions ..........
 nfex= 1  nfey= 1  nfez= 1
 nfix= 1  nfiy= 1  nfiz= 1
$end ............ fields .........................
 nex= 1  ney= 1  nexf= 1  neyf= 1  neplot= 1
$end ............ test particle ..................
 ntest= 0
$end ===============================================

.................................................................
```

The explanations of these parameters follow, grouped in data statements:

```
-----------------------------------------------
   BASIC MESH AND TIME EVOLUTION PARAMETERS
-----------------------------------------------
 irun= 4
 n1d=0  nsym= 0
 dx= 1.0  dy= 1.0
 dt= 0.2
 ntau= 5000  kpack= 20
...............................................
```

```
     irun=  Number of the run. This is just a label which appears
            on the header of graphics, etc. It help keep track
            of things.


      n1d= 0, Run the code as fully two-dimensional.

         = 1, Run the code as one-dimensional. The particles are
              "stretched" along y into sheets of charge. All
              2d quantities become independent of the y-index.
              For such a 1d run, I suggest ny=8 (ny=4,2 might
              create the usual indexing problems).  In this mode,
              ES2 will run about 4 times slower than PDW1.

     nsym= 0, Run the code treating each boundary as a physical,
              conducting and absorbing wall.

         = 1, Apply an inversion symmetry to the right-hand wall.
              The right-hand boundary is still an equipotential,
              but any particle which leaves through this boundary
              is intantaneously re-injected with position and
              velocity components reflected through the midpoint.
              Thus the new velocity is minus the old one.


       dx=  Mesh size in x.  The total system length in x
            is  (nx -1) * dx.

       dy=  Mesh size in y.  The total system length in y
            is  ny * dy.

       dt=  Time step increment.
```

```
    ntau= Number of time steps for the very first submission
          of the program.  If  ntau>nmax, the program aborts
          with an error message.

   kpack= The particle arrays are re-packed every  kpack  time-
          steps.  This feature is particularly important when
          particles are being injected into the system.  To avoid
          repacking, just set  kpack  to a very large value
          (i.e.:  kpack>nmax ).
```

```
--------------------------------------
       CIRCUIT PARAMETERS
--------------------------------------
 nshort= 0
 ncirc= 0
 rext= 0.0  blext= 100.0
 vssmax= 0.0
.....................................
```

These switches schematically implement the circuit:

```
               phi=0
                 .                   .
                 .                   .
                 .                   .
                 .   simulation      .
         .........     region        .........
         .       .                   .       .
         .       .                   .       .
         .       .                   .       .
         .                                   .
         .                                   .
         .                                   .
         .                                   .
         ........ R ..... L ...... V  ........
                  ext     ext    -  ss +
```

with the following options:

ncirc= 0, Overrides all other options: the circuit is open and the walls are "floating".

= 1, Put in a circuit, according to the specifications of the other parameters.

nshort= 0, The circuit behaves according to the diagram above.

= 1, The circuit is short-circuited, and ignores all other specifications.

rext= Value of the series resistance.

blext= Value of the series inductance.

vssmax= Value of the d.c. voltage source indicated above.

NOTA BENE: in the present version, the circuit subroutine is not fully operational for all combinations of the parameters rext, blext and vssmax. Ask me for details for a specific application.

```
-------------------------------------
  PARTICLES AND FIELD PARAMETERS
-------------------------------------
 fmime= 40.0  ebar= 0.018  rwe= 1.0
 theta= 0.0
 nperp= 1
 vmax= 5.0  vthe= 1.0  vthi= 0.1581139
 vdein= 0.0  vdell= 0.0  vderr= 0.0
 vdiin= 0.0  vdill= 0.0  vdirr= 0.0
.....................................
 npe= 20480  npi= 20480
 nee= 1  nback= 0  nii= 1
.....................................
```

fmime= Mass ratio, mi/me .

ebar= Electron charge; the electron mass is set equal to this, me= ebar .

rwe= Magnetic field strength (it is constant in this version).

theta= Angle (in radians) of the magnetic field to the x-axis.

nperp= 0, Magnetic field is in the x-y plane, with the angle specified by theta.

= 1, Magnetic field is parallel to z (theta is ignored).

vmax= Maximum value of (v/vthermal) that is used when loading the particles according to stationary or drifting Maxwellians; for large values of (vdrift/vthermal), vmax might have to be chosen large. For vdrift of the same order as vthermal, vmax=5 is already quite large.

vthe= Thermal velocity of the electrons (both in initial loading and in injection).

vthi= Thermal velocity of the ions (both in initial loading and in injection).

What follows are the drift velocities imposed upon the various Maxwellians:

....

vdein= Drift velocity imposed on the electrons initially loaded into the simulation region.

vdell= Drift velocity imposed on the electrons injected from the left.

vderr= Drift velocity imposed on the electrons injected from the right.

....

vdiin= Drift velocity imposed on the ions initially loaded into the simulation region.

vdill= Drift velocity imposed on the ions injected from the left.

vdirr= Drift velocity imposed on the ions injected from the right.

....

npe= Number of electrons initially loaded into the simulation region.

npi= Number of ions initially loaded into the simulation region.

NOTA BENE: To run with a single species, do not set npe or npi to 0 (this creates problems with initializations, etc). Rather, use the switches below:

nee= 1, Count the electrons in the dynamics.

nii= 1, Count the ions in the dynamics.

nback= 1, If (and only if) one, but not both, of nee and nii is equal to 0, then for nback=1, the species for which the switch is 0 is replaced by a fixed, uniform background which neutralizes the initial total charge of the other species.

Setting these switches to 0 disables the option.

```
--------------------------------------
      INJECTION PARAMETERS
--------------------------------------
 ninj= 0
 snejl= 0.0  snejr= 0.0  snijl= 0.0  snijr= 0.0
 nres= 1
 fnrel= 5.0  fnrer= 5.0  fnril= 5.0  fnrir= 5.0
 snei= 2.51  nbit= 1  ngau= 0
......................................
```

ninj= 0, No injection from either surface, and no pair creation inside the simulation region.

= 1, Injection or pair creation allowed, as defined by the other parameters.

If nres= 0, then the injection rates are set according to:

snejl= Number of electrons injected per time step from the left boundary.

snejr= Number of electrons injected per time step from the right boundary.

snijl= Number of ions injected per time step from the left boundary.

snijr= Number of ions injected per time step from the right boundary.

Fractional injection rates are allowed (they are realized as a time-average).

If nres=1, then the possibly cumbersome definitions of injection rates given above are replaced by the definition of densities of "plasma reservoirs" left and right of the simulation region. The plasma in these reservoirs has thermal velocities and
drift velocities as defined in the previous group of input parameters. Particles from these reservoirs are assumed to free-stream across the boundaries, and the injection rates are computed to be equivalent to this free-streaming.

nres= 0, No assumption of reservoir, injection set according to snejl, etc.

= 1, Assume reservoirs; ignore previous definitions of snejl, etc, and recompute them to be consistent with the presence of the reservoir.

```
fnrel=  Density (particles per unit area) of electrons in
        the left virtual reservoir.

fnrer=  Density of electrons in the right virtual reservoir.

fnril=  Density of ions in the left virtual reservoir.

fnrir=  Density of ions in the right virtual reservoir.
```

To set any injection to zero, set the corresponding density to zero.
Finally, note that there is a relation between kpack , the number of time-steps between repacking, and the injection rate:
The larger the injection rate, the smaller kpack has to be.
If at any one time-step the injection subroutine attempts to inject more particles than can be accomodated by the arrays (which are limited by npemax and npimax), the injection is inhibited and an error message sent out to the terminal and to the log file "rec".

Finally, note:

```
snei= Number of electron-ion pairs created per time step.
      Creation occurs uniformly at random within the
      simulation region; fractional values of snei
      (even less than 1) are fully allowed.

nbit= 1, Inject the pairs with bit-reversed velocity
         distributions.

ngau= 1, Inject the pairs with a gaussian distribution of
         velocities.  If both ngau and nbit=1, then ngau
         is automatically set to 0.
```

```
--------------------------------------
    PARAMETERS FOR PLOTS
--------------------------------------
 numl= 20
 numx= 1   numy= 1
 vme= 5.0  vmi= 1.0  xwind= 4.0
 kjump= 450
.....................................
 jsnap(1)= 2500  0
.....................................
```

numl= Number of contour lines in contour plots of 2d variables.

numx= Number of cross-sections of fields, along lines parallel to the y-axis. The spacings are set automatically.

numy= Number of cross-sections of fields, along lines parallel to the x-axis. The spacings are set automatically.

xwind= For each cross-section as specified above, the distribution functions of particles in a total width xwind about the line of section are found and displayed (if xwind is larger than the system size, then the distribution functions are for the entire system).

The first kind of snapshots to be specified involve "instantaneous" display of field quantities, of the scatter plots, and of the distribution functions.

kjump= If jsnap(1)= 0 (see just below), then the time-steps for snapshots are set automatically to occur every kjump steps. Upon a restart, counters are reset to 0.

jsnap(1), jsnap(2), . . .   specify the time-step number for snapshots (time step 1 is defined as the first in any given submission of xggd or of the restartable dropfile). One can specify up to jssmax snapshots, where jssmax is defined in the parameter statements in the source file (see above). The value given above, jssmax=50 should more than suffice for any needs! Should one have written by mistake

jsnap(i+1) < jsnap(i), the program will simply never
get to outputting a snapshot for kstep > jsnap(i), so that
no serious error will result (but no output either!).

```
--------------------------------------
  TIME AVERAGING AND OUTPUTTING TO
  A BINARY FILE
--------------------------------------
 naver= 1  nphiav= 1  kskip= 450  jkav= 100
......................................
 javhi(1)= 2500   4990
......................................
 nophi= 1   noden= 1  nopar=1
......................................
 npout= 1575  ihop= 13
......................................
```

The second kind of snapshots to be specified involve display of time-averaged field quantities.

```
         naver= 0, Do not do any time-averaging.
                   This overrides all
                   other parameters and no snapshots of time-averages
                   are produced.

              = 1, Produce a time-average of the potential over
                   jkav  time steps preceding the snapshot time.

        nphiav= 1, Produce graphics of the averaged potential.
                   The option  nphiav= 0  (no graphics)
                   is to be used, for instance,
                   when a large number of frames
                   of the average potential are needed for a movie.
                   In this case, with  nophi= 1 , everything is
                   stored in the binary file "outphi", and can
                   be post-processed later for graphics.
```

The snapshots for the time-averaged quantities are not specified by the array jsnap(1), jsnap(2), . . ., but by the array javhi(1), javhi(2), . . .

kskip= If javhi(1)= 0 (see just below), then the time-steps for snapshots are set automatically to occur every kskip steps. Upon a restart, counters are reset to 0.

jkav= Number of time-steps for a time-average of the potential. Terms in the average are weighted with a so-called "Hamming window", to reduce oscillatory effects from the edges of the time-history.

javhi(1), javhi(2), . . . specify the time-step number for a snapshot of time-averaged quantities. One can specify up to jssmax snapshots, where jssmax is defined in the parameter statements in the source file (see above). The value given above, jssmax=50 should more than suffice for any needs! Should one have written by mistake javhi(i+1) < javhi(i), the program will simply never get to outputting a snapshot for kstep > javhi(i), so that no serious error will result (but no output either!).

The averaging procedure also assumes that:

javhi(i+1) - javhi(i) > jkav

If this difference is positive but smaller than jkav, the averaging will occur over (javhi(i+1) - javhi(i)). If the difference is begative, no subsequent snapshots will appear.

The switches "nophi", etc, provide for saving information in a binary file:

nophi= 1, Save the snapshot in the time-averaged potential in a binary file "ophi..".

noden= 1, Save a snapshot of the y-averaged particle densities, ne(x) and ni(x), in the binary file "oden..".

nopar= 1, Save the particle positions xe-ye and xi-yi for "npout" (see below) particles. These are a sample of every "ihop"-th particle in the simulation, so that the largest index sampled will be npout * ihop. All is saved in the binary file "opar".

npout= Number of particles positions stored in "opar"; we must have npout < npsto (see the definition of npsto in the parameter statement of the source code gdd180).

ihop= Sampling step-size in the original arrays xe(i), ye(i), etc . . .

As usual, setting switches to 0 disables the option.

```
-----------------------------
   SWITCHES FOR GRAPHICS
-----------------------------
 ndens= 1  nfilt= 3
..........................................
 nxye= 1  nxvxe= 1  nxvye= 1  nxvze= 1  nvxvye= 1
 nxyi= 1  nxvxi= 1  nxvyi= 1  nxvzi= 1  nvxvyi= 1
..........................................
 nfex= 1  nfey= 1  nfez= 1
 nfix= 1  nfiy= 1  nfiz= 1
..........................................
 nex= 1  ney= 1  nexf= 1  neyf= 1  neplot= 1
..........................................
 ntest= 0
..........................................
```

For non-output, set the switch to zero, or omit from the data list, as the default for the switch is already zero.

ndens= 1, Output density plots ne(x,y) and ni(x,y).

nfilt= Number of passes of a 1-2-1 digital filter to smooth

```
      density plots (and also rho(x,y) plots).

..........................................................
 nxye= 1, Scatter plot of x-y coordinates for the electrons.
 .          .     .       .       .       .        .
 .          .     .       .       .       .        .
 .          .     .       .       .       .        .
 .          .     .       .       .       .        .
 nvxvyi=1, Scatter plot of vx-vy, for the ions.
..........................................................


..........................................................
  nfex= 1, Plot fe(vx) in sections, for the particles chosen
           over a width  xwind .
  .          .            .         .         .       .
  .          .            .         .         .       .
  nfiz= 1, Plot fi(vz) in sections, for the particles chosen
           over a width  xwind .
..........................................................


..........................................................
  nex= 1, Plot ex fields.

  ney= 1, Plot ey fields.

  nexf= 1, Plot ex Fourier transform.

  neyf= 1, Plot ey Fourier transform.

  neplot= 1, Plot direction field of electric field.
..........................................................



..........................................................
  ntest= 1, Plot the trajectory of a test electron.
..........................................................
```

5. Using DDT to modify parameters:

It is possible to use DDT to modify parameters before re-submitting a halted dropfile. To do this, enter:

```
ddt +xgdda nocopy / t v
```

Some care is required, as with the "nocopy" option, all modifications on +xgdda are final once the ddt session is ended. Also, it is certainly not possible to modify all parameters, as horrible contradictions in definitions may appear! As mentioned above, it is judicious to save dropfiles before re-submission.

I usually only modify the entries of the arrays "jsnap" and "javhi", which determine when a snapshot of the instantaneous or time-averaged state of the system is taken. Upon re-submission of the dropfile, all snapshot parameters are automatically reset to these new values.

All the SWITCHES "nxye", "nxvxe", . . . ,"neplot", "ntest" can be changed in a like manner.

6. Post-Processing for 3d-Plots:

During execution, if the options "naver=1" and "nophi=1" have been set, snapshots of the time-averaged distribution function are stored in the binary file "outphi". This is a cumulative file, which stores the results of all subsequent submissions of the code.

The program "phiin.25" can produce 3d graphics from the information in outphi:

(i) The first step is to adjust the parameters "nx" and "ny" in the parameter statements at the beginning of phiin.25 to precisely those values chosen for gdd180.

(ii) Compile phiin.25 :

```
cosmos phiin.25 ----------> xkkin
```

(iii) Copy the current version of "ophi" (between two restarts of the code) into a file called "otemp"; e.g:

```
copy ophi02 otemp
```

(iv) Submit xkkin : the post-processor will then look for ophi, and prompt the user for information on what to do.

The important variables to specify are:

(i) dx and dy.

(ii) nfilt= Number of times a 1-2-1 digital filter is passed to smooth the spatial profile (default=0).

(iii) na= 0, Non automatic operation: the user has to choose individually the snapshots he wants to output to the f3 file produced by xkkin.

= 1, Automatic operation: all snapshots are consecutively processed.

The other parameters can be left to their default values. The resulting f3 file will contain in succession, for each snapshot analysed:

(i) a header with the number of the snapshot.

(ii) a contour plot of phi(x,y).

(iii) a front 3d view of phi(x,y).

(iv) a back 3d view of phi(x,y).

Finally, I should note that I have developped some MOVIE-MAKING software. As using it requires considerable organization of the computer run, and of the post-processing of the resulting data, I will describe it in a separate note.

6. Execution Times:

Present estimates of execution times for the particle push are,
per particle:

	C and D :	6 microsec	(CFT)
MACHINE -----------	E :	4.5 microsec	(CFT)
	B :	10 microsec	(CIVIC)

The disadvantage of the slower push on the B-machine is strongly mitigated by the fact that jobs can run at much lower priority than on the C and D machines (especially those with large numbers of particles).

==

THE END.

==