LA-UR-85-451

CONF-850266--3

LA-UR--85-451

DE85 007701

TITLE: PARTICLE CODE SIMULATIONS WITH INJECTED PARTICLES

AUTHOR(S): Charles H. Aldrich, X-1

SUBMITTED TO: To be published in the proceedings of the Second International School for Space Simulations, Honolulu, Hawaii, February 3-16, 1985

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

JSW

# PARTICLE CODE SIMULATIONS WITH INJECTED PARTICLES

Charles H. Aldrich
Los Alamos National Laboratory
Los Alamos, New Mexico

ABSTRACT. As problems we are interested in become more complex, we often find our simulations stretching the limits of available computer resources. For example, an interesting problem is simulation of dissipation processes in sub-critical collisionless shocks. To simulate this system our simulation box must contain the shock and its upstream and downstream regions over the entire length of a run. If the shock moves with any appreciable speed the box must then be considerably larger than the shock thickness making it hard to resolve the shock front itself with a reasonable number of grid points.

A solution to this problem is to run the simulation in the frame of reference of the shock. Particles are injected upstream of the shock and leave the simulation box downstream. With the shock stationary in the simulation box, we only need to contain encugh of the up and downstream regions for the fields, etc., to settle down and separate the shock from the box boundaries.

In this tutorial we consider some basic algorithms used in a practical particle injection code, such as the two dimensional WAVE code used at Los Alamos. We will try to present these ideas in a simple format general enough to be easily included in any particle code. Topics covered are:

- Smoothly Injecting Particles.
- Generating the Distribution Functions.
- Time Dependent Injection Density.
- Boundary Conditions on Fields and Particles.
  (Flux and Charge Conservation)

## 1. Introduction

As we have seen in earlier tutorials, particle simulations codes have been an extremely useful tool used to investigate plasma physics. This is particularly true when one wants to understand self consistent shock structures. One problem of interest is how a subcritical shock evolves. The cross-field currents which generate the magnetic field for the shock excite micro-instabilities within the shock front, these instabilities are responsible for the dissipation that allows the shock to exist and strongly effect the basic structure of the shock front including the cross-field currents. In order to resolve both of these processes which are occurring perpendicular to each other (requirring a two dimensional code) we often must make as efficient use of the simulation grid as possible.

Early studies often produced shocks with a simple moving piston. Figure 1 shows a shock created by a piston moving into the box from the right side[1]. Initially the simulation box is full of particles at rest. As the piston moves into the box from the right wall the particles in front of it are reflected to form a beam moving to the left with twice the piston velocity. The front of this beam eventually forms a shock wave that moves through the box slowly separating away from the piston.

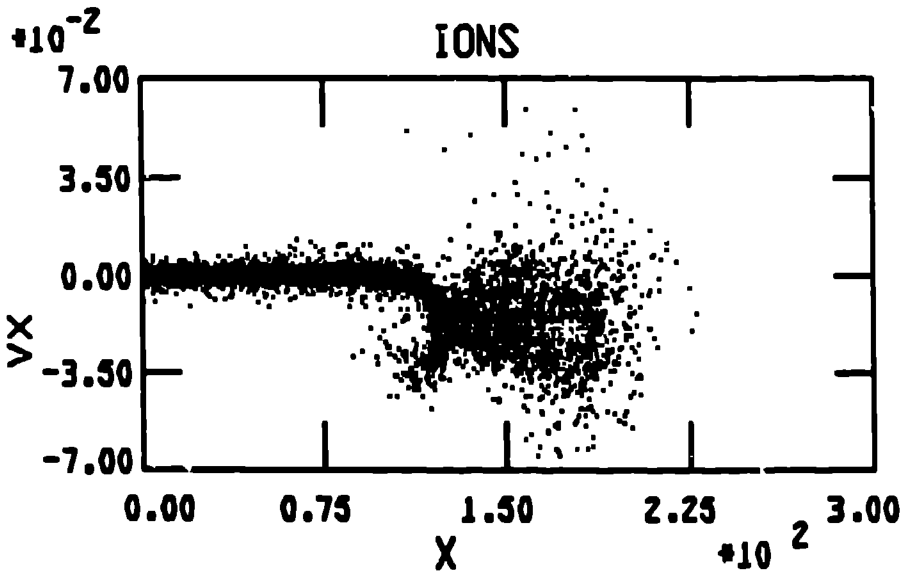

Figure 1. An example of the ion phase space of a shock produced by a piston traveling from the right end wall.

Clearly a way to do this problem making better use of the simulation box would to be running in the frame of reference of the piston. This requires injection. Figure 2 shows a shock where particles are injected on the left hand side of the box. When the problem was initialized the particles in the box were uniformly moving to the right. The particles hit the right hand side of the box and are reflected off a stationary piston and forming a shock downstream. As particles near the left hand side of the box move deeper into the box the space that they vacate is filled with injected particles.

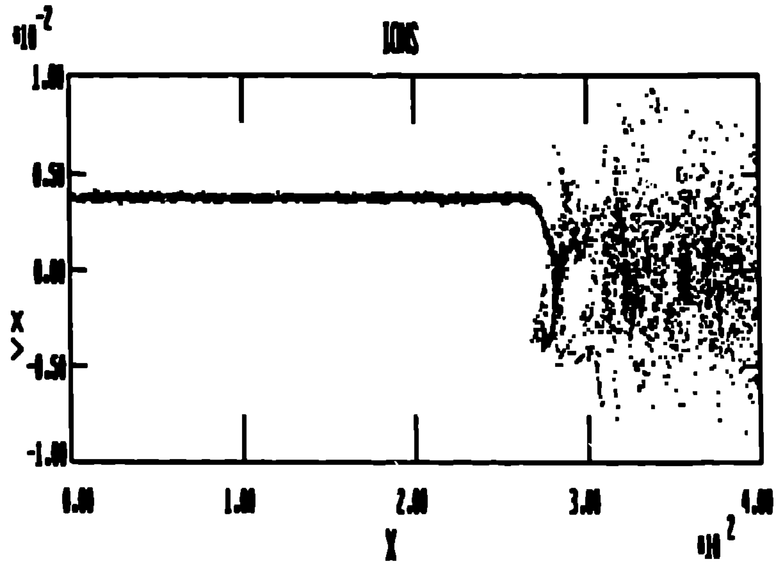

Figure 2. An example of the ion phase space of a shock produced by a piston located at the right wall, with particles injected on the left.

A major problem with the previous two methods is the presence of the piston itself. It takes a long time for the shock front to become distinctly separate from the piston structure. Indeed it may be nearly impossible to separate effects, such as particle heating, occurring at the shock front and the piston from each other. Also, some structures don't lend themselves to formation using such a simple system.

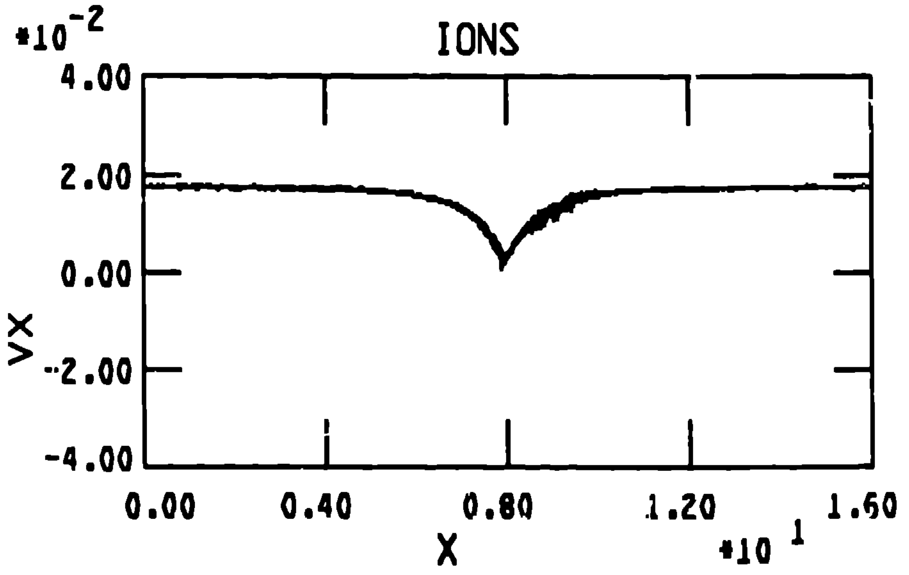

Figure 3. An example of the ion phase space for a perpendicular magnetic soliton held stationary in the simulation box by particle injection.

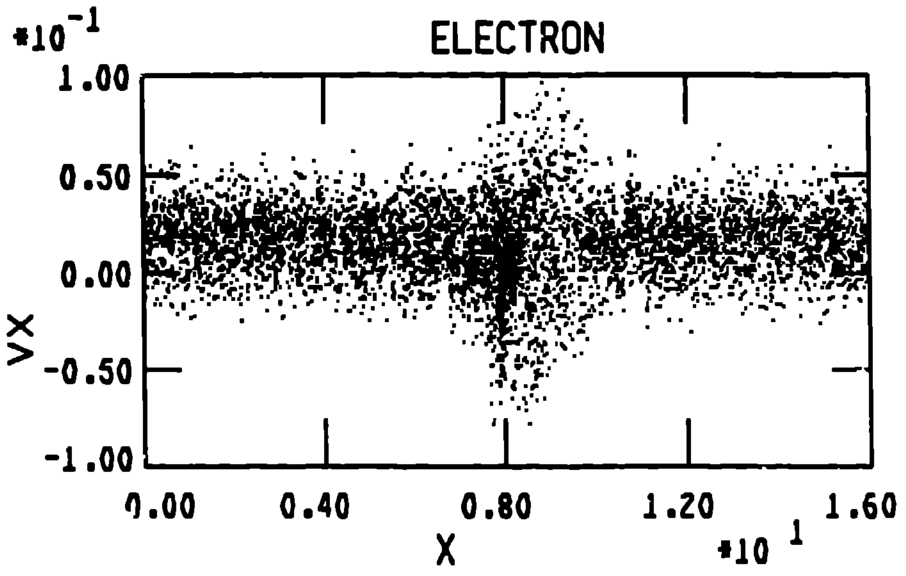

Figure 4. An example of the electron phase space for a perpendicular magnetic soliton held stationary in the simulation box by particle injection.

An example of this structure is a magnetic soliton. Figures 3 and 4 show the phase space for electrons and ions for a perpendicular soliton. The velocity, density, and field profiles for the soliton can be easily calculated from a simple differential equation[2]. When the soliton is initialized at the beginning of the run the known values for these quantities are used and the velocity of the injected ions and electrons is chosen so that the soliton is stationary in the center of the simulation box, with particles injected on the left hand side of the box as before. Note that since the drift velocity for the electrons is smaller than the electron thermal velocity a population of electrons will be moving right to left through the soliton structure, requiring an injection of a few electrons from the right hand side of the box with negative velocities (going left). This configuration is particularly suited for this problem, the soliton is completely stable and will continue to sit stationary in the middle of the box if nothing happens. The cross field currents within the soliton will tend to excite instabilities that in turn cause dissipation. This in turn will change the structure of the soliton evolving the structure towards a shock. Having the soliton stationary allows the simulation to run to long enough times for the instabilities to develop and have some visible effect.

Similarly, a stationary shock structure could be initialized in the simulation box. The velocity, density and field jumps across the shock front can be calculated using the Rankine-Hugoniot (R.H.) equations[3]. The shock speed calculated from the R.H. equations can be used to set the injection speed to make the shock stationary in the simulation box.

In this tutorial we start off in section 2 by describing two ways to produce a system of simulation particles according to some given distribution function. In section 3 we determine how many particles are crossing the simulation boundary and how to put the particles into the simulation box. This includes a discussion of how to choose the velocities of the injected particles randomly to produce the probability distribution we want in the bulk of the box. We then briefly consider injecting a beam with a time and/or spatial variation in section 4. We finish with section 5 with a couple of cautions with respect to the boundary conditions on the particles and electromagnetic fields.

## 2. Throwing the Dice - the Probability Distribution Function.

In the next two subsections we will consider the general problem of throwing random numbers to obtain a set of particles distributed according to some given distribution function. We need this to create the particles as they cross the boundary - as well as when the initial particles are put in the simulation box. The first method maps a uniform dice into a more complex distribution using an analytic formula. The second approach considered uses a monte carlo method where throws of the dice are kept or tossed out according to a rule.

### 2.1. Analytic.

One important distribution function used in most simulations represents a group of particles in thermal equilibrium with some thermal velocity $v_{therm}$. In suitable units this can be written (in two dimensional cartesian coordinates),

$$P(v_x, v_y) \sim exp(-(v_x^2 + v_y^2)/v_{therm}^2)dv_x dv_y, \tag{1}$$

or in cylindrical coordinates,

$$P(v,\theta) \sim v\exp(-v^2/v_{therm}^2)dvd\theta, \tag{2}$$

where $v$ is the particle velocity and $\theta$ is the angular coordinate.

Most computers have a random number function that will throw a pseudo-random number with uniform probability between the limits 0.0 and 1.0. Lets denote this function by rand(). As an example will construct an analytic formula to map this uniform distribution into a gaussian distribution.

Consider a variable $\varsigma$ bounded in the range $0 \leq \varsigma \leq 1$. If we choose values for $\varsigma$ with uniform probability within this range, ($\varsigma$ = rand()), then the probability of landing in some range of values $d\varsigma$ wide is simply $d\varsigma$ and the probability of having a value less than $\varsigma$ is $\varsigma$.

Next, consider another variable $v$ that is bounded in some range $0 \leq v \leq v_{max}$. The two variables are related by a single valued function $f$ such that each point in $\varsigma$ is mapped to one and only one point in $v$ and the order of the points are preserved,

$$v = f(\varsigma). \tag{3}$$

This is shown schematically in figure 5. By drawing several equal width intervals on the $\varsigma$ axis and mapping them onto the $v$ axis using the graph in figure 5, one can see that points equally spaced on the $\varsigma$ axis will be more densely on the $v$ axis near $v = 0$. More quantitatively, the probability of any point being between 0 and $v$ on the v axis ,given equation 3, must equal the probability of landing between 0 and $\varsigma$ on the $\varsigma$ axis. Since the $\varsigma$ values are all chosen with equal weight, this probability is $\varsigma$, or in terms of $v$, $f^{-1}(v)$, where the -1 denotes the inverse function. The density

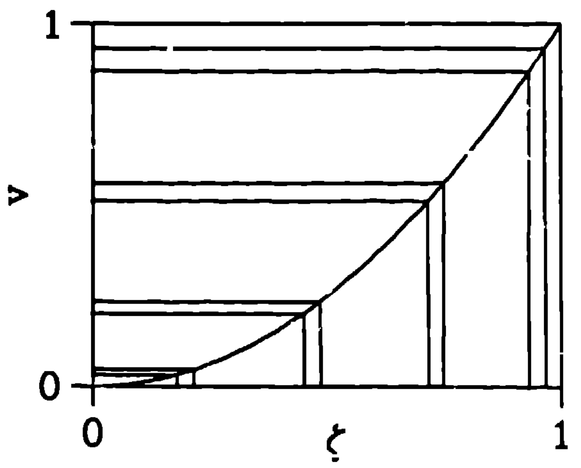

Figure 5. Example of an analytic function used to map a distribution function from one probability space to another. (see section 2.1).

function $P(v)$ is simply the derivative of this quantity, so the probability of landing a point in some range $dv$ can be written,

$$P(v)\,dv = \frac{\partial f^{-1}(v)}{\partial v}\,dv. \tag{4}$$

Returning to the gaussian distribution, we can let

$$v = f(\varsigma) = \sqrt{-\log(1-\varsigma)}, \tag{5}$$

or

$$\varsigma = f^{-1}(v) = 1 - \exp(-v^2). \tag{6}$$

The probability of being in some range of values $dv$ is then , by equation 4,

$$P(v)\,dv = 2v\exp(-v^2)\,dv. \tag{7}$$

This equation has the form of equation 2. The velocity vector can point randomly in any direction, so the angular variable is equally probable in the range 0 to $2\pi$. To set up a set of particles according to the gaussian distribution we can set,

$$\begin{aligned} v_x &= v\sin(\theta), \\ v_y &= v\cos(\theta), \end{aligned} \tag{8}$$

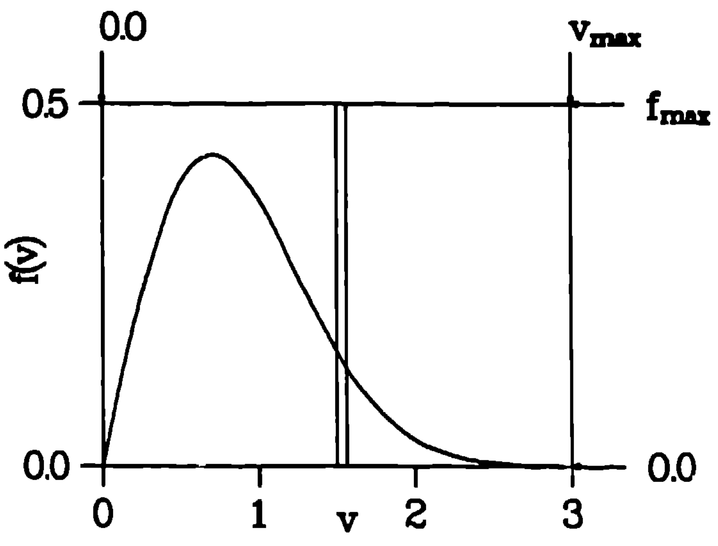

Figure 6. Graph used to produce the distribution function $f(v)$ using numerical monte-carlo method (section 2.2).

where,

$$v = v_{therm}\sqrt{-\log(1-\text{rand}())},\qquad (9)$$
$$\theta = 2\pi\ \text{rand}().$$

For an arbitrary distribution function $P(v)$ we can integrate equation 4 to obtain $\varsigma = f^{-1}(v)$ and then invert to obtain the mapping function. As we have just seen, the gaussian distribution turns out to be easy to generate in cylindrical coordinates. If we try to generate a gaussian in three dimensions, for example, using these methods directly, the transformation function $f^{-1}$ will contain error functions and the like which are hard to invert in order to obtain $f(v)$. The solution to this problem is to throw two out of the three coordinates, say $x$ and $y$ with the cylindrical formula, equation 8. We can then use equation 8 again to throw two more coordinates, one of which is thrown away while the other is used for $z$. If one is simply interested in a one dimensional gaussian, just throw away the coordinate not needed.

## 2.2. Numeric - Monte Carlo.

Often the distribution function we want is too complex to integrate the equations in section 2.1 analytically. In this case we can resort to a more general method that will work for almost any distribution function. In this method we will be tossing the dice and either keeping the results as our answer or throwing out the toss and redoing it until the toss fits certain rules that we arrange. As an example we can generate the distribution in equation 2. The first step in this process is to represent the function graphically in a box of some finite size. This is done in figure 6, where the high velocity tail of the distribution is being ignored ($v_{therm} = 1.0$).

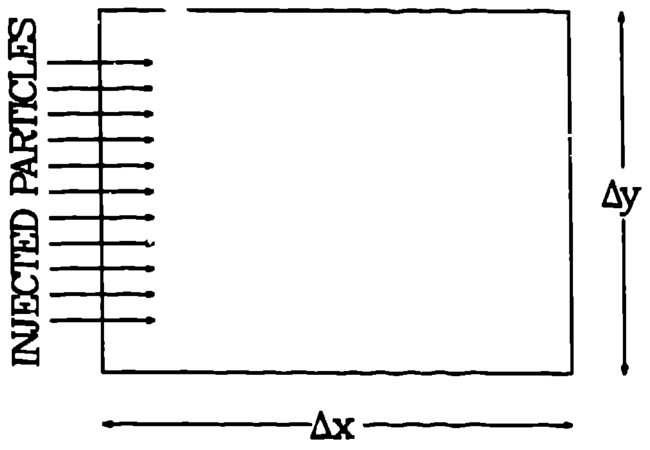

Figure 7. Simulation box with injected particles.

We can split the graph up into many vertical bins, one of that is shown in the figure 6. If we randomly choose a value of $v$ within the graph (between 0 and $v_{max}$ the chance of it being in a particular bin is $dv/v_{max}$. Throwing another number representing the vertical position on the graph ( between 0 and $f_{max}$) the fraction of points that will hit under the curve is $f(v)/f_{max}$. Thus by throwing two random numbers for each point and keeping only points that fall under the curve we fill each bin with an average of

$$N_{bin}(v) = \left(\frac{f(v)dv}{v_{max}f_{max}}\right)N_{thrown} = f(v)dvN_{kept}, \tag{10}$$

where $N_{kept}$ is the number of points that are saved. The probability of being in a bin is then $f(v)$. This means that if we randomly throw points onto the graph keeping the first one that hits under the curve $f(v)$ results in generating the distribution we want. This method has the advantage of being simple to implement and have working on a computer in little time.

## 3. Smoothly Injecting The Particles

In this section we will discuss how to inject a beam of particles into the left hand side of the simulation box as used to create the soliton in figures 3 and 4. This is shown diagrammatically in figure 7 to establish the notation for the rest of the paper. Outside the box the particles are assumed to in thermal equilibrium with some thermal velocity $v_{therm}$ and moving with some drift velocity $v_{beam}$. The distribution function can be written as a gaussian ,offset in velocity,

$$f(v) \sim \exp(-(v_x - v_{beam})^2 - v_y^2 - v_z^2)/v_{therm}^2)). \tag{11}$$

## 3.1 How Often?

We can relate the average time between injections to the drift velocity and injection density. If a particle has velocity $v$ it will travel $vdt$ in one time step. The boundary will have a rectangle of size $\Delta y$ by $vdt$ full of particles sweep across in one time step(see Figure 8).

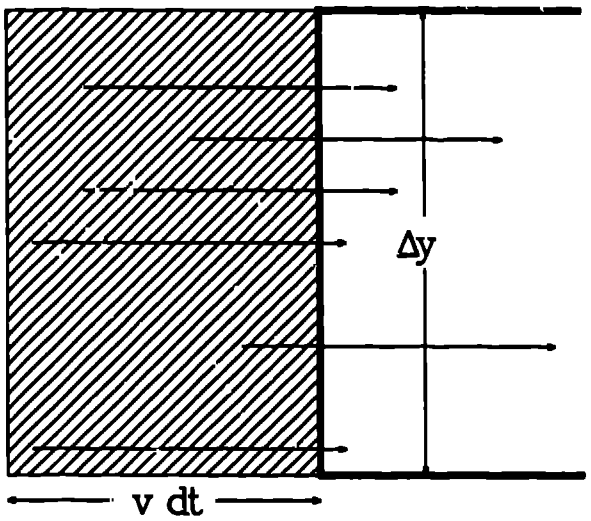

Figure 8. Particles with velocity $v_x$ in the shaded area are swept across the boundary in one time step.

Thus the number of particles in this region is the particle density $f(v)\ d^2v$ (assuming a constant injection density along the y direction) times the area $\Delta y\ vdt$, where $f(v)$ is the distribution function for the injected particles, or,

$$\Delta y\ vdt\ f(v)\ d^2v. \tag{12}$$

Integrating over the entire velocity space, the total charge crossing the boundary is then,

$$Q = \rho \Delta y\ dt < v_x > . \tag{13}$$

where,

$$< v_x > = \int v_x\ f(v)\ d^2v. \tag{14}$$

For our injected beam $< v_x >$ can be calculated analytically, giving,

$$< v_x >= \frac{1}{2}\left(\frac{v_{therm}}{\sqrt{\pi}} \exp\left(\frac{-v_{beam}^2}{v_{therm}^2}\right) + v_{beam}\left(1 + \text{erf}\left(\frac{v_{beam}}{v_{therm}}\right)\right)\right). \tag{15}$$

If the charge on a single simulation particle is $q_{species}$ then we find the number of simulation particles crossing the boundary per time step,

$$N = \frac{\rho \Delta y \; dt < v_x >}{q_{species}}, \tag{16}$$

or equivalently, the average time between injecting two particles,

$$dt_{inject} = \frac{q_{species}}{\rho \Delta y < v_x >}. \tag{17}$$

## 3.2 Where?

As far as the injection algorithm is concerned any particle code can be viewed as a series of black boxes,

- Calculate Fields and Potentials (Forces on particles).
- Push particles with Forces, Calculate Charges.
- Inject Particles.

These steps are repeated over and over again, once for each time step. Each cycle steps a time $dt$, the nth step beginning at time $t = (n-1)dt$.

To perform particle injection in a smooth manner we wish to decouple the injection process as much as possible from the size of the time step. To do this we inject the particles uniformly in time separated by time $dt_{inject}$. So that we inject the mth particle at time $t_{inject} = m dt_{inject}$. The time between particle injections may be a fraction of a time step $dt$ or many time steps. Since these two characteristic times are not necessarily multiples of each other the time when a each particle is injected will lie somewhere in the middle of a time step. We can move the particle into the box for this time period, $dt_{move} = t - t_{inject}$. If many particles are injected in one time step this will spread them out evenly in space.

To isolate the simulation problem from effects of the boundary conditions most calculations will be set up so that very little is happening at the boundaries of the box, that is no steep density gradients, large fields, etc. are allowed. The time step is usually small enough so that the particles travel only a small distence into the box. It is usually a good approximation to ignore forces on the particle for this fractional part of a time step and simply move them into the box by setting;

$$\begin{aligned} x_{inject} &= dt_{move} v, \quad and \\ y_{inject} &= \Delta y \; \mathrm{rand}(). \end{aligned} \tag{18}$$

This formula could be changed to inject relativistic particles by using the momentum. One could also be completely consistent and move the injected particles with the same particle pusher used in the second step above.

## 3.3 Distribution Function.

From equation 12 we can see that we have to pick the x velocities according to the probability formula,

$$f(v_x) \sim v_x \exp(-(v_x - v_{beam})^2/v_{therm}^2). \tag{19}$$

This formula is not easy to generate analytically and the techniques of section 2.2 are used to produce this distribution. For a full three dimensional velocity space the other two directions are a simple non-offset gaussian considered in section 2.1. Thus

$$\begin{aligned} v_y &= v_{par}\sin(\theta), \\ v_z &= v_{par}\cos(\theta), \end{aligned} \tag{20}$$

where,

$$\begin{aligned} \theta &= 2\pi \text{ rand}(), \\ v_{par} &= \sqrt{-\log(1-\text{rand}())}. \end{aligned} \tag{21}$$

## 4. Nonuniform Densities.

Generalizing this injection scheme to nonuniform density profiles both in space and time are relatively straightforward. We will quickly consider injecting a shaped beam and a pulsed injection.

### 4.1 A Shaped Beam.

To produce a uniform simulation across the side of the simulation box the y position was chosen randomly with equal probability of entering anywhere. To change this profile we can choose values for the coordinate in the y direction with one of the methods of section 2. For example if we wanted a beam whose density is zero at the top and bottom edges like,

$$\rho(y) = \rho_{max}\sin(y\pi/\Delta y) \tag{22}$$

Let $v = \Delta y\ y/\pi$ and therefore, from equation 4,

$$\frac{\partial f^{-1}(v)}{\partial v} = \frac{\sin(v)}{2}, \tag{23}$$

or, integrating,

$$\varsigma = f^{-1}(v) = \frac{Constant - \cos(v)}{2}, \tag{24}$$

and

$$\varsigma = \frac{1-\cos(\Delta y\ y/\pi)}{2}, \tag{25}$$

We can then choose y according to,

$$y = \frac{\Delta y}{\pi}\cos^{-1}(1-2\text{rand}()). \tag{26}$$

where rand() is the uniform generator mentioned earlier.

The average density $\rho$ used in equation 17 for the time between injections $t_{inject}$ becomes

$$\rho = \frac{2}{\pi}\rho_{max}. \tag{27}$$

## 4.2 Pulses.

Time variations in the injection parameters can be accomplished in several ways. If the time step is small enough and the variations in the injection velocity or density is slow enough to be resolved one can simply change the parameter as a function of time at each time step and use the new $dt_{inject}$ in equation 17 to inject the next set of particles.

If on the other hand we would like to inject a fast changing density profile, for instance, we can change the time a particle is injected to a nonlinear function of time. If we would like to have one pulse of the shape,

$$\rho(t) = \rho_{max}\sin^2(t\pi/T_0), \tag{28}$$

where $T_0$ is the length of the pulse, we can let

$$t_{inject} = T(m\ dt_{inject}), \tag{29}$$

where

$$T^{-1}(t) = \frac{t - \frac{T_0}{2\pi}\sin\left(\frac{2\pi}{T_0}t\right)}{2}. \tag{30}$$

In general for arbitrary $T(t)$, the density injected will be

$$\rho(t) = \rho\ \frac{\partial T(t)}{\partial t}. \tag{31}$$

Note for $T(t) = t$ then $\rho(t) = \rho$.

## 5. Boundary Conditions

Except for the process of injection itself the boundary conditions are not significantly differently from a particle code without injection. There are a couple of points worth noting in the following two subsections.

### 5.1 Fields

If we are simulating a shock structure and transforming into the frame of reference of the shock we have to be careful of the boundary conditions on the fields. In the frame of reference of the upstream particles we have some magnetic field $B_{upstream}$. Since the particles have zero velocity in this frame, they feel no electromagnetic force. Transforming to the frame of reference of the shock one has to be aware of how the fields transform with changes of reference frame. That is, if the shock is moving with velocity $v_{shock}$ in the upstream frame , an electric field component $\vec{E}_{upstream} = \vec{v}_{shock} \times \vec{B}_{upstream}$ will be present in the simulation frame upstream of the shock and at the boundaries, resulting again in zero net force on the particles. Neglecting this "flux conservation boundary condition will result in an electromagnetic disturbance moving in from the boundary in a fairly obvious way.

### 5.2 Particles

In figures 3 and 4 we showed the phase space for a subcritical perpendicular magnetic soliton, where both electrons and ions are injected simultaneously. The drift speed of the electrons is less than the electron thermal speed but much greater than the ion thermal speed. The right hand boundary can then be described as

- Let all ions leave the box.
- Let all electrons leave the box.
- Inject a few electrons moving to the left.

The electrons are injected with equation 19. This process is the same discussed earlier except that the sign of v is different and we are injecting the backward going tail.

Statistically the number of electrons crossing the boundary during any particular time step will fluctuate between time steps causing a corresponding fluctuation in the charge imbalance at the boundary. This often can create havoc with the code. To essentially guarantee charge neutrality at the boundary we can move the ions first, note how many ions leave the system, then move the electrons. When we move the electrons, if an electron crosses the boundary we let it leave if and only if there is a positive charge imbalance at the boundary. Since the thermal is much higher for the electrons, many more electrons cross the boundary in each time step. Once enough electrons leave to assure charge neutrality at the boundary the rest are injected according to the distribution, equation 31. This virtually assures neutrality except in the case of large fluctuations in the number density which will be rare.

## Acknowledgement

This work was supported by the United States Department of Energy.

## References

1. D. W. Forslund, K. B. Quest, J. U. Brackbill, and K. Lee. *Journal of Geophysical Research*,89 ,2142,1984.

2. D. A. Tidman and N. A. Krall, *Shock Waves in Collisionless Plasmas*, John Wiley and Sons,Inc.

3. D. A. Tidman and N. A. Krall, *Shock Waves in Collisionless Plasmas*, John Wiley and Sons,Inc.