8-WEEK REPORT
ON

BASICS OF PLASMA SIMULATIONS USING CLOUD IN A CELL METHOD

BY

B HAREESH GAUTHAM
PHYS1449

Prepared in partial fulfillment of
INDIAN ACADEMY OF SCIENCES'
SUMMER RESEARCH FELLOWSHIP

AT

INDIAN INSTITUTE OF SCIENCE, BANGALORE

Under the guidance of

Dr. Prateek Sharma

July 21, 2014

ABSTRACT

In Plasma simulations, various physical phenomena of plasma are reproduced using various computational techniques. In these simulations, the phase space of the system is first initialized with some distribution in the positions and velocities. The space is then divided into equally spaced grids. Current and charge densities are calculated at these grid points using some weighting scheme. Then, Maxwell's equations (discretized) are solved(at the grid points) for electric and magnetic fields using these densities and currents . These are then interpolated back to the particle positions where they are used to calculate the force (on particles) and hence increment their position and velocities. This process is repeated for the desired number of time steps. Various diagnostics are derived from these calculations and hence evaluated.

Code to simulate plasma for one dimensional electrostatic and two dimensional electromagnetic conditions is written during the course of this project. For Particle Integrator, various methods of integration were implemented and their corresponding efficiency tested. Von Neumann stability analysis of various integrating schemes was done and the constraints on their stability derived. Various Weighing schemes were studied including momentum and energy conserving ones. Their relevance and limitations were also taken into account. NGP (Nearest grid point) and CIC (cloud in a cell) weighings were implemented and corresponding results compared. As a part of this, a random number generator was implemented using base-n bit reversed fractions. It gave a more uniformly distributed particle positions (or velocities) than Matlab's inbuilt random number generator.

The code was run to simulate cold plasma ,two beam instability and landau damping. The equations of motion , equation of continuity and Maxwell's equation were solved analytically to get the theoretical value of plasma frequency, landau damping rate and the two beam dispersion relation.The results were compared to the theory and the results of other codes. They were in excellent agreement. The temporal and spatial variations of various quantities including position, velocity , density, electric field etc. were also studied.

Contents

Chapter 1

The Code

This chapter intends to explain the code written to simulate the plasma. There are three major components of the code: Particle pusher which increments the particle's position and velocity , Field solver which solves the field at the grid points and the charge and field interpolator which gets us charge and current densities from particle positions and velocities using some weighing scheme.

1.1 Particle Pusher

1.1.1 Non Relativistic path integrator

The trajectory of a particle in Electric($\vec{E}$) and Magnetic field($\vec{B}$) is given by lorentz equation. In non relativistic conditions, proper and ordinary velocities are identical, hence the trajectory is given by,

$$\frac{d\vec{v}}{dt} = \frac{q}{m}(\vec{E} + \vec{v} \times \vec{B}) \tag{1.1}$$

$$\frac{d\vec{x}}{dt} = \vec{v} \tag{1.2}$$

where $\vec{v}$ is the velocity measured in lab's frame of reference.There are many methods to numerically integrate the above equations. Two method; Leap Frog method and Euler's method along with their stability is discussed below.

Euler Method

The Euler algorithm is given by:

$$\frac{m(u_{i+1} - u_i)}{\Delta t} = \; = q(\vec{E}_i + \vec{v}_i \times \vec{B}_i) \tag{1.3}$$

$$\frac{\vec{x}_{i+1} - \vec{x}_i}{\Delta t} = \vec{v}_i \tag{1.4}$$

Using this algorithm we can integrate to find velocity and position of particles simultaneously after desired number of time steps.

Stability Analysis Let the growth rate be g given by $g = v^{i+1}/v^i$. Putting it in the above equation and resolving the components, we get:

$$v_x^i(g-1) - \omega_c \Delta t v_y^i = 0 \tag{1.5}$$

$$v_y^i(g-1) + \omega_c \Delta t v_x^i = 0 \tag{1.6}$$

Writing the above equations in matrix form and solving them for non trivial solutions require the determinant to be non zero. This gives us the value of g,

$$gg^* = 1 + (\omega_c \Delta t)^2 \tag{1.7}$$

which is always greater than 1, hence Euler's method is always unstable.

Leap frog method

The leap frog algorithm is given by:

$$\frac{m(\vec{v}_{i+\frac{1}{2}} - \vec{v}_{i-\frac{1}{2}})}{\Delta t} = q(\vec{E} + \frac{\vec{v}_{i+\frac{1}{2}} + \vec{v}_{i-\frac{1}{2}}}{2} \times \vec{B}) \tag{1.8}$$

$$\frac{\vec{x}_{i+1} - \vec{x}_i}{\Delta t} = \vec{v}_{i+\frac{1}{2}} \tag{1.9}$$

The index i is used to designate time .This method is time centered with velocity ahead of position by $\Delta t/2$ amount of time. Hence velocity and position are not calculated simultaneously at a given time. If the initial conditions are given at a time t, the velocity is calculated at time $t - \Delta t/2$ and hence the computation is continued.

Stability Analysis The growth rate(g) is given by,

$$\frac{v_x^{i+\frac{1}{2}}}{v_x^{i-\frac{1}{2}}} = g \tag{1.10}$$

If $||g|| > 1$, the method is stable else it is not. For Leap frog method we have (resolving the components),

$$(g-1)v_x^{i-\frac{1}{2}} = \frac{\omega_c \Delta t}{2}(g+1)v_y^{i-\frac{1}{2}} \tag{1.11}$$

$$(g-1)v_y^{i-\frac{1}{2}} = -\frac{\omega_c \Delta t}{2}(g+1)v_x^{i-\frac{1}{2}} \tag{1.12}$$

writing the above equations in matrix form and solving them for non trivial solutions, require the determinant to be non zero. This gives us the value of g,

$$gg^* = \frac{1 + (\omega_c \Delta/2)^2}{1 + (\omega_c \Delta/2)^2} = 1 \tag{1.13}$$

Hence this method is always stable. We will use this method to increment velocity and position. To increment the velocity , we use boris' scheme:

1. Evaluate $\vec{v}^{-} = \vec{v}_{i-\frac{1}{2}} + \frac{q\vec{E}\Delta t}{2m}$
2. Evaluate $\vec{v}^{'} = \vec{v}^{-} + \vec{v}^{-} \times \vec{t}$
3. Evaluate $\vec{v}^{+} = \vec{v}^{-} + \vec{v}^{'} \times \vec{s}$
4. And finally evaluate $\vec{v}_{i+\frac{1}{2}} = \vec{v}^{+} - \frac{q\vec{E}\Delta t}{2m}$

where $\vec{t} = \frac{q\vec{B}\Delta t}{2m}$ and $\vec{s} = \frac{2\vec{t}}{t^2+1}$. It has the advantage of reducing the number of multiplications required. It can also be easily generalized to relativistic cases. Details of how it works is given in Reference 1. Following is the code in MATLAB for leapfrog method using boris' scheme.

```
function [x]=incPos(x,v,dt)
%This function increments the position array(x)
%using the velocity array(v)
%dt is the time step
x(:,i)=rem(rem(x(:,i)+(v*dt),l)+l,l);

function [v]=incVel(v,q,e,b,dt,m)
%This function increments the velocity(v) (by boris's scheme )using:
%q, charge of the speices
%e, electric field interpolated to the individual particle positions
% b, magetic  field interpolated to the individual particle positions
% dt is the time step and
% m is the mass of the speices
for i=1:1:length(v)
    v1=v(:,i)';
    e1=e(:,i)';
    vm=v1+(q*e1*dt/(2*m));
    t=(q*b*dt)/(2*m);
    vdsh=vm+cross(vm,t);
    s= 2*t/(dot(t,t)+1);
    vp=vm+cross(vdsh,s);
    v1=vp+(q*e1*dt/(2*m));
    v(:,i)=v1;
end
```

Please note that incPos function is written to ensure periodicity of the system. Particles which cross at x=L enter the system at x=0 with same velocity.

1.1.2 Relativistic path integrator

The relativistic equations of motion is given by,

$$\frac{(d\gamma\vec{v})}{dt} = \frac{q}{m}(\vec{E} + (\vec{v} \times \vec{B})) \quad (1.14)$$

$$\frac{d\vec{x}}{dt} = \vec{v} \quad (1.15)$$

where $\vec{v}$ is ordinary velocity, the one we measure in laboratory frame ,m is the rest mass, $\gamma = \frac{1}{\sqrt{1-\frac{v^2}{c^2}}} = \sqrt{1+\frac{u^2}{c^2}}$ and other symbols have their usual meaning. The γ factor is attributed to the usage of proper time while calculating momentum. We define the proper velocity, $\vec{u} = \gamma\vec{v}$. Using this in the above equation and discretizing it (using leapfrog method), we get:

$$\frac{\vec{u}_{i+1/2} - \vec{u}_{i-1/2}}{\Delta t} = \frac{q}{m}\vec{E}_i + \frac{q}{m\gamma_i}\frac{\vec{u}_{i+1/2} + \vec{u}_{i-1/2}}{2} \times \vec{B} \tag{1.16}$$

Hence we can follow the relativistic boris' scheme in following steps:

1. Evaluate $\vec{u}_{i-1/2} = \gamma_{i-1/2}\vec{v}_{i-1/2}$.
2. Evaluate $\vec{u}^- = \vec{u}_{i-\frac{1}{2}} + \frac{q\vec{E}\Delta t}{2m}$
3. Evaluate $\vec{u}^{'} = \vec{u}^- + \vec{u}^- \times \vec{t}$ where $\vec{t} = \frac{q\vec{B}\Delta t}{2m\gamma_\pm}$
4. Evaluate $\vec{u}_+ = \vec{u}^- + \vec{u}^{'} \times \vec{s}$ where $\vec{s} = \frac{2\vec{t}}{t^2+1}$
5. Evaluate $\vec{u}_{i+\frac{1}{2}} = \vec{u}^+ - \frac{q\vec{E}\Delta t}{2m}$
6. And finally evaluate $\vec{v}_{i+1/2} = \vec{u}_{i+\frac{1}{2}}/\gamma_{i+1/2}$

Here the subscript on γ indicates the velocity to be used in its evaluation. Note that we get $\vec{u}_+$ by rotating $\vec{u}_-$, hence their magnitude is same and their γ. Following is the MATLAB implementation of the above scheme.

```
function [v]=incVel(v,q,e,b,dt,m,c)
%function for evaluating the velocity, v a (3 * N ) array to store
%three components of velocity of each particle using:
%charge q,
% electric field e,
% magnetic field b,
%time step dt,
% mass of each particle m and
% speed of light c
for i=1:1:length(v)
    v1=v(:,i)';
    u1=v1./sqrt(1-(norm(v1)/c)^2);
    e1=e(:,i)';
    b1=b(:,i)';
    um=u1+(q*e1*dt/(2*m));
    gamn=sqrt(1+(norm(um)/c)^2);
    t=(q*b1*dt)/(2*gamn*m*c);
    udsh=um+cross(um,t);
    s= 2*t/(dot(t,t)+1);
    up=um+cross(udsh,s);
    u1=up+(q*e1*dt/(2*m));
    v1=u1./sqrt(1+(norm(u1)/c)^2);
    v(:,i)=v1;
end

function [x]=incPos(x,v,dt,l)
%this function increments the position.
for i=1:1:length(x)
x(:,i)=rem(rem(x(:,i)+(v*dt),l)+l,l);
end
```

1.2 Field Solver

The following section deals with solving the Maxwell's equations for fields. We will start with 1D electrostatic case.

1.2.1 One Dimensional Electrostatic Fields

Following are the equations used for Electrostatic conditions,

$$\nabla^2\phi = -\frac{\rho}{\epsilon_0} \quad (1.17)$$

$$\vec{E} = -\nabla\phi \quad (1.18)$$

$$(1.19)$$

First is Gauss's equation whereas the second one follows from the definition of potential.For electrostatic conditions, current density is assumed to be constant. Hence the other maxwell equations assume constant or trivial solutions. If the space is divided into uniformly spaced grids, solving these equations at the grid points require them to be discretized :

$$\frac{\phi_{i+1} - 2\phi_i + \phi_i}{\Delta x^2} = -\frac{\rho_i}{\epsilon_0} \quad (1.20)$$

$$E_{x_i} = -\frac{\phi_{i+1} - \phi_i}{\Delta x} \quad (1.21)$$

$$(1.22)$$

Solving these equations for Electric field we require the charge density at the grid points. For charges distributed in space, charge density is found at grids using some weighing scheme(discussed in next section). We can solve these equations in fourier space, advantage being a fast algorithm (FFT) available to solve for fourier transform. We define the fourier transform and inverse transform ,to be:

$$\begin{aligned} G(n) &= \sum_{j=1}^{N-1} G(j)e^{-ij\frac{2\pi}{P}n} && (1.23) \\ G(j) &= \frac{1}{N}\sum_{n=1}^{N-1} G(n)e^{ij\frac{2\pi}{P}n} && (1.24) \end{aligned}$$

where P is the period, N being the number of grids ($N\Delta x = P$) , j is used to index the space variable and n to index fourier variable. This in the context of the above equations, can be defined as:

$$\begin{aligned} G(k) &= \Delta x \sum_{j=1}^{NG-1} G(j)e^{-ikX_j} && (1.25) \\ G(j) &= \frac{1}{L}\sum_{n=1}^{NG-1} G(n)e^{ikX_j} && (1.26) \end{aligned}$$

Here L is the period, the fourier variable k=$\frac{2\pi n}{L}$,Δx being the grid width and $X_j(= j\Delta x)$ is the space variable. If we take the continous fourier transform of the field equations, we get:

$$\begin{aligned} E_x(k) &= -ik\phi(k) && (1.27) \\ \frac{\rho(k)}{\epsilon_0} &= k^2\phi(k) && (1.28) \end{aligned}$$

Since we are discretizing the equations and taking discrete Fourier transforms, we get(from the above definitions),

$$\begin{aligned} E_{x_j} &= -\frac{\phi_{j+1} - \phi_{j-1}}{2\Delta x} \\ \frac{1}{L}\sum_{n=1}^{NG-1} E(n)e^{ikX_j} &= -\frac{1}{L}\frac{\sum_{n=1}^{NG-1}\phi(n)e^{ikX_{j+1}} - \sum_{n=1}^{NG-1}\phi(n)e^{ikX_{j-1}}}{2\Delta x} \\ &= -\frac{1}{L}\frac{\sum_{n=1}^{NG-1}\phi(k)e^{ikX_j}(e^{ik\Delta x} - e^{-ik\Delta x})}{2\Delta x} \\ &= -\frac{1}{L}\frac{\sum_{n=1}^{NG-1}\phi(k)e^{ikX_j}(e^{ik\Delta x} - e^{-ik\Delta x})}{2\Delta x} \\ &= -\frac{i}{L}\sum_{n=1}^{NG-1} k\frac{sin(k\Delta x)}{k\Delta x}\phi(k)e^{ikX_j} \end{aligned}$$

Comparing the left and right hand side of the above equation,

$$E(k) = -ik\frac{sin(k\Delta x)}{k\Delta x}\phi(k) \implies E(k) = -i\kappa\phi(k) \qquad (1.29)$$

Doing a similar analysis for the other equation, we get:

$$\phi(k) = \frac{\rho(k)}{K^2\epsilon_0}, \quad K^2 = k^2(\frac{sin\frac{k\Delta x}{2}}{\frac{k\Delta x}{2}})^2 \tag{1.30}$$

Hence the scheme for solving the equations is,

1. Divide the space into equally spaced grids.
2. Find the charge density by using some weighing scheme.
3. Find $\phi(k)$ by taking the fourier transform of charge density.
4. Find $E(k)$ from $\phi(k)$.
5. Find $E(x_g)$ by taking inverse fourier transform of the E(k).

Following is the MATLAB implementation of the field solver:

```
function [e]=Efield(rho,L,dx)
% rho is an array(1 * N) of density
% L is the length
% dx is the grid spacing
% it returns an 1 * N array electric field, e
rhok=fft(rho);
N=length(rhok);
k=2*pi*(0:1:N-1)/L;
Ksq=(k.*(sin(k*dx/2)./(k*dx/2))).^2;
phik=rhok./Ksq;
kappa=(k.*sin(k*dx))./(k*dx);
ek=(-1i*kappa).*phik;
e=ifft(ek);
e=real(e);
```

Here MATLAB's inbuilt functions, fft() and ifft() which returns fourier and inverse fourier transforms respectively have been used. An implementation of fft is given below,

```
function [f]=fftlocal(x)
% returns discrete fourier transform (f) of the array x using
%fast fourier transform algorithm for radix 2 cases
if(length(x)==1)
      f=x(1);
else
    n=length(x);
    e=x(1:2:n);
    o=x(2:2:n);
    fe=fftlocal(e);
    fo=fftlocal(o);
    f=zeros(n,1);
    for k=1:1:n/2
    f(k)=fe(k)+(fo(k)*(exp(-1j*2*pi*(k-1)/n)));
    f(k+(length(x)/2))=fe(k)-(fo(k)*(exp(-1j*2*pi*(k-1)/n)));
    end
end
```

but unlike MATLAB's inbuilt function this works only if array x has 2^n number of elements and is less accurate.

1.2.2 Electromagnetic fields

This section deals with electromagnetic fields, hence generalizing the above code. We will consider only TM fields i.e B_z, E_x and E_y. We can do that because the other three components, TE fields(E_z, B_y and B_z) are uncoupled to TM fields. We will consider the variation of fields and densities to be along x-axis only. Apart from the Poisson's equation we need to solve,

$$\frac{\partial E}{\partial t} = c^2 \nabla \times \vec{B} - \frac{1}{\epsilon_0}\vec{J} \tag{1.31}$$

$$\frac{\partial \vec{B}}{\partial t} = -\nabla \times \vec{E} \tag{1.32}$$

Resolving the equations into components under the assumed conditions, we have:

$$-\frac{\partial E_y}{\partial x} = \frac{\partial B_z}{\partial t} \tag{1.33}$$

$$-\frac{\partial B_z}{\partial x} = \mu_0 J_y + \mu_0 \epsilon_0 \frac{\partial E_z}{\partial t} \tag{1.34}$$

Working in units with $\mu_0 = \epsilon_0 = 1$(lorentz heaviside units), we get

$$(\frac{\partial}{\partial t} \pm \frac{\partial}{\partial x})F_\pm = -\frac{J_y}{2} \tag{1.35}$$

where,

$$F_\pm = \frac{1}{2}(E_y \pm B_z) \tag{1.36}$$

we can solve the above equation along the characteristic, $\Delta x = \Delta t$. Then the right hand side bracket becomes a total derivative with respect time. Hence we get:

$$F^{n+1}_{\pm j} = F^n_{\pm j+1} - \frac{\Delta t(J^-_{y,j+1} + J^+_{y,j})}{4} \tag{1.37}$$

where j is used to index the position and n to index time. $J^\pm$ is the current calculated using interpolation of velocity($\vec{v}(t + \Delta t/2)$) by positions at t and $t + \Delta t$ respectively. Following is the MATLAB implementation of the above scheme:

```
function [Ey,Bz]=incField(Ey,Bz,dt,jym,jyp)
% this function increments the fields, Bz and Ey using,
% jym,jyp the y components of currents  and
%dt time step.
Fp=(Ey+Bz)/2;
Fm=(Ey-Bz)/2;
ng=length(Ey);
for j=1:1:ng-1
    Fp(j)=Fp(j+1)-((dt/4)*(jym(j+1)+jyp(j)));
    Fm(j)=Fm(j+1)-((dt/4)*(jym(j+1)+jyp(j)));
end
Fp(ng)=Fp(1)-((dt/4)*(jym(1)+jyp(ng)));
Fm(ng)=Fm(1)-((dt/4)*(jym(1)+jyp(ng)));
Ey=Fp+Fm;
Bz=Fp-Fm;
```

Please not that E_x is found in the usual way, by solving the Poisson's equation.

1.3 Weighing

This section deals with the weighing used in the code to get charge densities at the grid as well as the forces from grid to individual particles. We will discuss two types of weighing, NG nearest grid and CIC cloud in a cell weighing. The convention used in code is that the grids are equally spaced at a distance of Δx. Hence the position of ith grid is given by, $X_i = i\Delta x, 0 \leq i \leq N-1$ where $N = L/\Delta x$. Since the system is assumed to be periodic, charge of particles at $(N-1)\Delta x \leq x \leq L$ are accumulated to the zeroth and $N-1$th grid. Forces are also interpolated back in a similar fashion.

1.3.1 Nearest grid point

In nearest grid point method, charge of a particle is assigned to the nearest grid. Hence charge accumulated at a grid is given by,

$$q_j = \sum_i q_i \mid (x_j - \frac{\Delta x}{2}) \leq x_i \leq (x_j + \frac{\Delta x}{2}) \tag{1.38}$$

Values of forces calculated at the nearest grid point is assigned to the particles.

1.3.2 Cloud in cell

In this method linear interpolation is used to assign charges to the grid points. Nearest two grid points are used to interpolate. Fluctuations are hence much smoother than in NGP method. Contribution of a charge to a grid point is given by,

$$q_j = \sum_i q_i \frac{|X_j - x_i|}{\Delta x} \mid (x_j - \Delta x) \leq x_i \leq (x_j + \Delta x) \tag{1.39}$$

Same technique is used to interpolate the fields back to the positions from the grids.

$$F_x = F_j \frac{(|x_j - x|)}{\Delta x} + F_{j+1} \frac{(|x_{j+1} - x|)}{\Delta x} \tag{1.40}$$

Following is the MATLAB implementation of this method.

```
function [ex]=interpolateVec(eg,x,dx)
% this function intepolates any vector eg known at grids
% with grid width dx,
% to the particle positions x and store the result in
% the array(1 *N) ex.
ex=zeros(1,length(x));
ng=length(eg);
for i=1:1:length(x)
    fl=floor((x(i)/dx));
    lf=eg(fl+1)*(fl+1-(x(i)/dx));
    if fl+1==ng
        gf=eg(1)*((x(i)/dx)-fl);
    else
        gf=eg(fl+2)*((x(i)/dx)-fl);
    end
    ex(i)=gf+lf;
end

function[rhog]=accumulateCharge(x,q,ng,rb,dx)
% this function accumulates charge/length at the grid points in the array
% rhog using:
% x, array of positions of particles,
% q, charge of particles(this is assumed to be same for all particles),
% ng, number of grids,
% rb, background neutralizing charge of the ions,
% dx, the grid width.
rhog=rb*ones(1,ng);
np=length(x);
for i=1:1:np
    fl=floor(x(i)/dx);
    rhog(fl+1)=rhog(fl+1)+q*(fl+1-(x(i)/dx));
    if fl+1==ng
    rhog(1)=rhog(1)+q*((x(i)/dx)-fl);
    else
    rhog(fl+2)=rhog(fl+2)+q*((x(i)/dx)-fl);
    end
end
rhog=rhog/dx;
```

1.4 Initialization

This section is about initializing the phase space of the system. The system is made neutral with a uniform ion background. Ions are assumed to be heavy and hence immobile. The positions of the (mobile) particles are initialized at half integral multiples of L/N where L is length of the system and N the number of particles. The positions and velocities can also be randomized. n-bit reversed fractions are far more uniformly distributed than the MATLAB's rand function. Following is a MATLAB implementation of the n-bit reversed fractions,

```
function [r]= randomNum(n,b)
% this function returns n-bit reverse fraction.
% b is the base
% n is the number to be reversed.
r=0;
p=0;
while (n>=1)
    p=p+1;
    if(mod(n,b)~=0)
        r=r+(mod(n,b)/b^p);
    end
    n=floor(n/b);
end
```

If the system is to be initialized with a distribution in velocity, there are two ways to do it. First one is the quiet start and other is randomized. Following are the steps to initialize the system to a distribution $f(v)$ in velocity.

1. Find the normalized distribution of f, $\hat{f}$.
2. Find the cumulative distribution of $\hat{f}$, F.
3. Find the solution of the equations,

$$F(v) = (i+1/2)/N, i = 0, 1...N-1 \quad \text{for quiet starts and} \tag{1.41}$$
$$F(v) = R, 0 \leq R \leq 1 \quad \text{for randomized} \tag{1.42}$$

   by inverting the functions.

Since one dimensional gaussian cannot be integrated analytically, it has to integrated numerically. Reverse interpolation is used to invert the function and get velocity. Following is the MATLAB implementation of the above steps for one dimensional Maxwellian quiet starts:

```
function [F]=maxwellInts(vmax,sigma,nsi)
% This function integrates(by mid point method)
%the 1D maxwellian distribution(mean is assumed to be zero) using
% vmax, the maximum velocity
% sigma, the standard deviation of the distribution
% nsi, the number of steps ofintegration.
% the cumulative integral is stored in F, (1 * nsi) array
dv=2*vmax/nsi;
v=-vmax+(dv/2);
F=zeros(nsi);
for i=2:1:nsi
    F(i)=F(i-1)+((exp(-(v^2)/(2*sigma^2)))*dv);
    v=v+dv;
end

function [v]= velocityInit(R,vmax,F)
% This function return the velocity v(by interpolating), using
% R, any number between -vmax and vmax
% vmax, maximum velocity
% F, cumulative maxwellian distribution
nsi=length(F);
dv=2*vmax/nsi;
if (F(nsi)>R)
    for i=1:1:nsi-1
        if (F(i+1)>R)
            break;
        end
    end
    v1=-vmax+(i*dv);
    v2=v1+dv;
    v=((v1*(F(i+1)-R))+(v2*(R-F(i))))/(F(i+1)-F(i));
else
    v=0;
end
```

1.5 Final code

Here is the final code for one dimensional electrostatic situations. Code can be changed to electromagnetic one by adding the functions discussed in the previous sections.

```
function finalCode
% number of particles
np=30000;
% length
l=30000;
%number of grids
ng=500;
% grid spacing
dx=l/ng;
%charge
```

```
q=-1;
%mass
m=1;
%background charge denity
rb=-q*np/l;
%initialization of phase space.
[x,v]=initializePhaseSpace(np,l);
x0=x;
rhog=accumulateCharge(x,q,ng,rb,dx);
eg=Efield(rhog,l,dx);
ex=interpolateVec(eg,x,dx);
%time step
dt=0.01;
%calculate velocity at t=-dt/2
v=velAdj(v,ex,dt,q,m);
% mode of perturbation
mode=1;
x=pertX(x,0.1,l,mode);
% time to which the code should run
tmax=10;
time=0:dt:tmax;
% implementation of leapfrog method
for t=time
    %increment velocity
    v=incVel(v,q,ex,dt,m);
    %increment position
    x=incPos(x,v,dt,l);
    %accumulate charge at grids
    rhog=accumulateCharge(x,q,ng,rb,dx);
    % calculate electric field
    eg=Efield(rhog,l,dx);
    % interpolate electric field to positions
    ex=interpolateVec(eg,x,dx);
end

function [x]=pertX(x,x1,l,n)
x=x+(x1*cos(2*pi*(n/l)*x));

function [v]=incVel(v,q,e,dt,m)
v=v+((q/m)*e*dt);

function [x]=incPos(x,v,dt,l)
x=rem(rem(x+(v*dt),l)+l,l);

function[x,v] = initializePhaseSpace(np,l)
x=(l/(2*np)):l/np:l;
```

```
v=zeros(1,np);
% v=randn(1,length(x));

function[rhog]=accumulateCharge(x,q,ng,rb,dx)
rhog=rb*ones(1,ng);
np=length(x);
for i=1:1:np
    fl=floor(x(i)/dx);
    rhog(fl+1)=rhog(fl+1)+q*(fl+1-(x(i)/dx));
    if fl+1==ng
    rhog(1)=rhog(1)+q*((x(i)/dx)-fl);
    else
    rhog(fl+2)=rhog(fl+2)+q*((x(i)/dx)-fl);
    end
end
rhog=rhog/dx;

function [e]=Efield(rho,L,dx)
rhok=fft(rho);
N=length(rhok);
k=2*pi*(0:1:N-1)/L;
Ksq=(k.*(sin(k*dx/2)./(k*dx/2))).^2;
phik=rhok./Ksq;
% phik(1)=0;
kappa=(k.*sin(k*dx))./(k*dx);
ek=(-1i*kappa).*phik;
% ek(1)=0;
e=ifft(ek);
e=real(e);

function [ex]=interpolateVec(eg,x,dx)
ex=zeros(1,length(x));
ng=length(eg);
for i=1:1:length(x)
    fl=floor((x(i)/dx));
    lf=eg(fl+1)*(fl+1-(x(i)/dx));
    if fl+1==ng
        gf=eg(1)*((x(i)/dx)-fl);
    else
        gf=eg(fl+2)*((x(i)/dx)-fl);
    end
    ex(i)=gf+lf;
end

function [v]=velAdj(v,ex,dt,q,m)
v=v-((2*q*ex)/(m*dt));
```

Chapter 2

Simulations

This chapter deals with the various simulations done using the above code.

2.1 Cold Plasma Simulation

This simulation is intended to study the response of a 1D plasma to small perturbations. Plasma particles oscillate when subjected to small perturbations about their positions with no waves not propagating. This is true only for one dimensional case with uniform neutralizing positive ion background. The frequency of the oscillation is characteristic of a given plasma. Following is a derivation of theoretical value of plasma frequency.

2.1.1 Derivation of the plasma frequency

The equations to be solved are continuity equation, gauss's equation and lorentz force equation.

$$\nabla.\vec{E} = \frac{\rho}{\epsilon} \tag{2.1}$$

$$\frac{d\vec{v}}{dt} = \frac{q}{m}(\vec{E} + \vec{v} \times \vec{B}) \tag{2.2}$$

$$-\nabla.(n_e\vec{v}) = \frac{\partial(n_e)}{\partial t} \tag{2.3}$$

Assuming linear perturbations, with $\vec{E} \rightarrow \vec{E}_0 + \vec{E}_1$, $n \rightarrow n_0 + n_1$ and $\vec{v} \rightarrow \vec{v}_0 + \vec{v}_1$, $n_0, \vec{v}_0$ and $\vec{E}_0$ being the equilibrium values.Substituting these in the above equations and neglecting the higher order terms, we get,

$$\nabla.\vec{E}_1 = \frac{\rho_1}{\epsilon} \tag{2.4}$$

$$\frac{\partial\vec{v}_1}{\partial t} + \vec{v}_0.\nabla\vec{v}_1 = \frac{q}{m}(\vec{E}_1 + \vec{v}_1 \times \vec{B}) \tag{2.5}$$

$$-\nabla.(n_{e0}\vec{v}_1 + n_{e1}\vec{v}_0) = \frac{\partial(n_{e1})}{\partial t} \tag{2.6}$$

Assuming electric field to be along x axis, magnetic field to be constant along z axis , the motion of the particles to be confined to x-y plane with variations only along x-axis , $\vec{v}_0 = 0$ and solutions of the form $e^{i(kx-\omega t)}$, we get:

$$-i\omega v_{1x} = \frac{q}{m}(E_1 + v_{1y}B) \tag{2.7}$$

$$-i\omega v_{1y} = -\frac{q}{m}v_{1x}B \tag{2.8}$$

$$ik_x E_1 = \frac{\rho_1}{\epsilon_0} \tag{2.9}$$

$$\rho_1 = \rho_0(\frac{kv_{1x}}{\omega}) \tag{2.10}$$

Solving the above system for a non trivial system requires,

$$\omega^2 = \omega_p^2 + \omega_c^2 \tag{2.11}$$

Where $\omega_p = \sqrt{\frac{q^2 n}{\epsilon_0 m}}$ is the plasma frequency and $\omega_c = \frac{qB}{m}$ is the cyclotron frequency. If the system is evaluated for a non magnetic conditions and non zero $\vec{v}_0$, we get:

$$1 = \frac{\omega_p^2}{(\omega - \vec{k}.\vec{v}_0)^2} \tag{2.12}$$

Now including the magnetic field we get the dispersion relation,

$$\omega = \pm\sqrt{\omega_p^2 + \omega_c^2} + \vec{k}.\vec{v}_0 \tag{2.13}$$

2.2 Two beam instability

System of two beams of charges(like or unlike) moving in opposite direction is unstable. The following simulation illustrates this phenomenon as well as compares the results with theory. The dispersion relation can be found as for the one streams case.The only difference comes in the evaluation of Gausss law,

$$\nabla.\vec{E}_1 = \frac{\rho_{11}}{\epsilon_0} + \frac{\rho_{12}}{\epsilon_0} \tag{2.14}$$

where ρ_{12} and ρ_{11} are the densities of the individual streams. Following the analysis similar to that of one stream case, we get the relation:

$$1 = \frac{\omega_{p1}^2}{(\omega - \vec{k}.\vec{v}_{01})^2} + \frac{\omega_{p2}^2}{(\omega - \vec{k}.\vec{v}_{02})^2} \tag{2.15}$$

which for the case of identical plasma($\omega_{p1} = \omega_{p2}$) , identical charge and opposite velocities($v_1 = -v_2$), reduces to

$$\omega = \pm((kv_0)^2 + \omega_p^2 \pm \omega_p(4k^2v_0^2 + \omega_p^2)^{1/2})^{1/2} \tag{2.16}$$

which can be real(stable) or imaginary(unstable).The condition for instability is,

$$\frac{kv_0}{\omega_p} < \sqrt{2} \tag{2.17}$$

2.3 Landau Damping

The following section discusses the landau damping for Maxwellian distribution. The problem with this system is that it has resonant electrons, i.e electrons with velocity equal to phase velocity where the solutions with real frequency become discontinuous. The particles moving faster than the phase velocity of the wave lose energy and particles moving at speeds less than phase velocity gain energy. In Maxwellian distribution, since the number of particles with velocity less than the phase velocity is more than that of particles with velocities greater phase velocity, the extra energy come from wave which causes it to damp. Following is the derivation of the decay rate.

2.3.1 Derivation of damping rate

We will use Vlasov's equation for this derivation which is nothing but continuity equation and lorentz equation written in terms of distribution function $f(\vec{x}, \vec{v})$:

$$\frac{\partial f}{\partial t} + \vec{v}.\frac{\partial f}{\partial \vec{x}} + \frac{q}{m}(\vec{E} + \vec{v} \times \vec{B})\frac{\partial f}{\partial \vec{v}} = 0 \tag{2.18}$$

Assuming no magnetic field, electric field along x axis, perturbing the system and linearizing the resulting equation to the first order, we get:

$$\frac{\partial f_1}{\partial t} + \vec{v}.\nabla f_1 + \frac{q}{m}(\vec{E}_1).\nabla_v f_0 = 0 \tag{2.19}$$

Assuming solutions of the form $e^{i(kx-\omega t)}$, we get

$$-i\omega f_1 + ikv_x f_1 = -\frac{q}{m}E_1\frac{\partial f_0}{\partial v_x} \tag{2.20}$$

$$f_1 = -\frac{\frac{q}{m}E_1\frac{\partial f_0}{\partial v_x}}{-i\omega + ikv_x} \tag{2.21}$$

from Poisson's equation we get,

$$\nabla.\vec{E}_1 = \frac{\rho_1}{\epsilon_0} \tag{2.22}$$

$$\implies ikE_1 = \frac{\rho_1}{\epsilon_0} \tag{2.23}$$

$$= q\frac{\int f_1 d^3\vec{v}}{\epsilon_0} \tag{2.24}$$

$$\tag{2.25}$$

Substituting the above result in the previous equation we get the dispersion relation,

$$\implies k = -\frac{q^2}{m\epsilon_0}\int \frac{\frac{\partial f_0}{\partial v_x}d^3\vec{v}}{(\omega - kv_x)} = -\frac{q^2}{m\epsilon_0}\int \frac{\frac{\partial f_{0x}}{\partial v_x}dv_x}{(\omega - kv_x)} \tag{2.26}$$

which has a singularity at $v = \frac{\omega}{k}$. Doing this integral requires contour integration, with integration done in complex space of v. Integration can be done analytically for the case of small damping (small Img(ω/k)). Taking the contour to be along real v axis with a small semicircle around the phase velocity,

$$1 = \frac{q^2}{km\epsilon_0}(P(\int \frac{\frac{\partial f_{0x}}{\partial v_x}dv_x}{(\omega - kv_x)}) + \frac{1}{2}2\pi i(\frac{\partial f_{0x}}{\partial v_0})_{v=\frac{\omega}{k}}) \tag{2.27}$$

where P denotes the principal value of the integral, integral evaluated skipping the singularity. Assuming $f_0(v)$ to be small at $v = \omega/k$ we can taylor expand the denominator and integrate it by parts. Solving for Maxwellian distribution and imaginary ω ,gets us:

$$\omega_i = -\omega_p\sqrt{\frac{\pi}{8}}\frac{\omega_p^3}{k^3}\frac{1}{v_t^3}\exp(-\frac{m\omega_p^2}{2Tk^2} - \frac{3}{2}) \tag{2.28}$$

which is the damping rate of the wave.

2.4 Results

Following are the results of the various simulations done. See the captions for details.

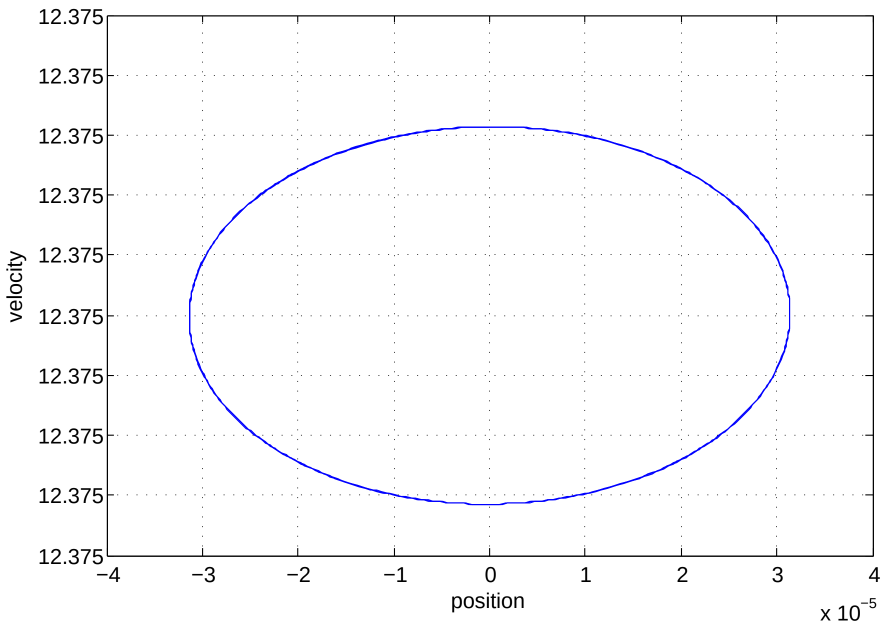

Figure 2.1: Phase space plot for a single particle.It is a circle describing the plasma oscillations. The parameters are:np=400, l=100,ng=100,q=-1,m=1,dt=0.01,tmax=10 and mode=1

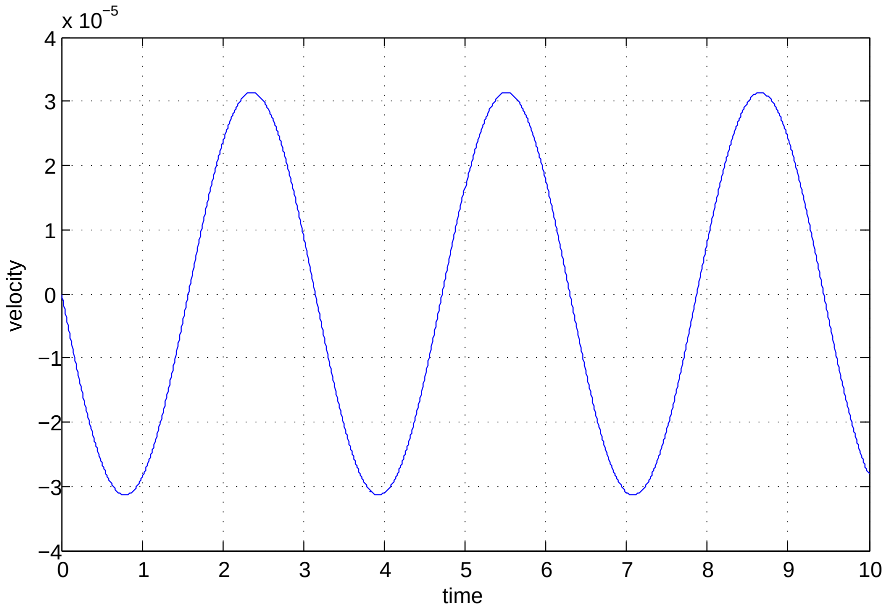

Figure 2.2: Velocity vs Time for a single particle. It is sinusoidal with frequency being the plasma frequency.The frequency got from this plot is 1.994662 in agreement with the theoretical value of 2. The parameters are:np=400, l=100,ng=100,q=-1,m=1,dt=0.01,tmax=10 and mode=1

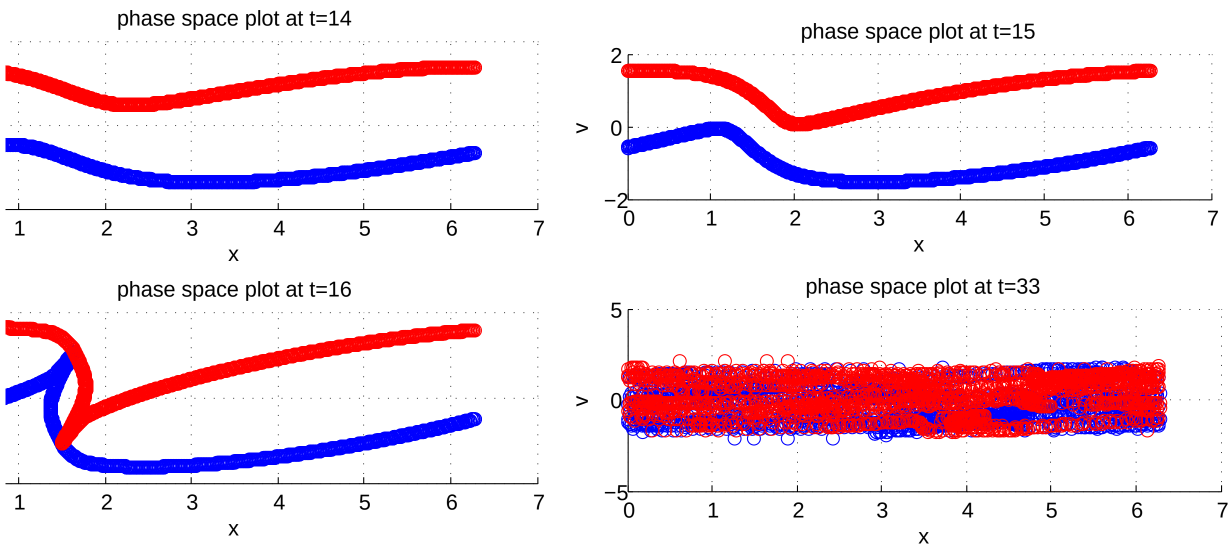

Figure 2.3: Amplitude vs Time for a system with two beams of opposite charges moving into each other. The parameters are: np=4000, l=2*pi, ng=2000, dx=l/ng, v=±1, q=± 0.056, q=m, t=0.01 and mode =1

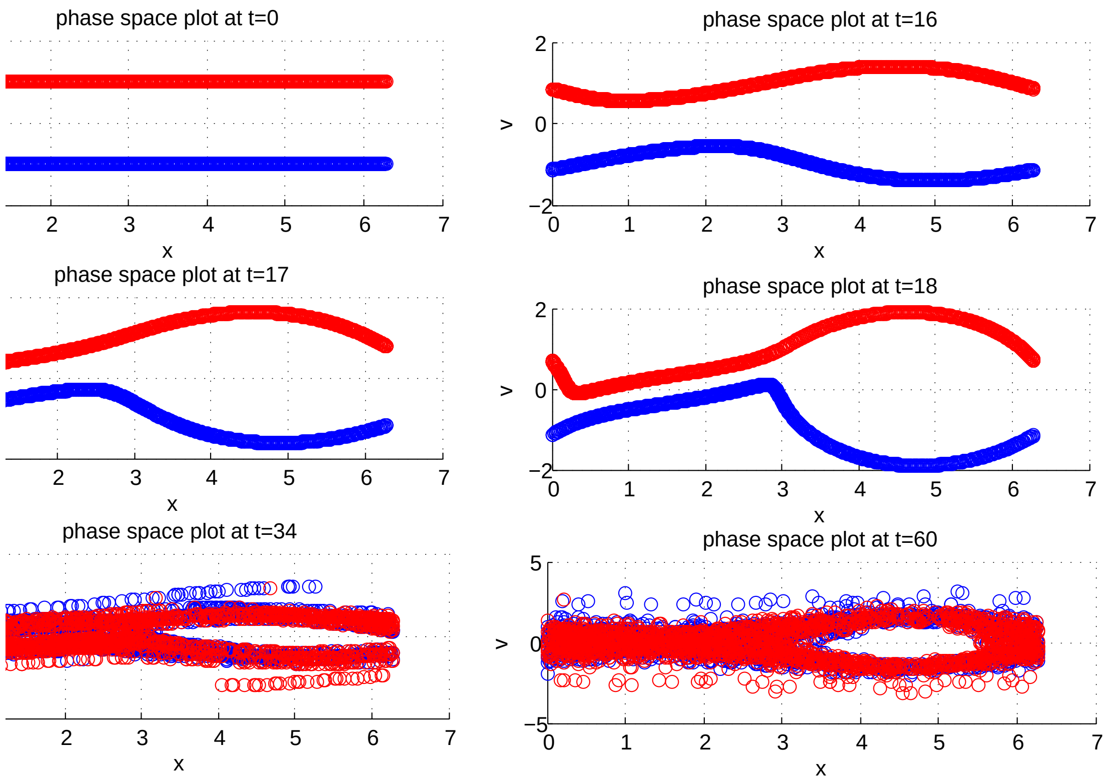

Figure 2.4: Phase space plots of a system with two beams of same charges moving into each other at various times. The parameters are: np=4000, l=2*pi, ng=2000, dx=l/ng, v=±1, q= 0.056 , q=m,t=0.01 and mode =1

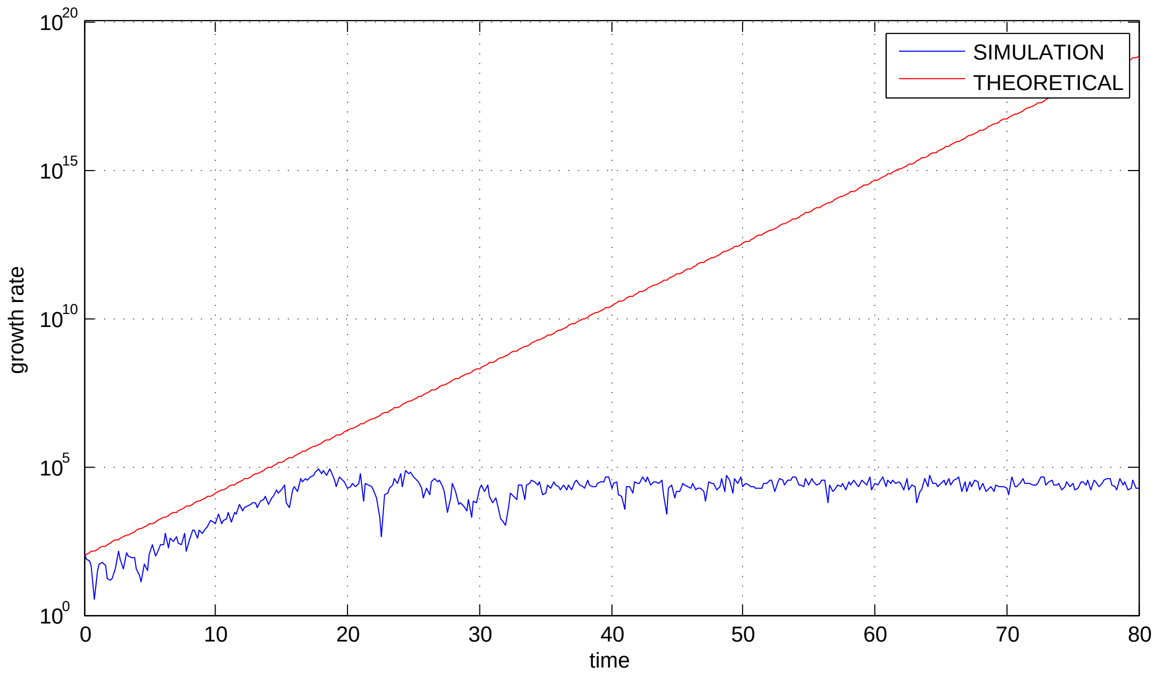

Figure 2.5: Amplitude vs time. The red line is the theoretical amplitude. System follows the linear growth for some time and then saturates.The parameters are: np=4000, l=2*pi, ng=2000, dx=l/ng, v=±1, q= 0.056 , q=m,t=0.01 and mode =1

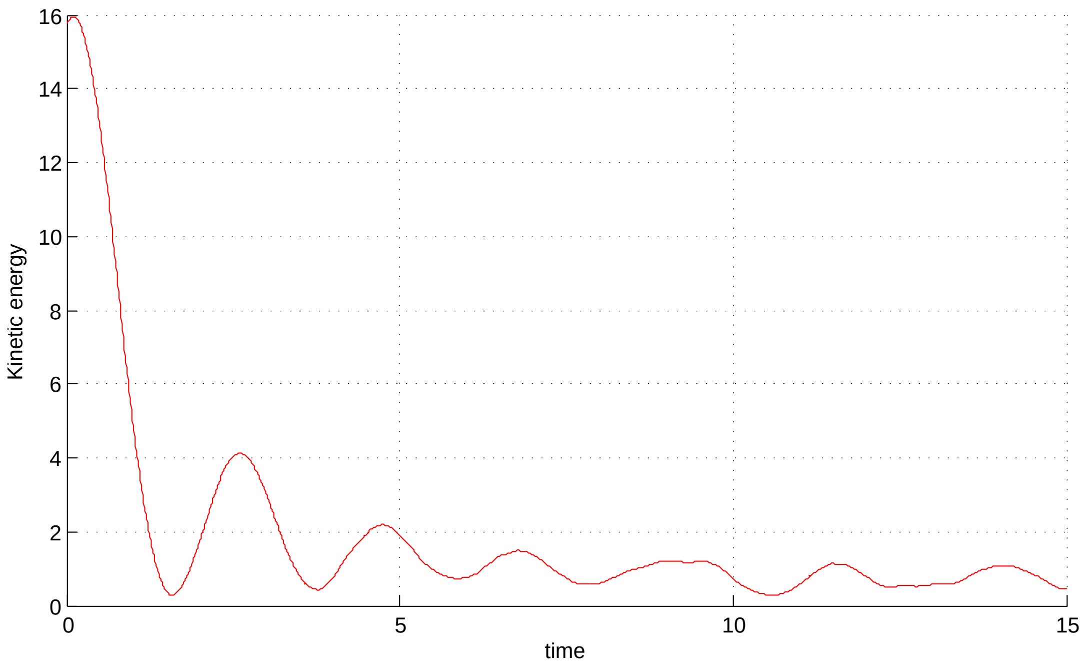

Figure 2.6: Kinetic energy vs time for a system showing the landau damping. randn function of MATLAB is used to initatize the system. Parameters used are: np=30000, l=2*2*pi, ng=500, qm=-1, q=-4.1888e-04, dt =0.1 and mode=1.

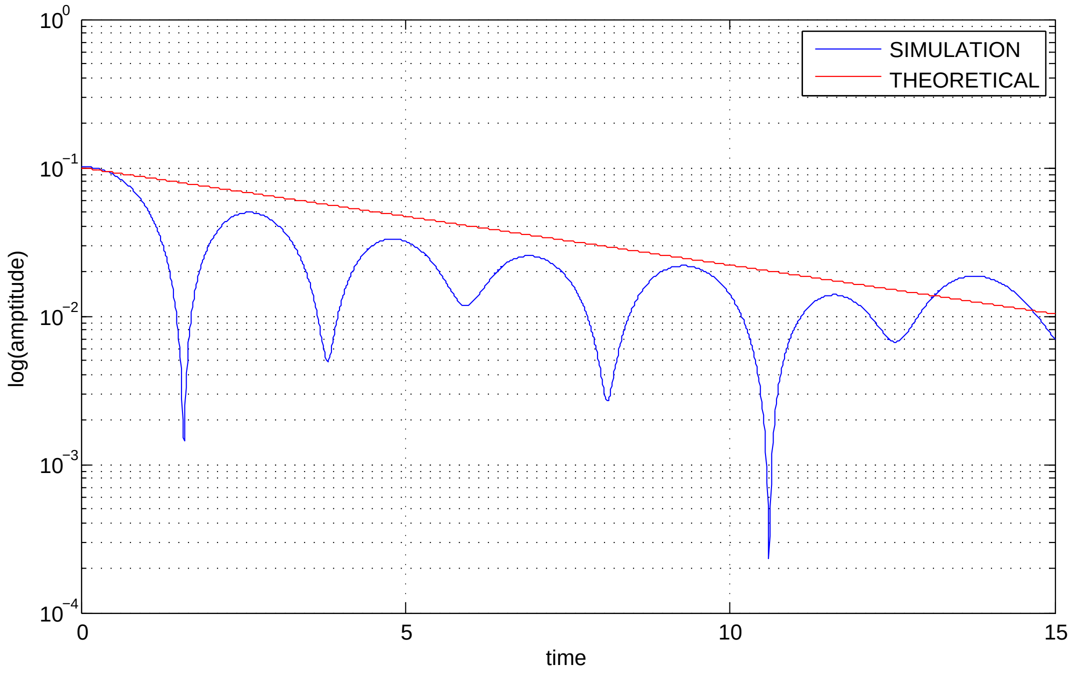

Figure 2.7: Amplitude vs time for system showing the landau damping. Red line is the theoretical rate of decay and blue is the one got from simulation. Parameters used are: np=30000, l=2*2*pi, ng=500, qm=-1, q=-4.1888e-04, dt =0.1 and mode=1.

Bibliography

[1] Birdsall ,Charles and Langdon ,Bruce. Plasma Physics via computer simulations, McGraw-Hill Book Company,1985.

[2] Markidis, Stefano. July 21, 2014, "Electrostatic Waves, Landau Damping, Two-Stream Instability and the Particle-in-Cell Method" .Web: www.pdc.kth.se/education/computational-plasma-physics/electrostatic-waves-and-particle-in-cell-method.

[3] Chen ,Francis. Introduction to plasma physics and controlled fusion, volume 1: plasma physics. Plenum Press, 1984.