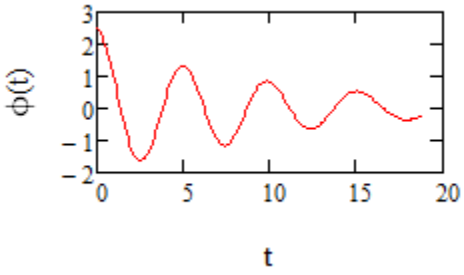
Particle-In-Cell Code for Landau Damping

This particle-in-cell (PIC) code solves for the motion of plasma electrons in an electrostatic wave and demonstrates Landau damping. At each time step, the equations of motion for the electrons are solved by iteration and the electric field is found from a solution of Poisson's equation. We assume that the ions are a uniform neutralizing background. The electrons are given a Maxwellian distribution of velocities. A wave is created by having the electron density vary as sin(Kx) at the initial time, where K is the wavenumber. The wave will be seen to damp by the Landau damping mechanism.



Plot of wave damping

The PIC code will show that the damping is greater if the waves have shorter wavelength and that the damping is negligible if the distribution of velocities is a "top hat" distribution that is flat.

The equations to be solved

For an electron, the equations of motion are: $\frac{ds}{dt} = v$ $\frac{dv}{dt} = \frac{-qE}{m}$

The particle position is s, m is the electron mass, and the other variables have their usual meaning. The grid points will be labelled x.

The electric field is found from the particle density: $\frac{dE}{dx} = -\frac{n_e q}{\varepsilon_0}$

Dimensionless units

From the plasma density we will define a characteristic time that is the inverse of the plasma frequency ω_{pe}. The characteristic distance is defined as the Debye length.

$\hat{t} = \sqrt{\frac{\varepsilon_0 m}{n_0 q^2}}$ $\hat{x} = \sqrt{\frac{\varepsilon_0 T_e}{n_0 q^2}}$

We divide t by the characteristic time above and we divide s by the characteristic distance to obtain the dimensionless forms of our equations:

$\bar{t} = t/\hat{t}$

$\bar{x} - x/\hat{x}$, etc.

$\frac{d\bar{s}}{d\bar{t}} = \bar{v}$ $\frac{d\bar{v}}{d\bar{t}} = -\bar{E}$ and $\frac{d\bar{E}}{d\bar{x}} = \frac{n_i q}{n_0 q} - \frac{n_e q}{n_0 q} = 1 - \bar{n}_e$

Note that the dimensionless electron density appears in the final math and is obtained by dividing n_e by the undisturbed density n_0. We assume that the ion density is fixed and equal to n_0. The net charge density is then (1 - n_e) in the dimensionless system of units.

Some additional characteristic scales are: $\hat{\phi} = \frac{T_e}{q}$ $\hat{v} = \frac{\hat{x}}{\hat{t}} = \sqrt{\frac{T_e}{m}}$ $\hat{E} = \frac{\hat{\phi}}{\hat{x}} = \sqrt{\frac{n_0 T_e}{\varepsilon_0}}$

Equations in finite-difference form

Let j be the grid point index, k be the particle index, and i be the time step index.

The finite difference form of the equation for E is: where E_J is the electric field at grid point j and Δx is the grid point spacing. In the code, the bar over the dimensionless variables is not used.

$$\overline{E}_{j+1} = \overline{E}_j + (1 - \overline{n}_{e,j})\Delta\overline{x}$$

The equivalent line in the PIC code is:

$$E_j \leftarrow E_{j-1} + \left[1.0 - \left(\frac{n_j}{n_0}\right)\right]\cdot\Delta x$$

The particle position is advanced in time using: The arrow indicates that the current value of S is being replaced with the value at the next time step. Δt is the time step.

$$S \leftarrow S + V\cdot\Delta t$$

The electron velocity is advanced using:

$$V \leftarrow V - E\cdot\Delta t$$

The computations are shortened by defining a variable $\Delta S = V^*\Delta t$ that is the distance moved between time steps: This equation for Δs is simply the equation for V multiplied by Δt.

$$\Delta S \leftarrow \Delta S - E\cdot\Delta t\cdot\Delta t$$

The equation advancing S becomes:

$$S \leftarrow S + \Delta S$$

In the PIC code, S will be a vector containing the positions of the k particles.
The equations for S and V are first order equations in time that advance to the next time step using the current value of acceleration. The error is of order Δt^2.

One dimensionless parameter L to vary

The dimensionless units system is based on the plasma density and temperature, hence these cannot be varied. The parameter than can be varied is the system length L expressed in dimensionless units. All physics experiments with L = 4, for example, will behave the same way if the same velocity distribution f(v) is used.

We specify L, the width of the simulation domain in dimensionless units. If L is smaller than 12, the wave is heavily damped.

$L := 16$

Highlighted values may be changed in order to simulate different experiments or to change the resolution.

We will place one wavelength of the wave, varying as sin (Kx), in the simulation domain. The wavenumber has the dimensionless value K defined as:

$$K := \frac{2\cdot\pi}{L} \qquad K = 0.393$$

Grid points and spatial resolution

In dimensionless units, we expect the spatial variation to be on a distance scale of order unity. Hence a grid spacing of 0.2 is sufficent to resolve the interesting physics. We will use x for the grid point locations and s (or S) for the electron locations. We will use the subscript i to indicate the values at successive times. Our program will output values of $M_{k,i}$ which is a matrix of values of the locations for the k^{th} particle at the i^{th} time step. The vector S of positions will be saved in successive columns of M at successive time steps.

$\Delta x := 0.2$ We specify the spatial resolution.

$jmax := \text{floor}\left(\frac{L}{\Delta x}\right)$ $jmax = 80$ This is the number of x grid points we need for the desired spatial resolution and desired L.

We will define a grid with jmax+1 points separated by distance Δx:

$j := 0 .. jmax$ $x_j := \Delta x \cdot j$ $x_{jmax} = 16$ Recall that L = 16.

Time step

A thermal velocity of 1 in the dimensionless units and the grid spacing $\Delta x = 0.2$ results in a typical particle crossing a cell in a time of 0.2 in dimensionless time units. We will use a time step Δt that is half the time for the particle to cross a cell. The code will use a distribution in velocities and the fastest particle could cross several cells in a time step. This would introduce some error because the value of E used in the iteration procedure is the value at the current particle position at the current time.

The characteristic velocity; $V := 1$ The time step: $\Delta t := 0.5 \cdot \frac{\Delta x}{V}$ $\Delta t = 0.1$

Particle number per cell

The dimensionless background charge density is unity which can be obtained by having 100 electrons per unit length each with charge 0.01, or any other combination of density and charge that generates unit charge density. There is random noise in the code because the number of electrons in a unit length takes on discrete values such as 99, 101, 97 etc. The noise would be 10 electrons (the square root of 100) if the average number is 100. Experience shows that 100 electrons per cell with about 5 cells per unit length provides sufficiently low noise that a wave of small amplitude (0.1 for example) can be seen above the noise.

$Ncell := 100$ The number of electrons in a cell.
Ncell determines the level of noise in the charge density calculation.

Initial electron positions to create a standing wave

$kmax := jmax \cdot Ncell$ kmax is the number of particles to fill jmax cells with Ncell particles. The particles will be numbered 0 through kmax-1.

$kmax = 8000$

$k := 0 .. kmax - 1$ The index of the particle positions s_k.

We will create a sinusoidal electric field at the initial time by modulating the electron density.

$Ewave := 0.5$ Initial amplitude of the electric field of the desired wave.

Recall that n = -dE/dx, hence if the electric field amplitude is Ewave*sin(Kx) then the density is 1-K*Ewave*cos(Kx). The line below finds the number of particles that should be in each cell. The cell with number j is between grid points j and j+1. The (j+0.5) below evaluates the density at the center of the cell rather than at an end. The **round** function rounds the number in the cell to the nearest integer.

$$Cell_j := round\left[Ncell \cdot \left[1 - K \cdot Ewave \cdot \cos\left[2 \cdot \pi \cdot \frac{(j + 0.5)}{jmax}\right]\right]\right]$$

Here we adjust the number in the last cell, $Cell_{jmax-1}$, in case a rounding has generated more or less than kmax particles.

$$Cell_{jmax-1} := kmax - \sum_{j=0}^{jmax-2} Cell_j \qquad \sum_{j=0}^{jmax-1} Cell_j = 8000$$

The program loop at right distributes the particles randomly in each cell. A random distance between 0 and Δx is added to x_j, the location of the left boundary of the cell. A uniform spacing within a cell could be used, but this would be misleading about the level of random noise in the program.

```
s :=  k ← 0
      for j ∈ 0 .. jmax − 1
        for kk ∈ 0 .. Cell_j − 1
          s_k ← x_j + Δx·rnd(1)
          k ← k + 1
      s
```

The average density is: $n0 := \frac{kmax}{L}$ $n0 = 500$

Charge density based on electron positions

The first step in finding the electric field E is to find the electron density. The program loop below determines electron density at grid points by counting electrons between grid points and using a weighting factor to assign a fraction of the density to nearest two points. If the particle is 10% of the way between a first and second grid points (for example), 90% of the charge is assigned to the first grid point and 10% is assigned to the second. The program loop is written as a function of the locations s_k so that only one line of code (the function call) is needed to find the number density in our PIC code. The function **floor**($s_k/\Delta x$) returns the index j of the grid point to the left of the particle. This program loop was explained in more detail in the first PIC code exercise.

We assume periodic boundary conditions, hence the grid point at the right hand boundary is the

same point as the grid point at the left hand boundary. The particle weightings assigned to the point j = jmax should also have been assigned to the point j = 0. Similarly, the particles assigned to j = 0 should be assigned to jmax. These assignments are accomplished in the program loop by adding the particle weightings at j = 0 and j = jmax.

In the program loop, the vector N is the weighted number of particles assigned to the grid point and n = N/Δx is the particle density per unit length evaluated at the grid point.

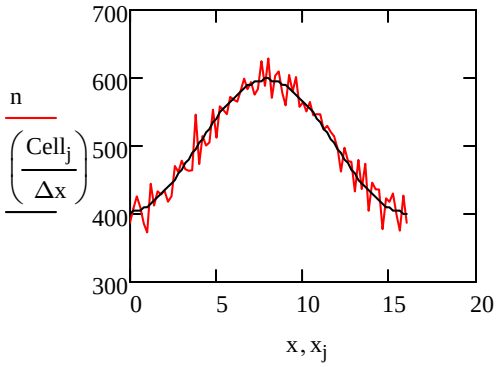
$\text{inv}\Delta x := \frac{1}{\Delta x}$ This variable is the inverse of Δx. The loop will execute more quickly if the divisions by Δx are replaced with multiplications by invΔx.

$$\begin{aligned}
&\text{NumberDensity}(s) := \\
&\quad n_{jmax} \leftarrow 0 \\
&\quad N_{jmax} \leftarrow 0 \\
&\quad \text{for } k \in 0 .. \text{rows}(s) - 1 \\
&\quad\quad \text{jlower} \leftarrow \text{floor}\left(s_k \cdot \text{inv}\Delta x\right) \\
&\quad\quad N_{\text{jlower}+1} \leftarrow \left(s_k - x_{\text{jlower}}\right) \cdot \text{inv}\Delta x + N_{\text{jlower}+1} \\
&\quad\quad N_{\text{jlower}} \leftarrow \left(x_{\text{jlower}+1} - s_k\right) \cdot \text{inv}\Delta x + N_{\text{jlower}} \\
&\quad N_0 \leftarrow N_0 + N_{jmax} \\
&\quad N_{jmax} \leftarrow N_0 \\
&\quad n \leftarrow N \cdot \text{inv}\Delta x
\end{aligned}$$

The next line calls the NumberDensity function.

$n := \text{NumberDensity}(s)$

Plot of the electron density



The plot shows the noise due to the random locations of the particles within cells.

Initial electric field at grid points based on electron positions

Recall that the electric field is found using: $$E_j \leftarrow E_{j-1} + \left[1.0 - \left(\frac{n_j}{n_0}\right)\right]\cdot\Delta x$$

This program loop implements the integration, starting at x = 0 with boundary condition $E_0 = 0$ even though zero may not be the correct boundary condition.

$$E := \left| \begin{array}{l} E_{jmax} \leftarrow 0 \\ \text{for } j \in 1..jmax \\ \quad E_j \leftarrow E_{j-1} + \left[1.0 - 0.5\cdot\left(\frac{n_j + n_{j-1}}{n0}\right)\right]\cdot\Delta x \\ E \end{array}\right.$$

The density in a cell is obtained by averaging the values at the adjacent grid points. The result is correct to order $(\Delta x)^2$ because the n value is "centered" in x. The time integration in our code is not time centered.

Now we will find the potential φ on the grid points by assuming that the potential on the left boundary is zero and integrating E. For second order accuracy, we will use the E field value half way between grid points, and we will find this value by averaging the value at the grid points.

The equation relating φ to E is: $$E = -d\phi / dx$$

The solution for φ in finite difference form is: $$\phi_j = \phi_{j-1} - 0.5\left(E_j + E_{j-1}\right)\Delta x$$

This program loop creates an array containing the φ values at the grid points:

$$\varphi(E) := \left| \begin{array}{l} \varphi_{jmax} \leftarrow 0 \\ \text{for } j \in 1..jmax \\ \quad \varphi_j \leftarrow \varphi_{j-1} - 0.5\cdot\left(E_j + E_{j-1}\right)\cdot\Delta x \\ \varphi \end{array}\right.$$

The solutions obtained for E and φ are not unique because of unspecified constants of integration. We will require that the potential be zero at the left and right boundaries (periodic boundary conditions). To implement this condition in the code, we begin the integration for E using a zero value for E at the left boundary. We then integrate E to obtain φ at the right boundary. If φ at the right boundary is nonzero, we add a constant E so that φ at the right boundary becomes zero.

This is the constant electric field (constant of integration) that should be added to our trial solution for E. $$Econst := -\frac{\varphi(E)_{jmax}}{L}$$

This line "fixes" E so that it satisfies the boundary condition: In this command, Mathcad subtracts a scalar from each entry in a vector of values. $$E \leftarrow E + \frac{\varphi_{jmax}}{L}$$

This program loop defines a function Efield(n) that returns a vector containing the E that satisfies the equations and the boundary conditions. We will use this function in our PIC code.

$$\text{Efield}(n) := \left| \begin{array}{l} E_0 \leftarrow 0 \\ \text{for } j \in 1 \,..\, \text{jmax} \\ \quad E_j \leftarrow E_{j-1} + \left[1.0 - 0.5 \cdot \left(\frac{n_j + n_{j-1}}{n0}\right)\right] \cdot \Delta x \\ \varphi \leftarrow -0.5 \cdot \Delta x \sum_{j=1}^{\text{jmax}} \left(E_{j-1} + E_j\right) \\ E \leftarrow E + \frac{\varphi}{L} \end{array} \right.$$

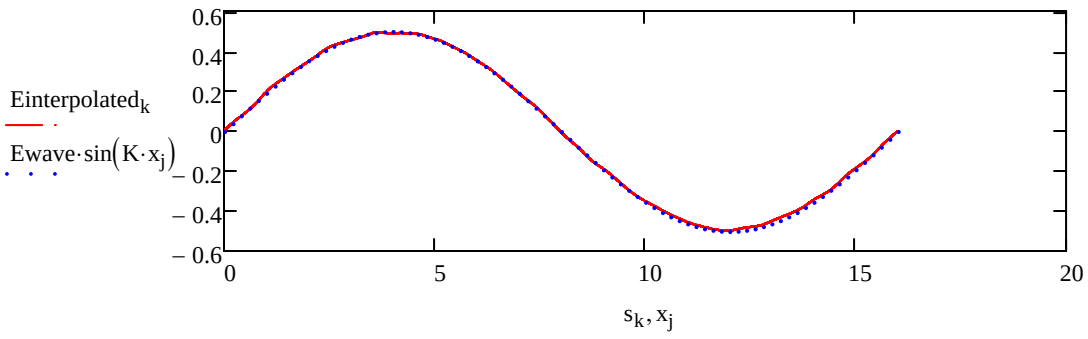
Initial e+lectric field at particle locations

In order to find the electric field at the particle locations (not necessarily at grid points), we will need to interpolate using the values at the nearby grid points. We will use the interpolation function **interp** which fits a second order polynomial locally. The coefficients for the polynomial are stored in a variable coeffs which is obtained by calling the function **pspline**.

This function call finds the vector array of coefficients: $\text{coeffs} := \text{pspline}(x, E)$

This interpolation routine finds the electric field at the particle locations. $\text{Einterpolated}_k := \text{interp}\left(\text{coeffs}, x, E, s_k\right)$

Plot of the electric field of the initial wave



Recall that we defined the wave amplitude as: The graph shows that we assigned the density values correctly for the desired Ewave value.

$\text{Ewave} = 0.5$

Specifying an electron distribution function f(v)

We will need a program loop to assign an initial thermal velocity to each electron. The velocity is selected randomly from a specified distribution. We will first define a Maxwellian distribution f(v) in order to observe Landau damping and, second, a top hat distribution that will not cause Landau damping.

This is the Maxwellian distribution function in our dimensionless system of units:

$$f(v) := \frac{1}{\sqrt{2\pi}} \cdot e^{-\left(\frac{v^2}{2}\right)}$$

$$\int_{-\infty}^{\infty} f(v)\, dv = 1$$ This line verifies that our distribution is normalized to unity.

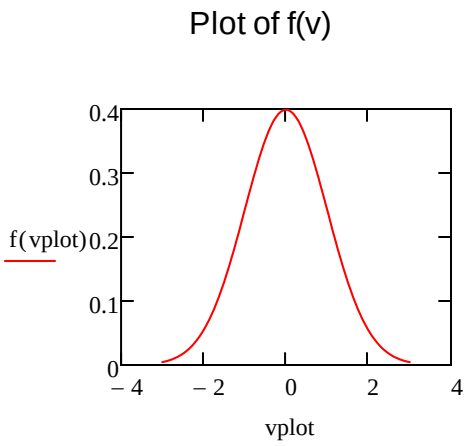
$$Vrms := \sqrt{\int_{-\infty}^{\infty} v^2 \cdot f(v)\, dv} \qquad Vrms = 1$$ This line verifies that the rms velocity is unity.

$vmax := 3$ Define a range of v values to use for plots that extends from -vmax to +vmax.

$m := 0 .. 100$ $\quad vplot_m := -vmax + \frac{2 \cdot vmax \cdot m}{100}$ The range is spanned by 100 points.

We will not select particles with v > vmax or v < -vmax because f(v) is small at these velocities.

We will not select particles with v > 3 or v < -3 because f(v) is small at these velocities.



Alternative distribution function

The disabled function at right is a top hat distribution that has a value of 0.25 when the velocity is between -2 and +2. This distribution shows no Landau damping and could be substituted for f(v).

$$f(v) := \begin{cases} 0.25 & \text{if } v < 2 \wedge v > -2 \\ 0 & \text{otherwise} \end{cases}$$

Try it: Remove the "disable" from the top hat distribution and observe the absence of damping.

The variable test is set to a value a little larger than the maximum value of f(v) and is used in the program loop below.

$$test := \max(f(vplot))$$

$$test := 1.1 \cdot test$$

$$test = 0.439$$

Program loop to select particles from a distribution function

This program loop chooses a velocity randomly on the interval -vmax to vmax and then chooses a random test value from 0 to test. If the value of f(v) is less than the test value, the v value is kept, otherwise it is rejected and a new v is created. Because the probability of v being kept is proportional to amplitude of f(v), the selected values of v have a probability proportional to f(v). This procedure is called the rejection method.

```
Vselect(b) := | v ← 0
              | y ← test
              | while y > f(v)
              |     v ← 2·vmax·(rnd(1) − 0.5)
              |     y ← test·rnd(1)
              | v
```

The random test value Vselect(b) is a function of a variable b that is not specified. Why? If Vselect is made a function of a variable, it is evaluated each time it is called. If Vselect is not made a function, then it is given a value only once in the program. Hence making it a function is a "work around" that causes the program loop to be executed each time Vselect(b) is called.

Test of the velocity selector Vselect(b)

We will call the function Vselect(b) many times and **histogram** the result.

$Ntest := 5000$ Number of Vtest values.

```
Vtest := | hmax ← 5000
         | G_hmax ← 0
         | for h ∈ 0 .. hmax
         |   G_h ← Vselect(2)
         | G
```

Below we create a histogram of velocities:

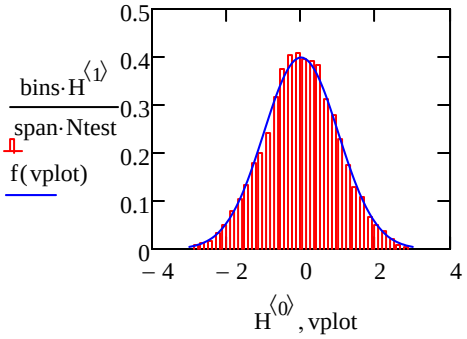
$bins := 30$

$span := \max(Vtest) - \min(Vtest)$

$H := histogram(bins, Vtest)$

Histogram of f(v)

Our program Vtest is working correctly because the scaled histogram of the velocities agrees with f(v). The scale factor for the histogram height is determined from the bin width (span/bins) and the total number of test particles Ntest.



Number of iterations

$Noscillations := 3$

We will follow the electrons for sufficient time for about 3 oscillation periods of the wave:

$$iters := floor\left(Noscillations \cdot \frac{2 \cdot \pi}{\Delta t}\right)$$

$iters = 188$

Shorten caclulation time

$\Delta t \Delta t := \Delta t^2$ We will use this variable in place of Δt^2 in order to reduce the number of multiplications. This variable appears in the innermost **for** loop which is executed the largest number of times.

$iters \cdot kmax = 1.504 \times 10^6$ Number of times the innermost loop is calculated.

We also reduce execution time by using the vector variable $E\Delta t\Delta t$ which is defined as $E*\Delta t*\Delta t$.

The program loop

Recall that: $kmax = 8000$ $jmax = 80$ $iters = 188$

We will use the **time()** function to measure the time to do the program loop: $StartTime := time(2)$

```
M := | E_jmax ← 0
     | n_jmax ← 0
     | M_(kmax−1, iters) ← 0
     | ΔS_(kmax−1) ← 0
     | EΔtΔt_(kmax−1) ← 0
     | S ← s
     | for k ∈ 0 .. kmax − 1
     |     V ← Vselect(3)
     |     ΔS_k ← V·Δt
     | for i ∈ 0 .. iters
     |     n ← NumberDensity(S)
     |     E ← Efield(n)
     |     vs ← pspline(x, E)
     |     EΔtΔt ← ΔtΔt·interp(vs, x, E, S)
     |     ΔS ← ΔS − EΔtΔt
     |     S ← S + ΔS
     |     for k ∈ 0 .. kmax − 1
     |         S_k ← S_k + L  if S_k < 0
     |         S_k ← S_k − L  if S_k > L
     |     M^⟨i⟩ ← stack(S, ΔS/Δt)
     | M
```

These lines initialize the vectors E, n, S, ΔS and the matrix of answers M.

Recall that s is the initial vector of electron positions with modulation by the wave.

Mathcad "knows" that EΔtΔt is a vector with an element for each S value.

If the electron moves outside the domain 0 to L, it is moved into the domain (recycled) using the notion of periodic conditions. For example, if the particle moves to L+0.01, the new location is 0.01.

$EndTime := time(2)$

The time to run the loop is: $EndTime - StartTime = 4.55$ seconds

The answer matrix M contains the postions S stacked on top of the velocites ΔS/Δt.

We will recover the velocities from the bottom half of the matrix: $rows(M) = 16000$

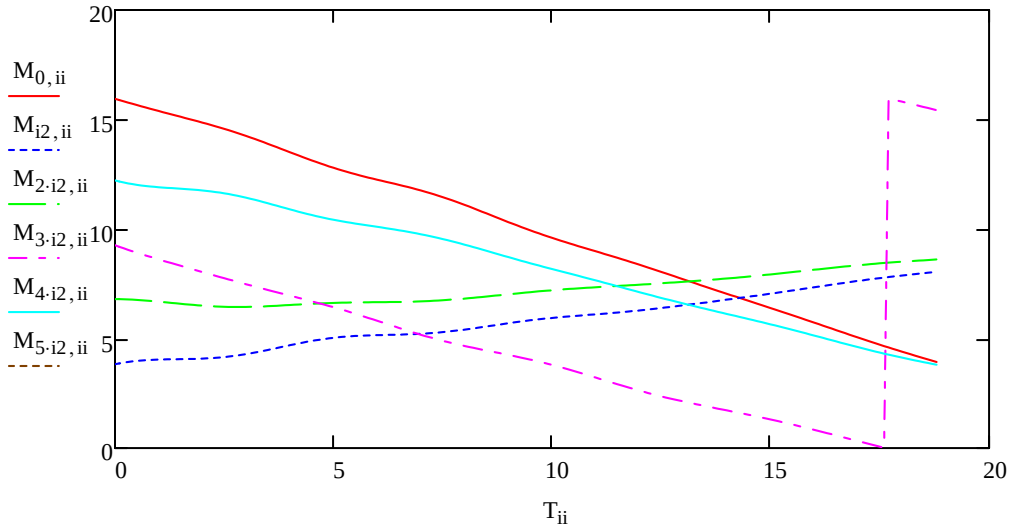
$Velocities := submatrix(M, kmax, kmax + kmax - 1, 0, iters)$

$rows(Velocities) = 8000$ $M := submatrix(M, 0, kmax - 1, 0, iters)$

We define the time T for each time step to use in plots: $ii := 0 .. iters$ $T_{ii} := ii \cdot \Delta t$

$i2 := floor\left(\frac{rows(M)}{5}\right)$ We will plot trajectories of electrons numbered 0, i2, 2*i2, 3*i2, etc.

Plot of several electron trajectories showing recycling at the boundaries



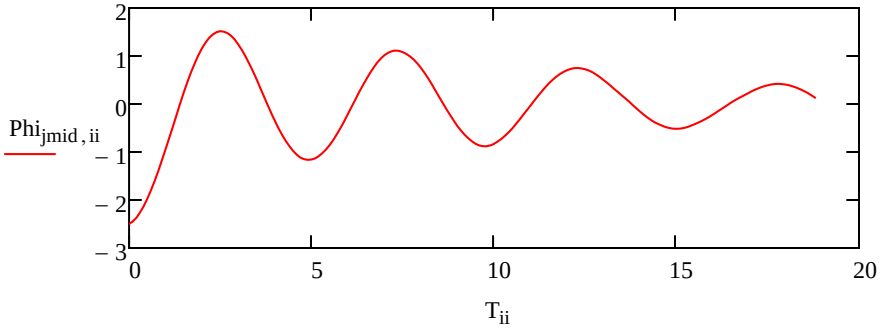
Program loop for potential φ(t)

This program loop takes the matrix M and calculates the potential at the grid points at each time step. The columns of the new matrix Phi contain the values of φ at grid points. We will plot φ(t) at the middle of the domain.

$jmid := floor\left(\frac{jmax}{2}\right)$ Index of the middle point.

```
Phi := | Phi_{jmax, cols(M)−1} ← 0
       | for i ∈ 0 .. cols(M) − 1
       |     s ← M^⟨i⟩
       |     n ← NumberDensity(s)
       |     E ← Efield(n)
       |     Phi^⟨i⟩ ← φ(E)
       | Phi
```

Plot of φ(t) at the midpoint of the domain, showing wave damping



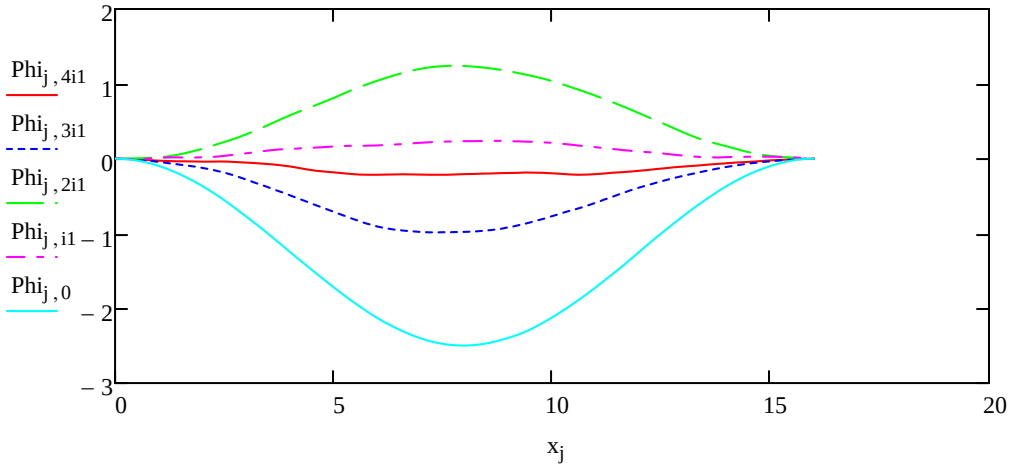
K value:

K = 0.393

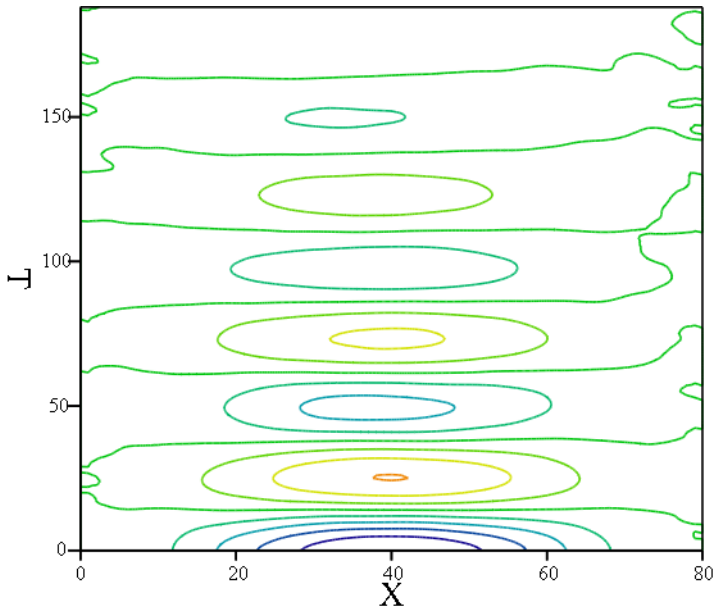
L = 16

$i1 := \text{floor}\left(\frac{\pi}{2 \cdot \Delta t}\right)$ The plot below is at times i1, 2*i1, 3*i1, etc.

Plot of φ(x) at several times



Contour plot of the potential as a function of time



Phi

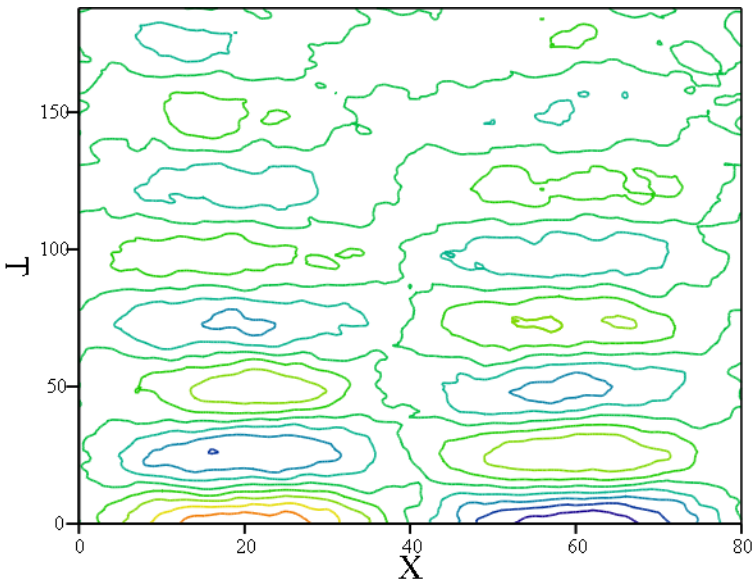
The vertical axis is the index i of the time step. The horizontal axis is the index j for the distance x.

Electric field as a function of time

This program loop takes the matrix of particle positions M and calculates the electric field at the grid points at each time step. The columns of Ematrix contain the jmax values of E at successive time steps.

```
Ematrix := | Ematrix_(jmax, cols(M)−1) ← 0
           | for i ∈ 0 .. cols(M) − 1
           |     s ← M^⟨i⟩
           |     n ← NumberDensity(s)
           |     E ← Efield(n)
           |     Ematrix^⟨i⟩ ← E
           | Ematrix
```

Contour plot of E(x,t)



Ematrix

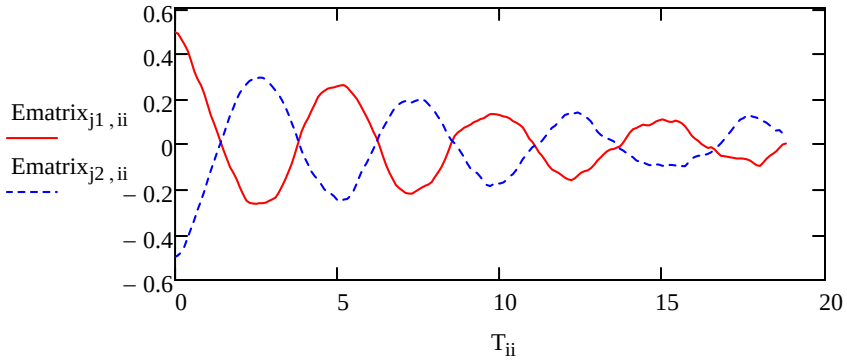
The vertical axis is the index i of the time step. The horizontal axis is the index j for the distance x.

Try it: Observe the change in the plot when Ncell is changed. The plot becomes smoother when the number of particles in a cell is increased to 400, for example. What happens when Ncell is changed to 25?

The electric field has maxima and minima at 1/4 and 3/4 of the distance across the simulation domain.
Below we find the indices j1 and j2 corresponding to these positions.

$j1 := \text{floor}\left(\frac{1}{4}\cdot jmax\right)$ $j2 := \text{floor}\left(\frac{3}{4}\cdot jmax\right)$ $C := 1$

Plot of E(x,t) at the x positions where E(x,t) is greatest



Recall that we defined an initial wave amplitude: $Ewave = 0.5$

Energy conservation as a check on accuracy in our PIC code

We will examine the "goodness" of our code by looking at energy conservation. We will find the kinetic energy of the particles and the energy in the wave and see if their sum is constant with time.

Kinetic Energy

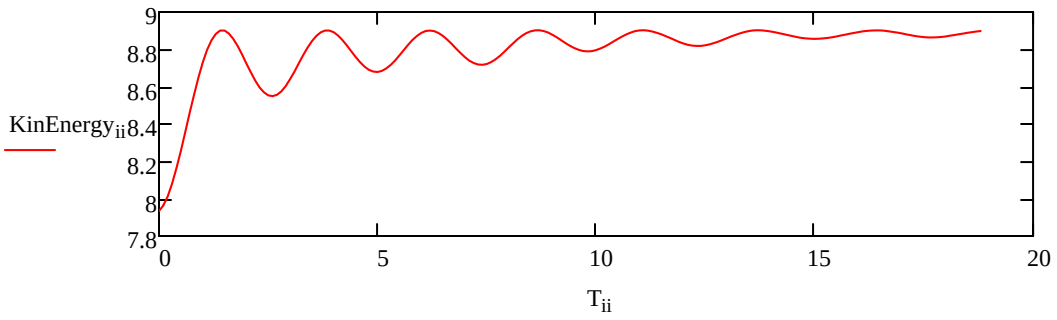
Create a vector that is the sum over particles of the kinetic energy 0.5 v^2 at each time step. Recall that the vector Velocities contains the velocities of the particles.

$Nparticles := rows(Velocities)$ $Nparticles = 8000$ Number of particles.

$$KinEnergy_{ii} := \frac{0.5 \cdot L}{Nparticles} \cdot \sum_{k=0}^{Nparticles-1} \left(Velocities_{k,ii}\right)^2$$

The sum of the squares of the velocities divided by the number of PIC particles gives the mean squared velocity. The dimensionless density is unity, hence the "true" number of particles is L. The rms velocity is multiplied by L to obtain the total kinetic energy of particles.

Plot of kinetic energy of particles as a function of time



Wave energy

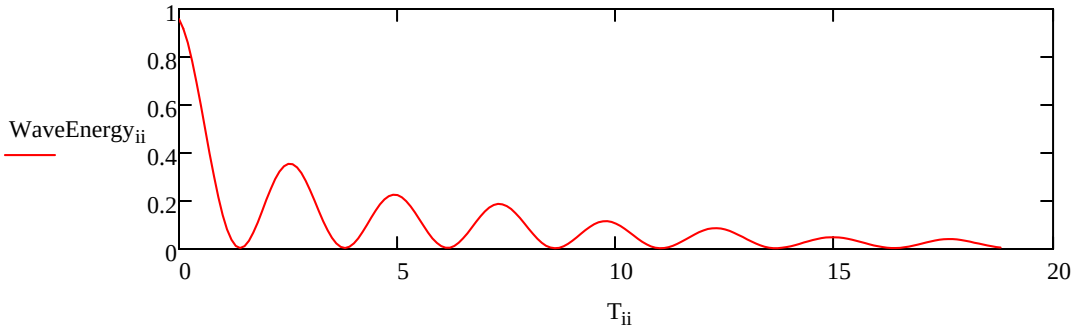
Recall that Ematrix has the E values at the grid points $rows(Ematrix) = 81$

We will integrate 0.5 E^2 with distance to find the electric field energy:

$$WaveEnergy_{ii} := 0.5 \cdot \Delta x \cdot \sum_{k=1}^{rows(Ematrix)-1} \left(\frac{Ematrix_{k,ii} + Ematrix_{k-1,ii}}{2}\right)^2$$

The above sum is the numerical equivalent of the integral of 0.5 E^2 over the distance L. E is evaluated at the center of the cell by averaging the values at the adjacent grid points.

Plot of the electric field energy in the wave



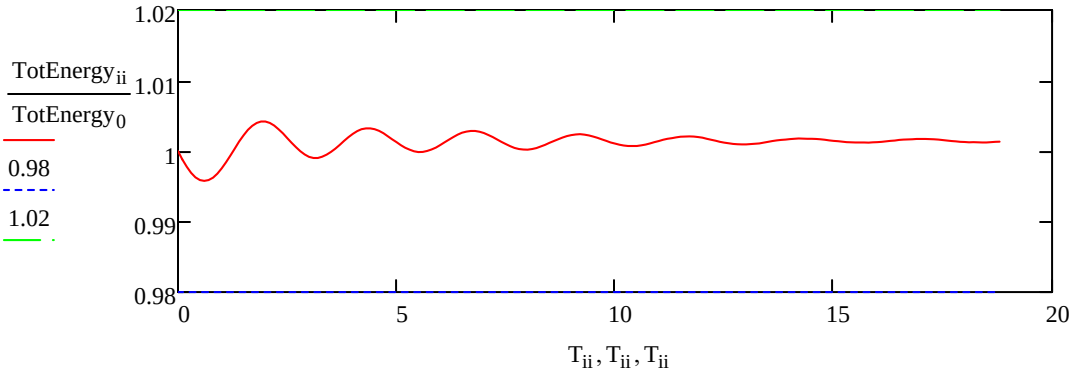
Total Energy is nearly conserved

$$TotEnergy := WaveEnergy + KinEnergy$$

$$\frac{\max(TotEnergy)}{TotEnergy_0} = 1.004 \qquad \frac{\min(TotEnergy)}{TotEnergy_0} = 0.996$$

The maximum, minimum, and starting energies are compared.

Normalized total energy as a function of time



The total energy has been divided by the initial energy to obtain a normalized value near unity. The graph will "autoscale" to make the variation look large, hence we have added points at 98% and 102% of the starting value so that the variation is shown on an appropriate scale.

Note that the error in the total energy is less than 1%, indicating that our PIC code conserves energy (almost).

Modification of the distribution function by wave-particle interaction

To see the effect of the wave on f(v), we can plot f(v) before and after the wave is damped. The last column of the matrix Velocites contains the velocities at the last time step.

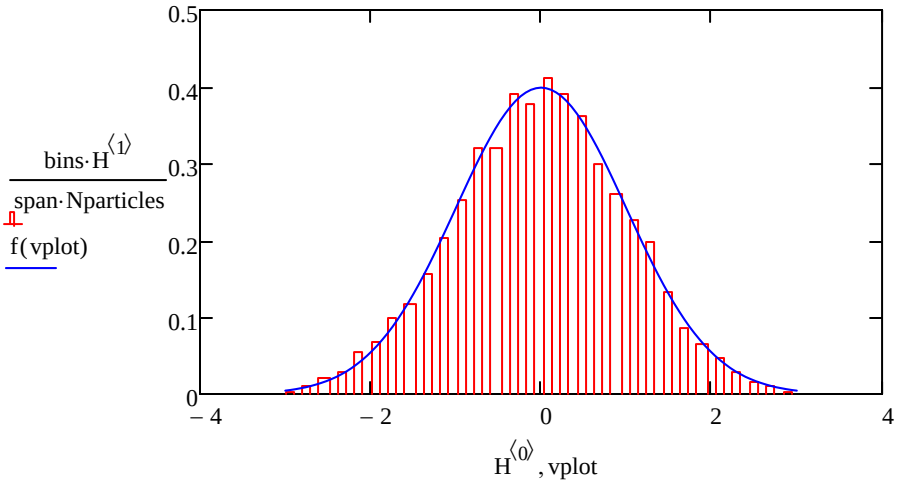
$\text{cols}(\text{Velocities}) - 1 = 188$ This is the same as the number of time steps. $\text{iters} = 188$

These commands make a histogram of the velocities at the zeroth time step:

$\text{span} := \max\left(\text{Velocities}^{\langle 0 \rangle}\right) - \min\left(\text{Velocities}^{\langle 0 \rangle}\right)$ $\text{span} = 6.045$

$H := \text{histogram}\left(\text{bins}, \text{Velocities}^{\langle 0 \rangle}\right)$

Histogram of f(v) before damping



These commands make a histogram of the velocities at the last time step:

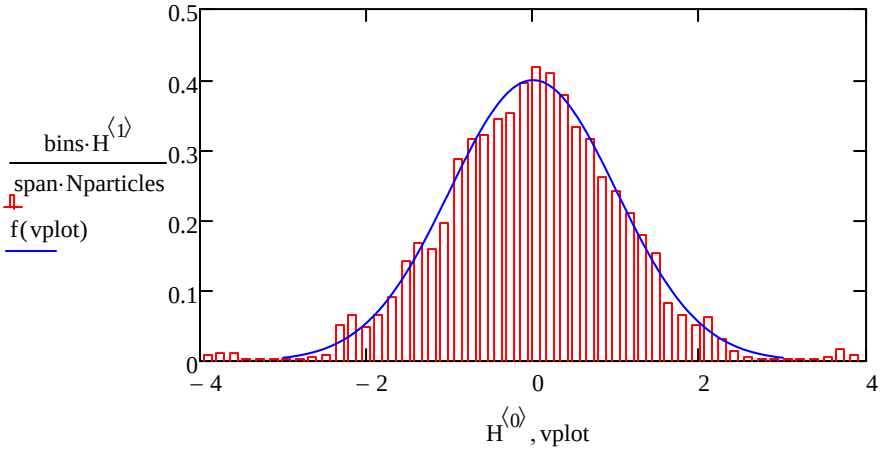
$\text{bins} := 50$

$H := \text{histogram}\left(\text{bins}, \text{Velocities}^{\langle \text{iters} \rangle}\right)$

These are the maximum and minimum values in the histogram chart:

$\text{span} := \max\left(\text{Velocities}^{\langle \text{iters} \rangle}\right) - \min\left(\text{Velocities}^{\langle \text{iters} \rangle}\right)$ $\text{span} = 7.92$

Histogram of f(v) after damping



Comparison of the histograms shows that there are more particles near velocity 4 after the wave has damped as a result of the transfer of energy from the wave to the particles.

Reference :

"Plasma Physics via Computer Simulation" by C.K. Birdsall and A.B Langdon, Institute of Physics, Bristol and Philadelphia, 1991.

Exercises

1. Can we verify the dispersion relation for electrostatic waves?

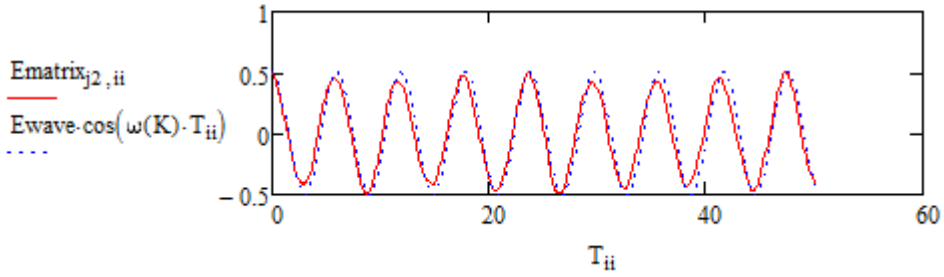
In our dimensionless units system with the velocity scale being the square root of T_e/m, the dispersion relation is:

$$\omega(K) := \sqrt{1 + 3 \cdot K^2}$$

Below is an image of a plot of oscillations with L = 32 which results in a wave very near the plasma frequency.

Find the K value again: $L = 32$ $K := \frac{2 \cdot \pi}{L}$ $K = 0.196$ $\omega(K) = 1.056$

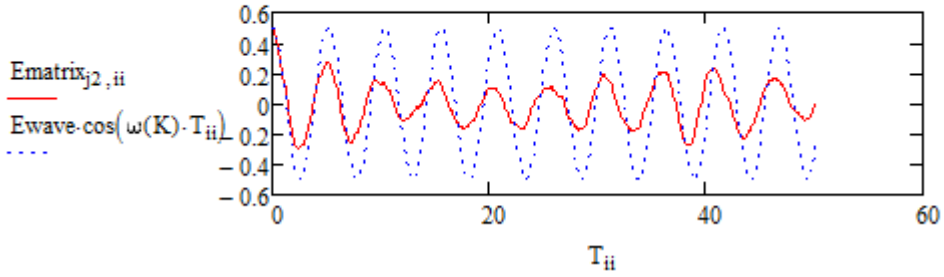
Here we plot the expected wave on the same plot as the wave itself:



Here is an image of a plot that shows the oscillations with L = 16.

Find the K value again: $L = 16$ $K := \frac{2 \cdot \pi}{L}$ $K = 0.393$ $\omega(K) = 1.209$

Here we plot the expected wave (which damps) on the same plot as the wave itself:



Note that there are 8 oscillations in a time of 16π with L = 32 and 9.5 oscillations with L = 16. The observed oscillation frequency is slightly higher than the frequency from the dispersion relation. The agreement is better if the 3 in the dispersion relation is changed to 3.3. Nevertheless, we see that the frequency is nearest the plasma frequency for long wavelength and is higher at short wavelength.

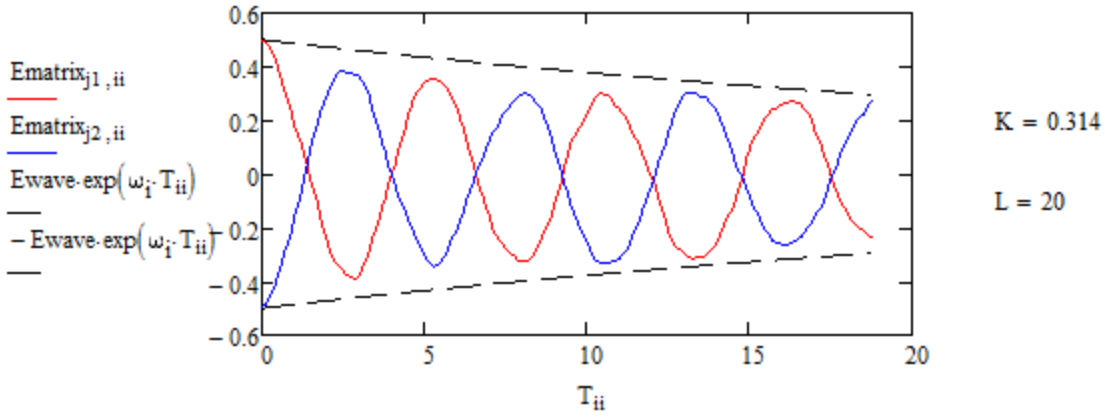
2. Is the observed damping near the theoretical value?

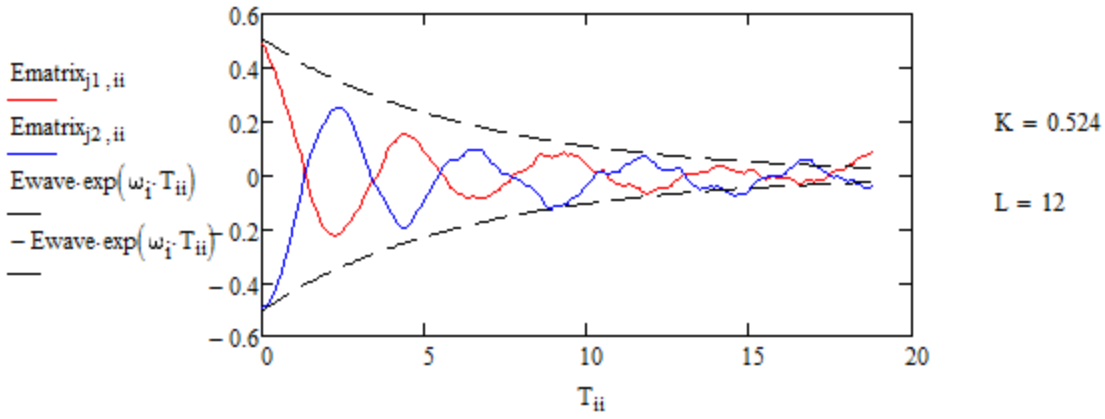
The theoretical expression for Landau damping is (in our dimensionless units):

$$\omega_i := -0.22\cdot\sqrt{\pi}\cdot\left(\frac{1}{K\cdot\sqrt{2}}\right)^3\cdot e^{-\left(\frac{1}{2K^2}\right)} \qquad \omega_i = -0.089$$

where ω_i is the imaginary part of the frequency. Below are images of plots of the expected decay of the wave envelope with the electric field from the PIC code. The plots show damping for two values of K.

Comparison of theoretical damping and PIC code damping for two K values

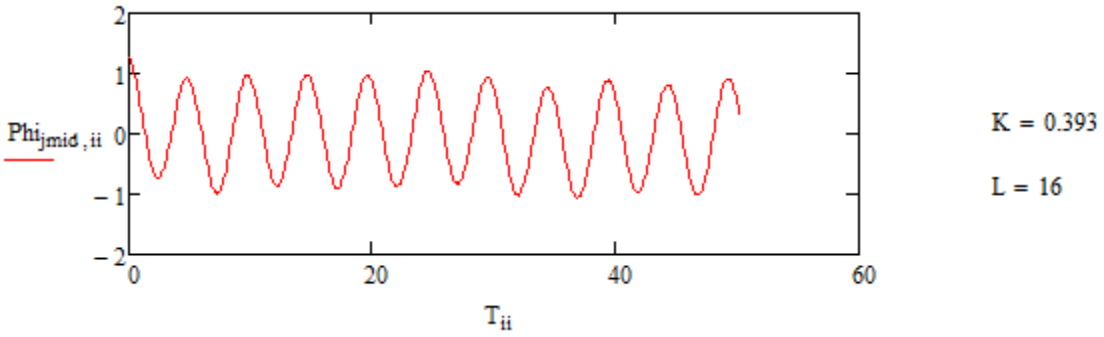




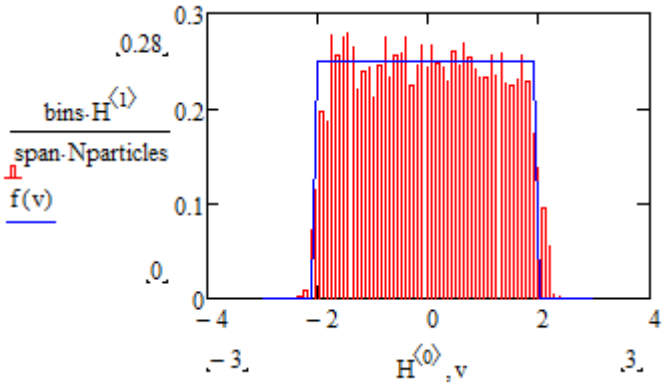
The observed field of the wave has an envelope that compares well with the expected envelope. There appears to be some loss of amplitude in the first half oscillation that is not expected.

3. What if the distribution function is a top hat?

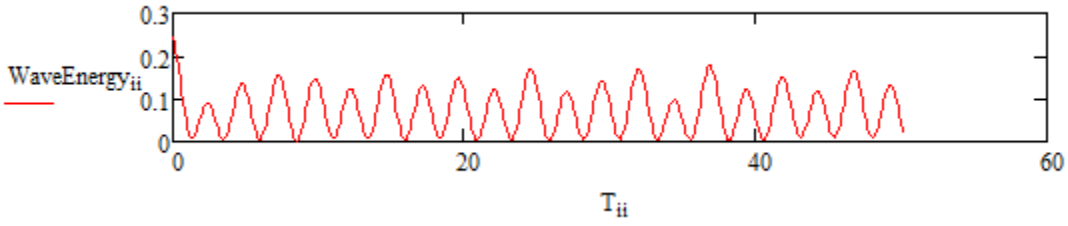
The image below is of a plot of φ(t) made using the top hat distribution. It shows that there is very little damping. Note that the number of iterations was increased so that the wave was followed for more periods.



The histogram below shows the effect of the wave on the distribution function.



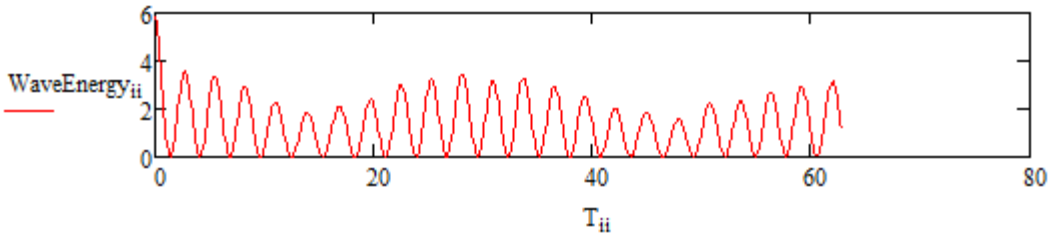
Plots of the wave and particle energy show that some energy is transferred back and forth between the wave and the particles but there is no long term trend.



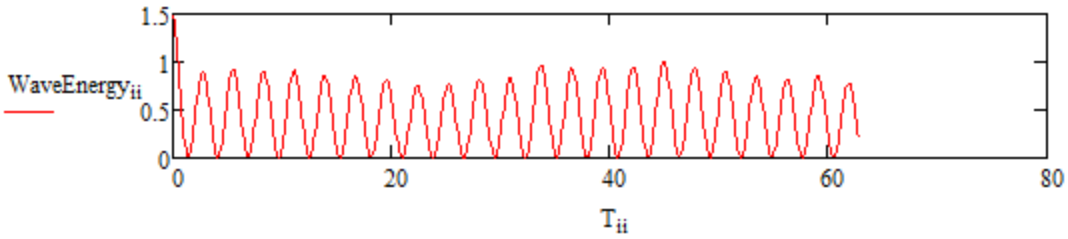
4. Can we observe echoes?

Echoes occur because the electrons oscillate in the potential wells of the waves. This oscillation occurs for electrons with velocities very near the wave phase velcocity. Faster or slower electrons pass over the tops of the potential hills and do not oscillate. The oscillation frequency is determined by the amplitude of the wave, Ewave, and the wavenumber K which together determine the "spring constant" of the electron oscillations. The period varies as the square root of Ewave. Here we compare the wave energy as a function of time for two wave amplitudes. Note that the amplitude oscillations have a shorter period for the wave of higher amplitude. The period is reduced to about 0.7 of its former value if Ewave is doubled. The effect is called an echo because the wave "reappears" after decreasing.

$Ewave = 1 \quad L = 24 \quad \omega := \sqrt{\frac{2\cdot\pi\cdot Ewave}{L}} \quad Period := \frac{2\cdot\pi}{\omega} \quad Period = 12.28$



$Ewave = 0.5 \quad L = 24 \quad \omega := \sqrt{\frac{2\cdot\pi\cdot Ewave}{L}} \quad Period := \frac{2\cdot\pi}{\omega} \quad Period = 17.366$



These plots show that the Landau damping rate should be measured for waves with small amplitude to prevent the result from being affected by the amplitude oscillations.