One Dimensional Electrostatic Plasma Simulation

An application of the particle-in-cell technique

This experiment concerned the use of two closely related particle-in-cell simulation techniques, and their comparison with eachother and with the theoretical results for the behaviour of standing waves in the plasma charge distribution. The two techniques compared were the cloud-in-cell (C-in-C) and nearest grid point (NGP) particle distributions. It was found that, while the C-in-C was invariably the slower method, it was generally the more accurate for the extra time it required. Landau damping of the standing waves was observed and the damping behaviour was compared with that predicted by theory, with mixed results.

Andrew N Jackson 22nd January 1996

Introduction

The solid, liquid and gaseous states of matter which occur at the surface of our planet are not typical of matter in the universe at large. Most of the visible matter in the universe exists as plasma, whereas lightning and the aurora are the only natural manifestations of the plasma state on earth.In a plasma, the energy of the particles is so great that the electric forces which bind the atomic nucleus to its electrons are overcome. Its behaviour is interesting in that while the waves that occur in it have much in common with waves in electrically conduction solids, it's bulk behaviour is best described using terminology and equations developed for fluids. These unique characteristics have lead to plasmas being referred to as a fourth state of matter.

The sun, like most stars, is composed of plasma; in its core, the forces are so great that the atomic nuclei, dissociated from their electrons, can overcome their mutual repulsion and fuse together. The attempt to reproduce on Earth the process of controlled thermonuclear fusion, coupled with the need to understand the behaviour of plasma in space, is at present the main impetus for research in plasma physics.

A plasma is an electrically neutral conducting gas, consisting of positive ions and electrons in a state of quasi-neutrality. ie:

$$n_e \approx Zn_i$$

where n_e and n_i are the electron and ion densities respectively and Z is the average ion charge number. As any significant charge imbalance generates large space charge fields, quasi-neutrality is maintained.

This experiment is concerned with the propagation of waves through the plasma, and specifically with the damping of these waves due to the thermal vibrations of the plasma's constituent particles (Landau damping). Under these conditions we can ignore the bulk fluid behaviour and focus on the electrical conduction behaviour. In other words, the behaviour of the plasma is determined by the aggregate electric fields generated by the charges, as opposed to binary collisions, as in the case of a fluid.

This means that if we neglect any relativistic effects, the plasma's general behaviour can be determined by solving Maxwell's and Newton's equations. The basic simulation techniques do this directly, not by considering the full set of particles, but a relatively small number of 'super-particles', each of which is equivalent to a vary large number of electrons or ions.

Theory

• Basic Physical Plasma Theory

The plasma simulation has a particularly simple form in one dimension where magnetic fields are not generated. This means that only electro-static longitudinal waves occur, and that Maxwell's equations are reduced to Poisson's equation:

$$\frac{dE}{dx} = \frac{\rho}{\varepsilon_0}$$

where ρ is the charge density;

$$\rho = e(Zn_i - n_e).$$

The electrons and ions are represented by super-particles each corresponding to a large number of electrons or ions, f_e and f_i respectively.

The equations of motion for a super-particle are

$$\frac{dv}{dt} = \frac{qE}{m}$$

and

$$\frac{dx}{dt} = v$$

where q and m are the super-particle charge and mass respectively, corresponding to an electron or ion super-particle.

• Particle-In-Cell Simulation Theory

The first step in the particle-in-cell simulation is to break the plasma domain down into a finite difference mesh over which the electric field is calculated. In this case we choose a series of *it* cells in x each of width ▲x, so that the mesh points (cell-centres) are at

$$x_i = (i+{}^1/_2)\blacktriangle x \qquad 1 \le i \le it$$

and the cell boundaries at

$$x_{i+½} = (i+1)\blacktriangle x \qquad 0 \le i \le it$$

where i is an integer.

If the charge difference in the cell is known the field at the boundaries follows from

$$E_{i+½} = E_{i-½} + \rho_i \blacktriangle x/\varepsilon_0$$

and the field at the cell centre

$$E_i = ½(E_{i-½} + E_{i+½}).$$

The recurrence relation for E is started off by assuming the boundary condition:

$$E_{-½} = 0.$$

The inaccuracy of this assumption can be corrected by using the following condition on the potential drop over the system:

$$\phi = \sum_{i=1}^{it} E_i \blacktriangle x = 0$$

This is used to calculate a uniform field to be added to the charge determined component.

The charge density ρ is determined by assigning the super-particle charge on to the mesh using a weighting function δ_i such that

$$\rho_i = \frac{1}{\blacktriangle x}\sum_{j=1}^{NT} \delta_i(x_j) q_j$$

where x_j and q_j are the position and charge of particle j of NT particles. In practice two very simple weights are used:

• 1: Nearest Grid Point (NGP)

$$\delta_i(x) = \begin{cases} 1 & \text{if } |x-x_i| < ½\Delta x \\ 0 & \text{otherwise} \end{cases}$$

This assigns all the charge to the cell in which the particle lies.

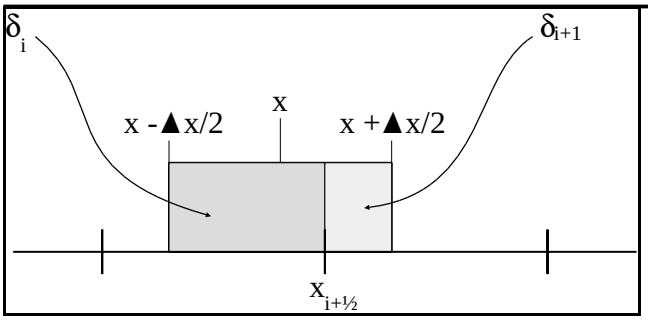

Figure 2-1.

• 2: Cloud-in-Cell (C-in-C)

$$\delta_i(x) = \begin{cases} [1 - |x-x_i| / \Delta x] & \text{if } |x-x_i| < \Delta x \\ 0 & \text{otherwise} \end{cases}$$

This assigns the charge over two cells as if the super-particle formed a uniform distribution of width Δx (see figure 2-1).

Given our finite difference form for the evaluation of the electric field over the grid, we now need a finite difference solution for the motion of the particles. Due to the relatively large mass of the ions, we can assume then to be stationary, and so we only need to worry about the electrons. Their motion can be calculated by the following leapfrog method:

$$v_j^{n+½} = v_j^{n+½} + q_j/m_j \,.\, E_j^n \,.\, \Delta t$$

$$x_j^{n+1} = x_j^n + v_j^{n+½} \,.\, \Delta t$$

The field at the particle is calculated from the fields at the cell centres using the same weighting functions as for assigning the charge:

$$E_j^n = \sum_{i=1}^{it} \delta_i(x_j^n) E_i^n$$

This means that, given a set of charge positions x_j^n and velocities $v_j^{n+½}$ from the proceeding time step, the calculation follows the sequence:

$$x_j^n \text{ » } \rho_i^n \text{ » } E_i^n \text{ » } v_j^{n+½} \text{ » } x_j^{n+1}$$

The time-step Dt is limited by the explicit nature of the algorithm, and the existance of wave solutions whose frequency is:

$$\omega_p = \sqrt{\frac{n_e e^2}{\varepsilon_0 m}}$$

• Landau Damping

In order to observe Landau damping we must first modify the simulation to allow it to simulate simple plasma waves. There are two main points to this modification:

• The system is periodic, ie

$E_{-½} = E_{it+½}$

with zero potential drop over the mesh.

• Electron super-particles leaving one boundary of the mesh re-enter through the other, with no change in their velocity, ie:

$$x \text{ » } x - l \quad \text{if } x>l$$
$$x \text{ » } x + l \quad \text{if } x<0$$

where $l = (it \times \Delta x)$, ie the total mesh length.

With these modifications the only further requirement is that the initial positions of the electron super-particles are defined such that they form a wave which can then be Landau damped. In other words, while the ion density is uniform, the electron density has a sinusoidal perturbation superimposed on the uniform background. The particles are given velocities at random according to a Gaussian distribution of variance kT/m, where T is an appropriate plasma temperature (~10^4 K), k is Boltzmann's constant and m is the mass of the electron.

This system will oscillate with a frequency:

$$\omega_0 = \sqrt{\omega_p^2 + 3k^2 v_{th}^2}$$

Which is based on the natural plasma frequency above plus a correction for the effect of the thermal velocities of the particles. The waves

are stationary if the temperature is zero, but progressive if warm.

For Landau damping to occur, the energy associated with the wave must accelerate particles whose random velocity is close to the wave phase velocity. This resonance bleeds energy from the wave and into the general electron background. For this to happen the spatial frequency of the waves $k\lambda_D$ must be greater than or equal to 0.2, where λ_D is the Debye length, defined as:

$$\lambda_D = \sqrt{\frac{\varepsilon_0 kT}{n_e e^2}}$$

This damping effect is not easily measured; we must monitor the damping of the field energy in the specific Fourier mode, and the initial wave amplitude must be small. If the amplitude is larger then particle trapping may occur, leading to the decay of the wave via breaking.

As mentioned above, the Landau damping effect is only significant for certain spatial frequencies of wave. The degree of damping can be ascertained by fitting an exponential curve of the form

$$E = E_0 e^{-\gamma t}$$

to the results. According to theory the relationship between the damping factor γ and the wave number k should be:

$$\gamma = \sqrt{\frac{\pi}{8}} \left(\frac{\omega_p}{\lambda_D^3 k^3} \right) \exp - \left(\frac{1}{2\lambda_D^2 k^2} + \frac{3}{2} \right)$$

And the experiment should confirm this.

Method

One technique to minimise the amount of work a program will have to do is to use dimension-less variables, and this is possible in this case, using the following relations:

$T = \omega_p t$ $\quad DT = \omega_p \Delta t$
$X = x/\lambda_D$ $\quad DX = \Delta x/\lambda_D$
$V = v/v_t$
$\lambda_D w_p = v_t$

So that the finite difference equations (cf page 3) become:

$$E_{i+½} = E_{i-½} + N_i$$
$$V_j^{n+½} = V_j^{n+½} + E_j^n \cdot \frac{DT \cdot DX}{SPperI}$$
$$X_j^{n+1} = X_j^n + V_j^{n+½} \cdot DT$$

Where N_i = the number of super-particles in cell i.
SPperI = the average number of super-particles in a cell.

This alter the initial conditions such that the values of X lie in the range 0 to λ/λ_D multiplied by the number of periods we are describing, and the values of V are those of a random Gaussian distribution of unit variance.

λ/λ_D is the now the parameter that decided on the behaviour of the system. Previously we said that the condition for Landau damping was:

$$k\lambda_D \bullet 0.2$$

As $k = 2\pi/\lambda$ this becomes:

$$\lambda/\lambda_D \rfloor 2\pi/0.2$$
$$\rfloor 31.141$$

And so the range we are interested in investigating is $\lambda/\lambda_D = [0 - 35]$.

This dimensionless approach also means that our forlumae for expected results become:

$$\tau_0 = 2\pi \cdot [1 + 12\pi^2 \cdot (\lambda/\lambda_D)^{-2}]^{-½}$$

and,

$$\gamma = \sqrt{\frac{\pi}{8}} \left(\frac{1}{\lambda_D^3 k^3} \right) \exp - \left(\frac{1}{2\lambda_D^2 k^2} + \frac{3}{2} \right)$$

Weighting Function Implementation

We have said that when we assign the charges to the grid we can use one of two weights, the Nearest Grid Point and Cloud in Cell distributions. These are implemented as follows:

• *NGP* : a particle at x goes into cell i such that:
$i = INT(X_j/DX)$

• *CinC* : a particle at x goes into two cells:
Let $i = INT(X_j/DX + ½)$
so $i - X_j/DX + 0.5$ goes into cell i,
and $1 - (i - X_j/DX + 0.5)$ goes into cell i+1.

Fourier Mode Analysis

The calculation of the system energy is done by analysing the Fourier mode of vibration. The formula required for this is as follows:

$$a_k = 1/(2\pi IT) \geqslant_i E_i \cos[2\pi.i.k/IT]$$
$$b_k = 1/(2\pi IT) \geqslant_i E_i \sin[2\pi.i.k/IT]$$
$$\text{Energy} = \bullet(a_k^2 + b_k^2)$$

General Form Of Calculation Routine

The finite element calculation was implemented using the following general structure:

Initialise electric field variables:

$E_{-½} = 0$, $E_{res} = 0$.

•

Loop over all mesh elements(i):

$E_i = E_{-½}$

$E_{-½} = E_{-½} - N_i$

$E_i = ½(E_i + E_{-½})$

$E_{res} = E_{res} + E_i$

$N_i = -SPperI$

End loop.

•

$E_{res} = E_{res}/it$

$DXoSPperI = DX/SPperI$

$len = DX \times it$

•

Loop over all particles (j):

Use weighting function to find electric field E_x

$V_j = V_j - DT \times DXoSPperI \times (E_x - E_{res})$

$X_j = X_j + DT \times V_j$

IF $X_j > len$ THEN $X_j = X_j - len$

IF $X_j < len$ THEN $X_j = X_j + len$

Use weighting function to find relevant cell numbers and full up N_i accordingly.

End loop.

•

Calculate new system energy.

This is repeated as many time as is requested by the user, in terms of real time converted into no's of iterations.

General I/O Requirements:

As well as the actual calculation routines, a number of input/output routines were also written. The actual details of these are not important but it should be said that they allow the entry of the important parameters of the calculation of presented output at various stages during the simulation. The output was in the form of various graphs:

- The resultant electric field,
- A phase space plot of the particles,
- A frequency plot of particle velocity, and,
- A plot of system energy against time.

All this could also be output to a file for rendering in a desk-top-publishing program via a data plotter (GNUPlot).

All important code is given in the appendix at the end of this report.

Method Of Analysis

The analysis of the program breaks down into four parts:

1 • General plasma behaviour:

Using the NGP program, analyse the general behaviour of the plasma, by obvserving the changes in the electric field as time passes.

2 • Comparison of weighting functions:

Collect a set of results from both the NGP and the CinC program for a range on accuracies. In other words see how altering the number of particles used for the simulation alters the result and try to determine which routine is the best.

3 • Comparison of experimental results to theoretical predictions:

Using the best program, collect data on the damping and period of the oscillations for a range of λ/λ_D. Then compare these results to those predicted by theory.

4 • Effect of wave amplitude:

Alter the initial conditions so that results are obtained for a whole range of amplitudes (given as a fraction of the average number of particles in a cell). Use velocity frequency plot to illustrate these results.

Results

1 • General plasma behaviour:

The sequence of figures from 4-1 to 4-4 illustrate the general chronological evolution of the plasma's electric field. Figure 4-5 shows the change in the system's energy over time, and the points on that graph which correspond to figures 4-1 to 4-4. It is clear from this that the second peak on this graph is lower than the first, and so Landau damping must be occuring. This damped oscillation trend continues at times beyond those shown in fig. 4-5.

All results were taken at $\lambda/\lambda_D = 15.0$.

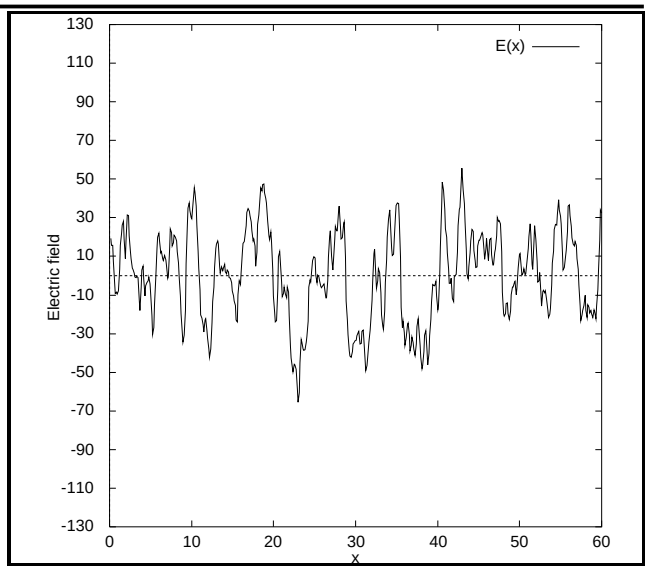

Figure 4-3

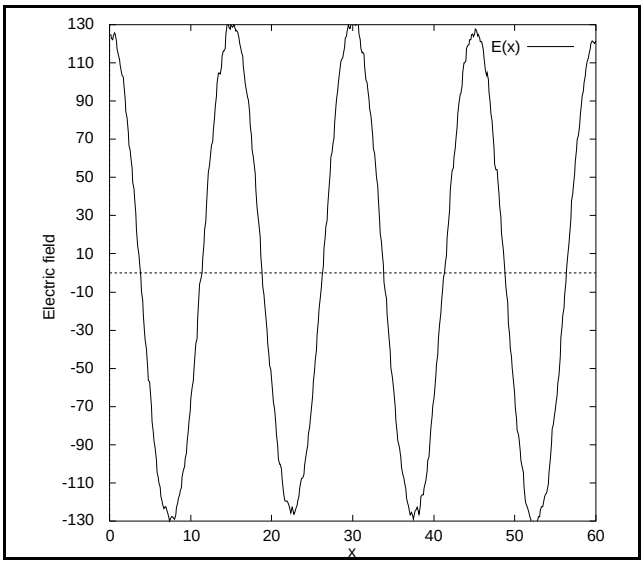

Figure 4-1

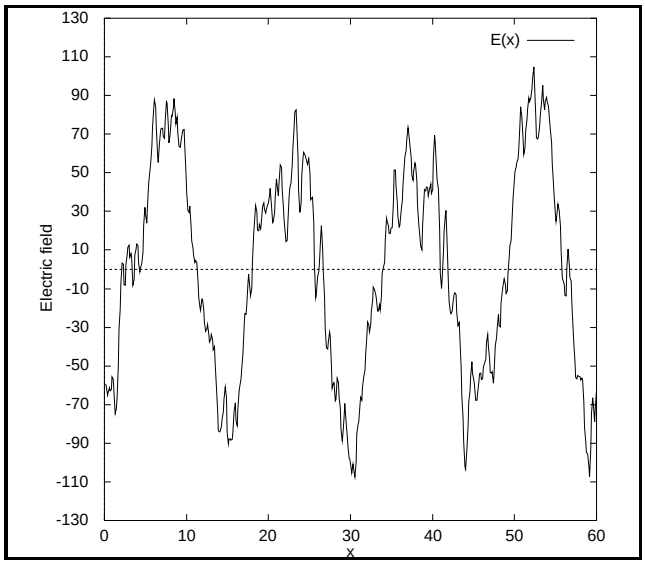

Figure 4-4

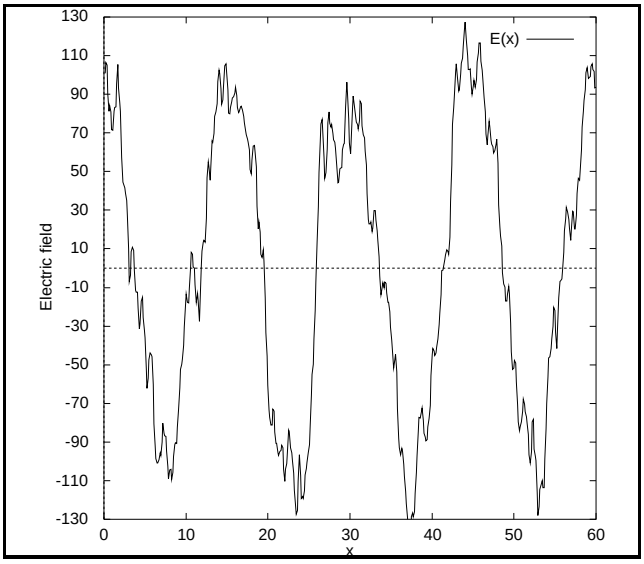

Figure 4-2

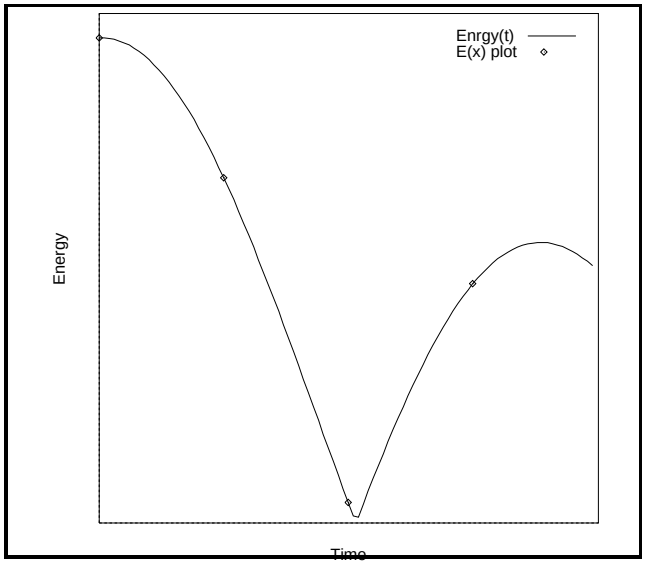

Figure 4-5

2 • Comparison of weighting functions:

For both my NGP and my C-in-C programs, you were allowed to use up to 100,000 particles for the simulation, and to select what fraction (f) of this number to use for any particular simulation.

While results were taken for the whole range of f (0.0-1.0), the results of the comparison can be clearly seen from the results at f = 0.1, 0.2, 0.4 and 0.6. Figures 4-6a,7a,8a and 9a show energy against time plots for the NGP method, and figures 4-6b,7b,8b and 9b show the C-in-C results.

All results were taken at $\lambda/\lambda_D = 15.0$.

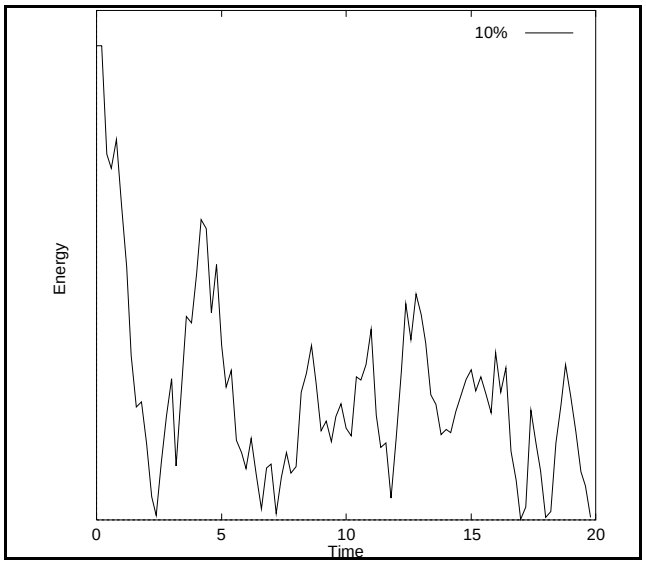

Figure 4-6a : NGP at f=0.1.

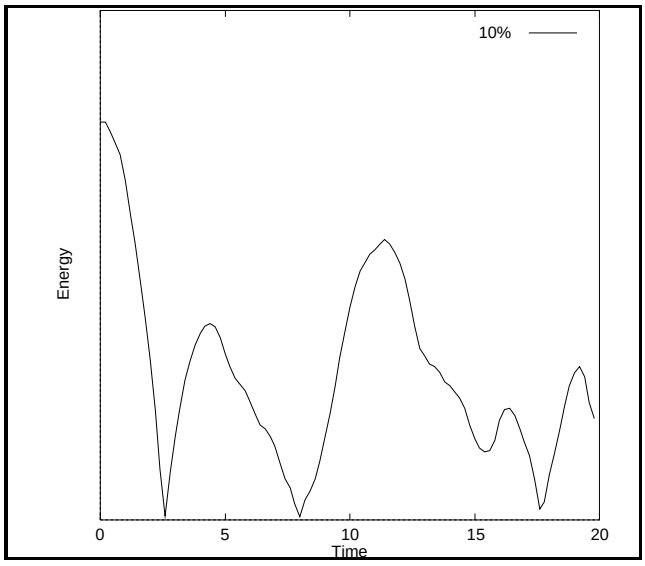

Figure 4-6b : C-in-C at f=0.1.

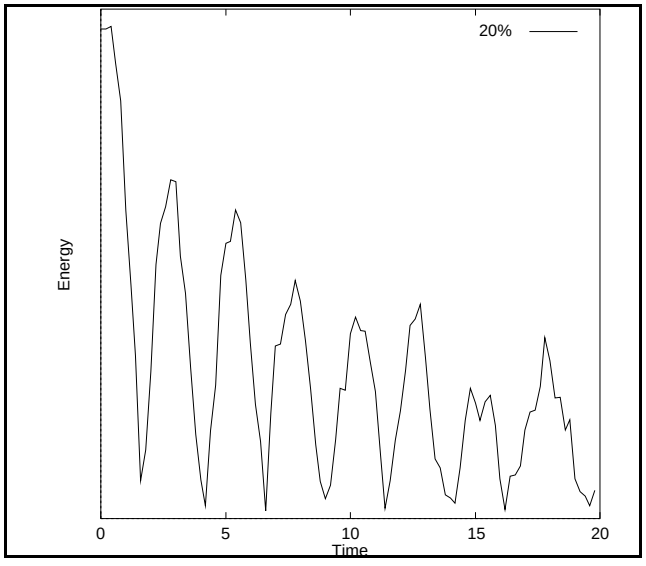

Figure 4-7a : NGP at f=0.2.

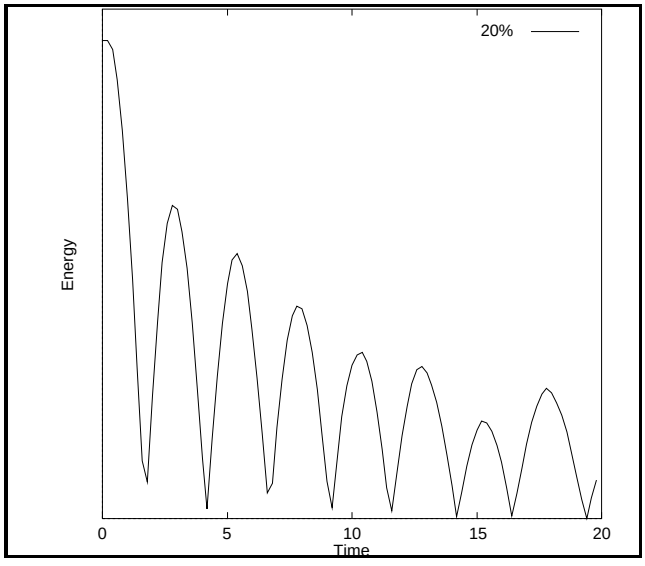

Figure 4-7b : C-in-C at f=0.2.

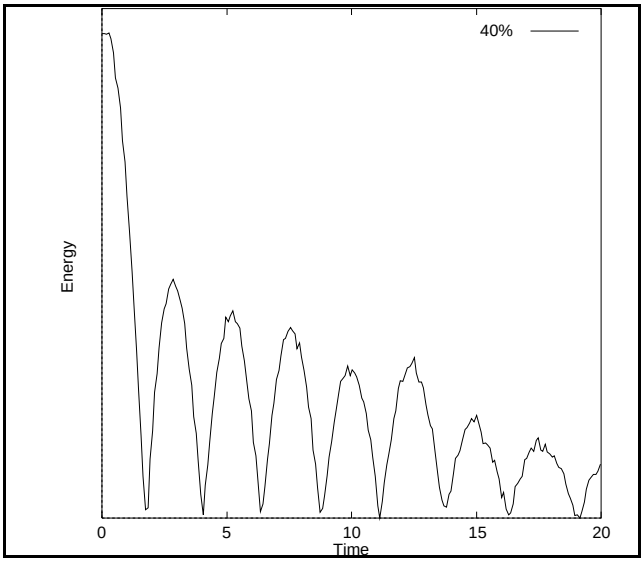

Figure 4-8a : NGP at f=0.4.

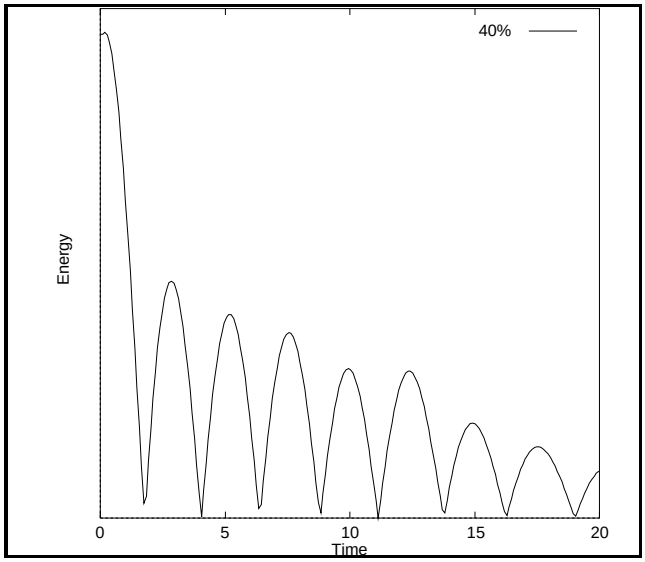

Figure 4-8b : C-in-C at f=0.4.

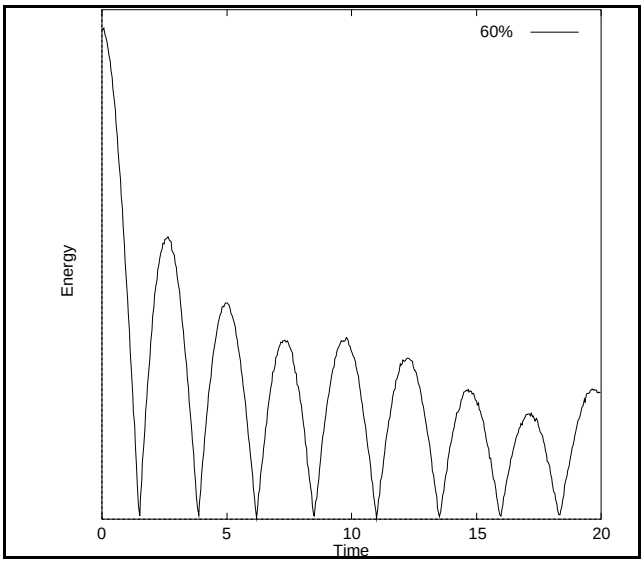

Figure 4-9a : NGP at f=0.6.

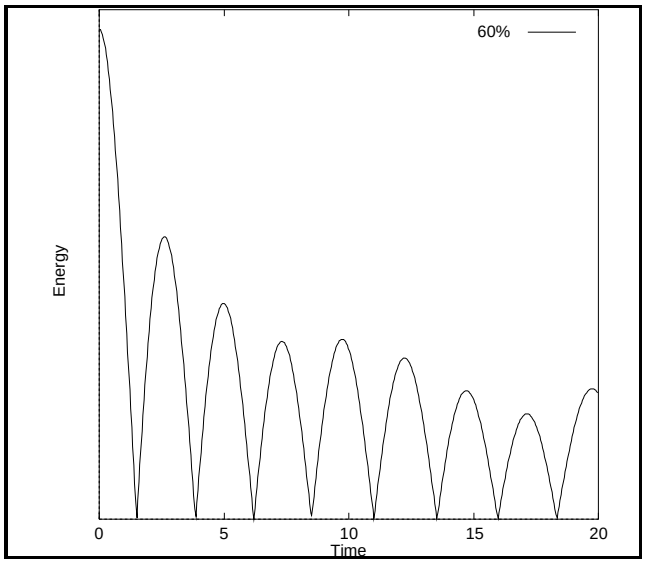

Figure 4-9b : C-in-C at f=0.6.

It can be seen from these results that the Nearest Grid Point method gives rise to a much noisier set of results than Cloud-in-Cell. It is also clear than we only need 60% of the 100,000 particles available in order to get a good set of results.

While Cloud-in-Cell is clearly more accurate, there is no point in using a method which is twice as accurate if it is four times slower to execute. Table 4-1 shows the timings taken for the two methods for a range of values of f.

f	NGP time	C-in-C time	% slower
0.1	11.2	9.2	21.7
0.2	16.3	12.4	31.5
0.3	26.6	19.3	37.8
0.4	59.8	42.6	40.4
0.5	112.6	79.2	42.2

Table 4-1

While I cannot say exactly how much more accurate the Cloud-in-Cell method is from my results, I can say that I believe it to be sufficiently better to warrant the extra ~45% of run-time it requires.

3 • Comparison of experimental results to theoretical predictions:

With the Cloud-in-Cell code set up to use 60% of the 100,00 particles, it is possible to investigate the behaviour of the system over a range of λ/λ_D and compare the results with theory. As the described earlier, the Landau condition for damping is that λ/λ_D is below about 31.4, so for this analysis the range from 5.0 to 40.0 was investigated in steps of 5.0.

Figure 4-10 shows the variation of the period of oscillation with λ/λ_D and compares it to the result predicted by theory. This shows a clear agreement of theory with experiment in the case, with little or no deviation overall.

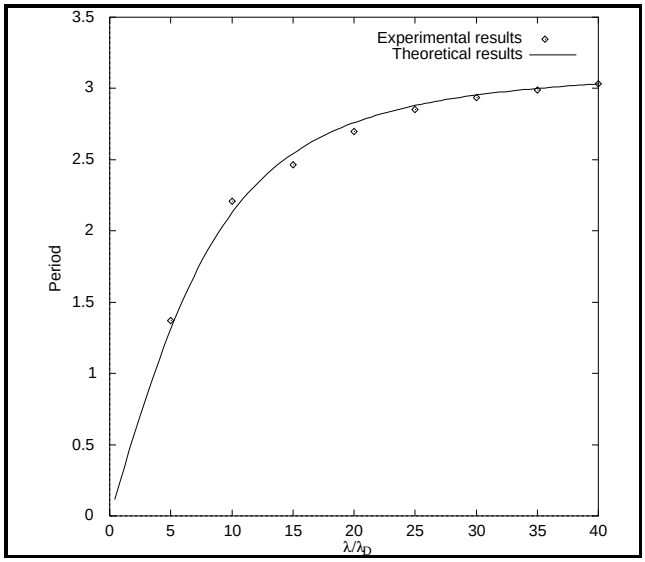

Figure 4-10

Figure 4-11 shows the results for the variation of the damping factor of oscillations (γ) with λ/λ_D and gives a comparison to the theoretical prediction. Here it can be seen that while the general form of the results fits well with those of the theory, the curves they trace do not superimpose due to the experimental results forming a somewhat taller version of the theoretical predictions. The peak of the curve is also shifted to the low λ/λ_D end. This means that the experiment generally overestimates the damping effect. The simulation makes a number of assumptions (such as the stationary positive ion background), and I cannot identify a particular reason for this disagreement with theory. However, but I do believe the disagreement is due to a rash assumption in the theory rather than a mistake in the computational implementation.

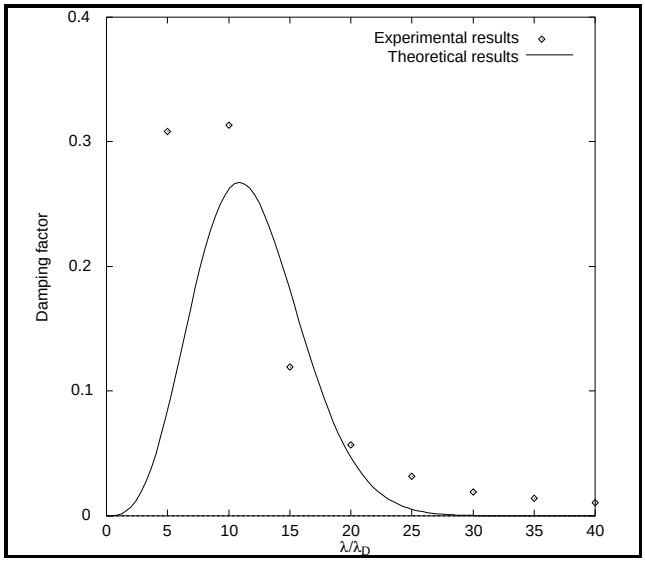

Figure 4-11

4 • Effect of wave amplitude:

Throughout the experiment, the amplitude of the initial sinusiodal charge distribution has been set to a value that was known to work, ie at 5% of the average number of particle in a cell (so that the distribution maxima contain 105% of the average no. of particles in a cell and the minima 95%). In this last section the effects of a range of amplitudes are investigated.

Figure 4-12 shows the velocity frequency plots of the electron super-particles in the plasma after one half period of oscillations (ie at the first peak in the energy versus time plot) for amplitudes of 5%, 25%, 50%, 75% and 90% of the average no. of particles per cell. The thinnest, tallest graph is at 5% where the distribution is Gaussian, ie the initial thermal distribution has not been altered greatly by the wave motion in the system. As the amplitude is increased this curve flattens out, as a large no. of the particles are now being controlled more by the wave energy than the inital conditions.

While this effect is interesting, we can learn more by investigation the chronological development of the wave at a given high amplitude. The next set of figures (4-13 to 4-21) show the varation of the frequency plot with time for a system with 50% initial wave amplitude.

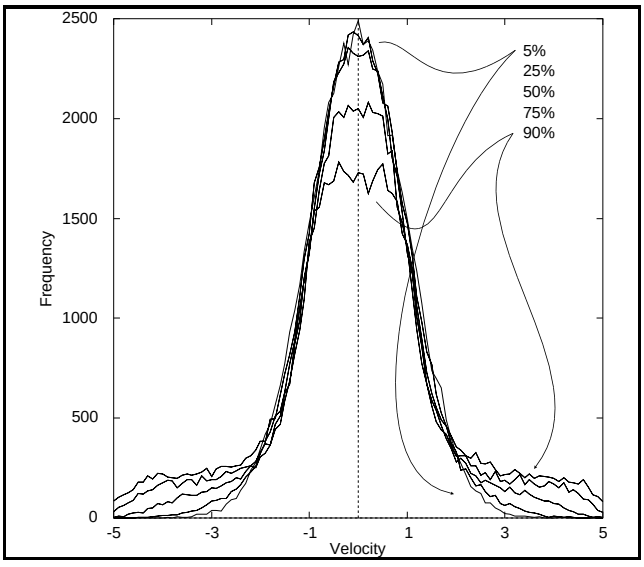

Figure 4-12

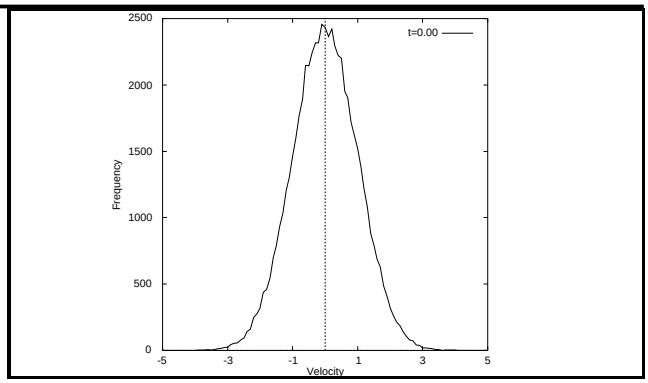

Figure 4-13

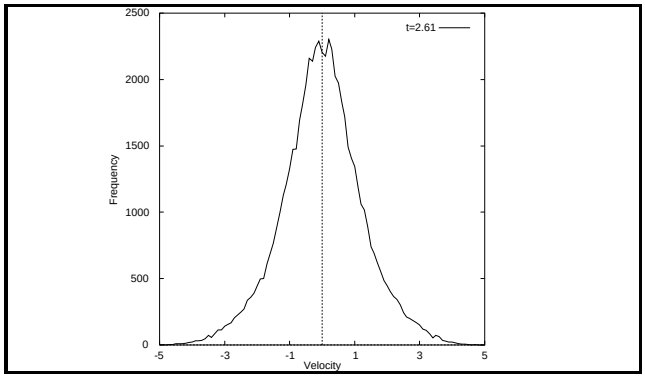

Figure 4-14

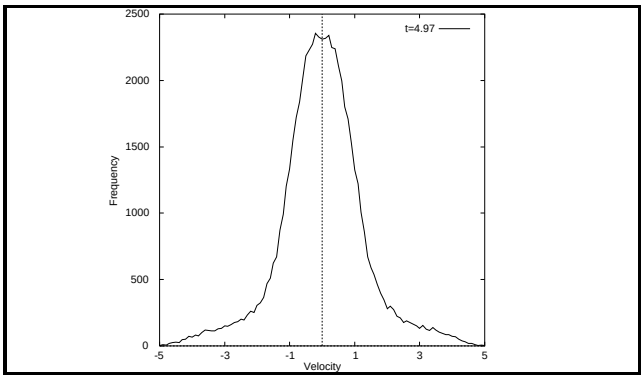

Figure 4-15

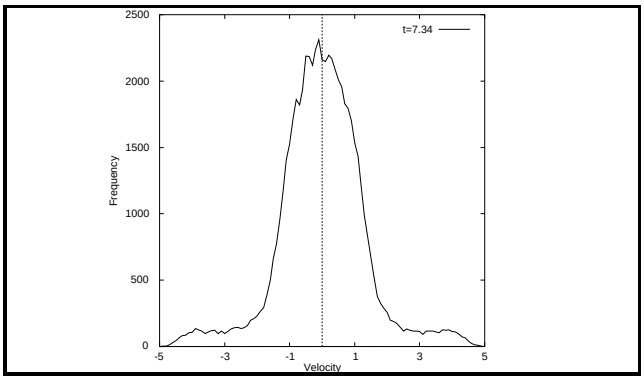

Figure 4-16

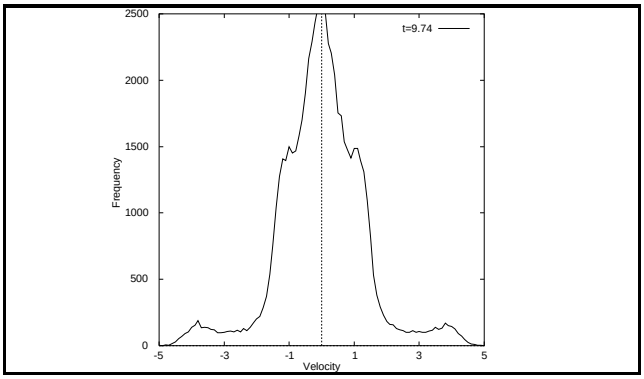

Figure 4-17

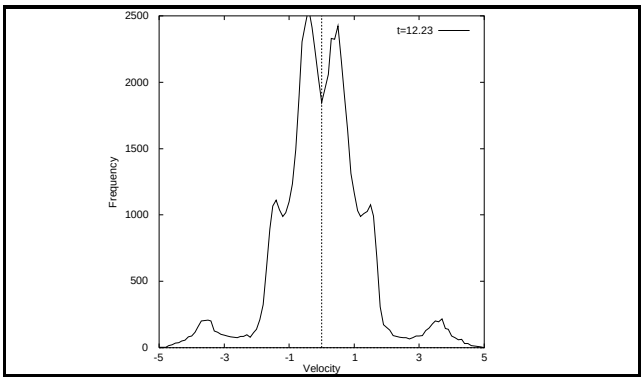

Figure 4-18

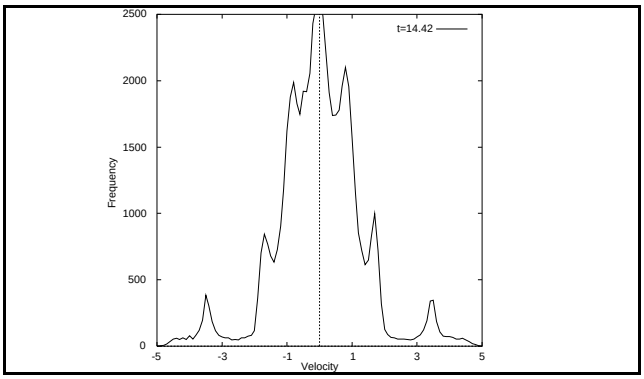

Figure 4-19

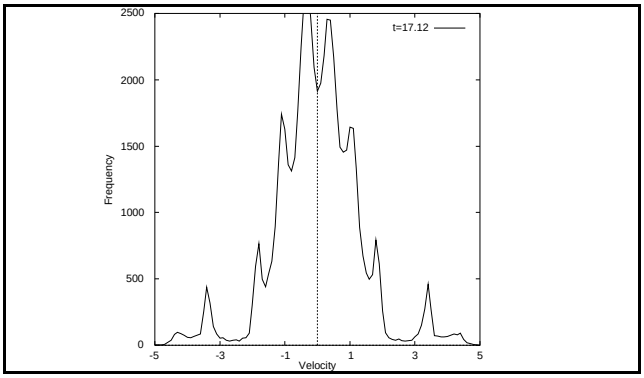

Figure 4-20

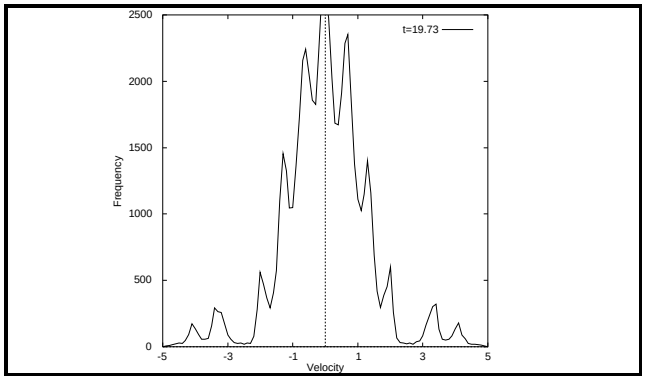

Figure 4-21

What I believe to be happening is this: The standing wave of charge in the plasma can be thought of as two waves moving in opposite directions which are colliding. In the case of string waves, large numbers of the particle are forced to move with the wave in either the positive or negative x direction. This means particle are drained from the surrounding velocities and forced into the wave velocity. This causes the particles to act like a set of oscillators, sending spikes of 'popular' velocities through the velocity frequency plot. These oscillations overlap to casue the frequency plot to appear as shown above (esp. figure 4-21).

Conclusions

While the model studied here performs well overall, it's lack of agreement with the predicted results for the Landau damping effect show that there is some room for improvement. In my opinion the theory behind the simulation could be improved by perhaps taking more account of the movement of the heavy +ve ions, and the actual simulation may benefit from using a better algorithm than the leap-frog method, which is a rather poor in comparison to the others available. This might allow more particles and cells to be used more accuratle without demanding so much more processor time (ie routines able to cope with a larger time-step).

Appendix - part 1 - NGP core code:

```
c
c Main calculation procedure:
c
      subroutine calcchange
      DIMENSION
E(0:2062),N(0:2062),V(206200),X(206200),cell(206200)
      DIMENSION EvT(-1:204800)
      COMMON /PICINT/ cell,it,NT,step,full,file,nps
      COMMON /PICREL/
E,N,V,X,DX,DT,SPperI,T,Ampl,TotTime
      COMMON /PICNRG/ Etot,EvT,Eres,Emax,Emin
      REAL
E,N,V,X,DX,DT,T,SPperI,Ehalf,Eres,Etot,EvT,Num
      REAL DXoSPperI,len
      INTEGER it,i,j,NT
      len=DX*it
      DXoSPPerI=DX/SPperI
      Ehalf=0.0
      Eres=0.0
      DO i=1,it
       Num=N(i)
       E(i)=Ehalf
       Ehalf=Ehalf-Num
       E(i)=0.5*(E(i)+Ehalf)
       Eres=Eres+E(i)
       N(i)=-SPperI
      ENDDO
      Eres=Eres/it
      DO j=1,NT
       i=MIN0(INT(x(j)/DX)+1,IT)
       v(j)=v(j)-DT*DXoSPperI*(E(i)-Eres)
       x(j)=x(j)+DT*v(j)
       IF (x(j).GT.len) x(j)=x(j)-len
       IF (x(j).LT.0.0) x(j)=x(j)+len
       i=MIN0(INT(x(j)/DX)+1,IT)
       N(i)=N(i)+1
      ENDDO
      return
      end
```

Appendix - part 2 - C-in-C core code:

```
c
c Main calculation procedure:
c
      subroutine calcchange
      DIMENSION
E(0:2063),N(0:2063),V(206200),X(206200),cell(206200)
      DIMENSION EvT(-1:204800)
      COMMON /PICINT/ cell,it,NT,step,full,file,nps
      COMMON /PICREL/
E,N,V,X,DX,DT,SPperI,T,Ampl,TotTime
      COMMON /PICNRG/ Etot,EvT,Eres,Emax,Emin
      REAL
E,N,V,X,DX,DT,T,SPperI,Ehalf,Eres,Etot,EvT,Num
      REAL DXoSPperI,len,Ex,fracA,fracB
      INTEGER it,i,j,NT
      len=DX*it
      DXoSPPerI=DX/SPperI
      Ehalf=0.0
      Eres=0.0
      DO i=1,it
       Num=N(i)
       E(i)=Ehalf
       Ehalf=Ehalf-Num
       E(i)=0.5*(E(i)+Ehalf)
       Eres=Eres+E(i)
       N(i)=-SPperI
      ENDDO
      N(0)=0.0
      N(it+1)=0.0
      Eres=Eres/it
c Add cloud-in-cell boundary continuity.
      E(0)=E(it)
      E(it+1)=E(1)
      DO j=1,NT
       i=INT((x(j)/DX)+0.5)
       fracA=i-(x(j)/DX)+0.5
       fracB=1.0-fracA
       Ex=E(i)*fracA+E(i+1)*fracB
       v(j)=v(j)-DT*DXoSPperI*(Ex-Eres)
       x(j)=x(j)+DT*v(j)
       IF (x(j).GT.len) x(j)=x(j)-len
       IF (x(j).LT.0.0) x(j)=x(j)+len
       i=INT((x(j)/DX)+0.5)
       fracA=i-(x(j)/DX)+0.5
       fracB=1.0-fracA
       N(i)=N(i)+fracA
       N(i+1)=N(i+1)+fracB
      ENDDO
c Correct for off-boundary bins (cic).
      N(it)=N(it)+N(0)
      N(1)=N(1)+N(it+1)
      return
      end
```

Appendix - part 3 - General code:

```
c
c Mode energy calculation:
c
      subroutine Energycalc
      DIMENSION
E(0:2063),N(0:2063),V(206200),X(206200),cell(206200)
      DIMENSION
EvT(-1:204800),MaxEvT(0:2048),MinEvT(0:2048)
      COMMON /PICINT/ cell,it,NT,step,full,file,nps
      COMMON /PICREL/
E,N,V,X,DX,DT,SPperI,T,Ampl,TotTime
      COMMON /PICNRG/ Etot,EvT,Eres,Emax,Emin
      COMMON /PICMAX/
MaxEvT,MinEvT,MaxPlot,NoMax,NoMin,Mflag
      REAL E,N,V,X,DX,DT,T,SPperI,EvT,Etot,Eres,a,b
      INTEGER cell,it,step,i,nps,file,E1,E2
      INTEGER MaxEvT,MinEvT,MaxPlot,NoMax,NoMin,Mflag
c Loop over electric field to find energy of fourier
mode:
      a=0.0
      b=0.0
      DO i=1,it
       a=a+E(i)*COS(6.283185307*REAL(i*nps)/REAL(it))
       a=a+E(i)*SIN(6.283185307*REAL(i*nps)/REAL(it))
      ENDDO
      a=a/(6.283185307*REAL(it))
      b=b/(6.283185307*REAL(it))
      EvT(INT(T))=SQRT(a*a+b*b)
      E1=EvT(INT(T)-1)
      E2=EvT(INT(T))
c Check for max:
      IF (Mflag.EQ.-1 .AND. E1.GT.E2) THEN
       IF (((INT(T)-1)-MaxEvT(NoMax))*DT.GT.2.0) THEN
        NoMax=NoMax+1
        MaxEvT(NoMax)=INT(T)-1
        Mflag=1
       ENDIF
      ENDIF
c Check for min:
      IF (Mflag.EQ.1 .AND. E1.LT.E2) THEN
       IF (((INT(T)-1)-MinEvT(NoMin))*DT.GT.2.0) THEN
        NoMin=NoMin+1
        MinEvT(NoMin)=INT(T)-1
        Mflag=-1
       ENDIF
      ENDIF
      return
      end
c
c Initialisation of all parameters
c
      subroutine initialise
      DIMENSION
E(0:2063),N(0:2063),V(206200),X(206200),cell(206200)
      DIMENSION
EvT(-1:204800),MaxEvT(0:2048),MinEvT(0:2048)
      COMMON /PICINT/ cell,it,NT,step,full,file,nps
      COMMON /PICREL/
E,N,V,X,DX,DT,SPperI,T,Ampl,TotTime
      COMMON /PICNRG/ Etot,EvT,Eres,Emax,Emin
      COMMON /PICMAX/
```

```
MaxEvT,MinEvT,MaxPlot,NoMax,NoMin,Mflag
      REAL
E,N,V,X,DX,DT,T,SPperI,xcen,SPforI,Eres,Ehalf,EvT
      REAL TotTime,Tlen,Tstep,Percent
      INTEGER
cell,it,nps,i,j,Ampl,NT,idum,step,full,file
      INTEGER MaxEvT,MinEvT,MaxPlot,NoMax,NoMin,Mflag
      INTEGER*2 k
      CALL TEXT_MODE@
c Ask user for specific configuration:
      WRITE(*,*) 'One Dimensional Plasma Wave
Simulation:'
      WRITE(*,*)
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'
      WRITE(*,*) ' '
      WRITE(*,*) '[Cloud-in-cell implementation.]'
      WRITE(*,*) ' '
      WRITE(*,*) ' '
      WRITE(*,*) 'Enter time between plots:'
      READ(*,*) Tstep
      WRITE(*,*) ' '
      WRITE(*,*) 'Enter total time of simulation:'
      READ(*,*) TotTime
      WRITE(*,*) ' '
      WRITE(*,*) 'Enter total length of system:'
      READ(*,*) Tlen
      WRITE(*,*) ' '
      WRITE(*,*) 'Enter fraction of particles to use:'
      READ(*,*) Percent
      WRITE(*,*) ' '
      WRITE(*,*) 'File? (0 = no output)'
      WRITE(*,*) '      (1 = phase space/frequency
plot)'
      WRITE(*,*) '      (2 = electric field plot)'
      WRITE(*,*) '      (3 = energy v time plot)'
      READ(*,*) file
      WRITE(*,*) ' '
 666  WRITE(*,*) 'Phase space plot? (y/n)'
      full=-1
      CALL GET_KEY@(k)
      IF (k.EQ.ICHAR('y') .OR. k.EQ.ICHAR('Y')) full=1
      IF (k.EQ.ICHAR('n') .OR. k.EQ.ICHAR('N')) full=0
      IF (full.EQ.-1) goto 666
      WRITE(*,*) ' '
 667  WRITE(*,*) 'Plot spline throughout? (y/n)'
      MaxPlot=-1
      CALL GET_KEY@(k)
      IF (k.EQ.ICHAR('y') .OR. k.EQ.ICHAR('Y'))
MaxPlot=1
      IF (k.EQ.ICHAR('n') .OR. k.EQ.ICHAR('N'))
MaxPlot=0
      IF (MaxPlot.EQ.-1) goto 667
c
      WRITE(*,*) ' '
      WRITE(*,*) 'Initialising; please wait...'
      idum=-1
      nps=4
      NT=(102400)*Percent
      it=(NT/100.0)*Percent
      SPperI=NT/it
      Ampl=SPperI*0.05
      DX=Tlen*nps/it
      DT=DX/4.0
      IF (DT.GT.0.2) DT=0.2
      step=Tstep/DT
      NT=0
      T=0
      NoMin=0
      NoMax=0
      Mflag=-1
      EvT(-1)=0.0
c Define sinusoidal electron arrangement of nps periods,
for N:
      DO i=1,it
       xcen=(i-0.5)*DX
       SPforI=Ampl*sin(6.283185307*nps*xcen/(it*DX))
       N(i)=-SPperI
       DO j=1,NINT(SPperI+SPforI)
        NT=NT+1
        x(NT)=(i-RAN1(idum))*DX
        v(NT)=GASDEV(idum)
       ENDDO
      ENDDO
      N(0)=0.0
      N(it+1)=0.0
c Calculate value of cic charge array (N)
      DO j=1,NT
       i=INT((x(j)/DX)+0.5)
       fracA=i-(x(j)/DX)+0.5
       fracB=1.0-fracA
       N(i)=N(i)+fracA
       N(i+1)=N(i+1)+fracB
      ENDDO
      N(it)=N(it)+N(0)
      N(1)=N(1)+N(it+1)
c Calculate initial E field:
      Ehalf=0.0
      Eres=0.0
      DO i=1,it
       Num=N(i)
       E(i)=Ehalf
       Ehalf=Ehalf-Num
       E(i)=0.5*(E(i)+Ehalf)
       Eres=Eres+E(i)
      ENDDO
      Eres=Eres/it
      E(0)=E(it)
      E(it+1)=E(1)
c Scan for electric field range:
      Emin=E(1)
      Emax=Emin
      DO i=1,it
       IF (E(i).LT.Emin) Emin=E(i)
       IF (E(i).GT.Emax) Emax=E(i)
      ENDDO
      Emin=Emin-Eres
      Emax=Emax-Eres
      return
      end
```