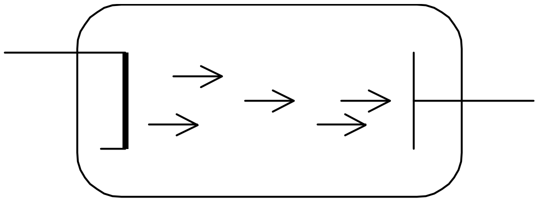
PIC code for cathode sheath

In this exercise, we will inject electrons into the space between two grounded conductors. This simulates a vacuum tube when there is a cathode emitting on the left side and a grounded anode on the right side. Also, if the right hand boundary is moved far away, the geometry corresponds to a surface in space that is emitting photoelectrons.



Vacuum diode with electrons

We will use the PIC code developed previously with some modifications:
1. We will omit ions from Poisson's equation.
2. Our solution to Poisson's equation will apply two boundary conditions. The conditions are zero potential at the left and right boundaries which are assumed to be grounded electrodes.
3. Electrons will be injected into the system at each time step. Electrons leaving the system will be omitted from calculation or recycled.

The equations to be solved

For an electron, the equations of motion are:

$$\frac{ds}{dt} = v \qquad \frac{dv}{dt} = \frac{-qE}{m}$$

The particle position is s 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

Electrons are emitted into the region from the left side. The emitted electrons can be considered to be a beam with density n_0 and mean velocity V. From the beam density we can define a plasma frequency ω_{pe} that is the inverse of a characteristic time. A characteristic distance can then be defined as V/ω_{pe} which is similar to a Debye length. These definitions are:

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

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

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

$$\frac{d\bar{s}}{d\bar{t}} = \bar{v} \qquad \frac{d\bar{v}}{d\bar{t}} = -\bar{E} \qquad \text{and} \qquad \frac{d\bar{E}}{d\bar{x}} = -\frac{n_e}{n_0} = -\bar{n}_e$$

The dimensionless scales of the electric field and potential are:

$$\hat{E} = \sqrt{\frac{n_e m V^2}{\varepsilon_0}} \qquad \hat{\phi} = \frac{mV^2}{q}$$

Equations in finite-difference form

Let k be the grid point number and i be the time step number.

The finite difference form of the equation for E is: where E_k is the field at grid point k.

$$\overline{E}_{k+1} = \overline{E}_k - \overline{n}_e Q \Delta \overline{x}$$

The equivalent line in the PIC code is:

$$E_k \leftarrow E_k - n \cdot Q \cdot \Delta x$$

Q is a variable, defined below, that specifies the charge on an electron in the PIC code.

We will convert our dimensionless equations to finite difference form.

The particle position is advanced in time using: Here the arrow indicates that the old value is replace by a new value.

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

The velocity is advanced using:

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

The computations can be shortened by eliminating V and using the variable $\Delta S = V * \Delta t$ that is the distance moved between time steps:

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

The equation advancing S becomes:

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

These 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. The error would be of higher order if we used the acceleration evaluated at the time half way between time steps, but this would require advancing the particle positions half a time step to evaluate the acceleration. If the results from the code do not change when the time step is reduced, then the time step has been made sufficiently small.

The physics that we expect to see

For a simulation domain of length L, a charge density of -1 unit creates an electric field that is L/2 units at one boundary and -L/2 units at the other boundary. Integration of -E from the center to the boundaries finds a potential that is $L^2/8$ units lower in the center than at the edge. If our simulation box is 4 units long, unit density creates a potential at the center that is -16/8 = -2 units. The electron energy at the mean velocity in dimensionless units is intially 0.5. Before the potential reaches -2, additional electrons will have been repelled by the large negative potential and some will have returned to the left boundary. If we use a narrow distribution in electron velocities, we will find that the cloud of electrons in the simulation box will have a density that alternately increases and decreases in time. These oscillation are called virtual cathode oscillations because the cloud of electrons has become an electron emitter, a virtual cathode. These oscillations have been observed in vacuum tubes and were studied in the first PIC codes (see the references).

One dimensionless parameter L to vary

The dimensionless units system is based on the beam density and mean velocity, 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.

Convert experimental units to dimensionless units

Suppose we have a circular emitter of area A emitting current I and the electrons have an average energy W. We will use an approximate value for the mean electron velocity that corresponds to the average energy.

The electron density is given by: n = J / qV where J = I / A

Some useful physical constants are: $q := 1.6\cdot 10^{-19}$ $m_e := 9.11\cdot 10^{-31}$ $\varepsilon_0 := 8.854\cdot 10^{-12}$

$I := 0.001$ Amps, for example

$\text{Length} := 3\cdot 0.0254$ $\text{Length} = 0.076$ Separation of the two electrodes, SI units

$r := 4\cdot 0.0254$ $r = 0.102$ Radius emitting area, SI units

$A := \pi\cdot r^2$ $A = 0.032$ Area of emitter

$W_{eV} := 1.0$ Average electron energy in eV

$\text{Vemission} := \sqrt{\frac{2\cdot q\cdot W_{eV}}{m_e}}$ $\text{Vemission} = 5.927\times 10^5$ m/s Velocity of emission

$n_e := \frac{I}{A\cdot q\cdot \text{Vemission}}$ $n_e = 3.252\times 10^{11}$ m^{-3} Electron density

Scales to make time and distance dimensionless:

$\hat{t} = \sqrt{\frac{\varepsilon_0 m}{n_0 q^2}}$ $\text{t\_hat} := \sqrt{\frac{\varepsilon_0\cdot m_e}{n_e\cdot q^2}}$ $\text{t\_hat} = 3.113\times 10^{-8}$ seconds

$\hat{x} = \sqrt{\frac{\varepsilon_0 m V^2}{n_0 q}}$ $\text{x\_hat} := \sqrt{\frac{\varepsilon_0\cdot m_e\cdot \text{Vemission}^2}{n_e\cdot q^2}}$ $\text{x\_hat} = 0.018$ meters

$\text{Ldimensionless} := \frac{\text{Length}}{\text{x\_hat}}$ $\text{Ldimensionless} = 4.13$

We will enter the dimensionless value of Ldimensionless above into the line below, rounded to the nearest integer value:

$L := 4$ We specify L, the width of the simulation domain in dimensionless units.

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.

$\Delta x := 0.20$ We specify the spatial resolution.

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

$jmax := \frac{L}{\Delta x}$ $jmax = 20$ This is the number of x grid points we need for the desired spatial resolution.

We will define a grid with jmax points separated by distance Δx defined above:

$j := 0 .. jmax$ $x_j := \Delta x \cdot j$ $x_{jmax} = 4$

Mean particle velocity

$V := 1$ This is the initial particle velocity that is required by our dimensionless units. We will also define a velocity distribution f(v) to use instead of a fixed velocty.

Time step

The characterisitic velocity of 1 and the grid spacing Δx = 0.2 results in a 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 may use a distribution in velocites, and in this case the fastest particle could cross several cells in a time step. This would introduce error because the value of E used in the iteration procedure is the value at the current particle position at the current time.

$\Delta t := 0.5 \cdot \frac{\Delta x}{V}$ $\Delta t = 0.1$

Particle number per cell, particle charge, and particle creation rate

One electron per unit length corresponds to a charge density of -1 in dimensionless units which is $-n_0q$ in real units. In other words, one electron in the simulation has charge $-n_0q$. One particle in a cell, however, results in too much noise because the number density would take on discrete values such as 0, 1, 2 etc. as the particles move from cell to cell. We will reduce the noise by increasing the number of particles and reducing the charge on a particle. With 50 electrons in a cell, the random noise will be about 7 electrons (from the square root of 50) or about 14%. To maintain a beam of unit charge density, with 50 particles in a cell of length Δx = 0.2 we must have 1 = 50*Q/Δx, or Q = Δx/50 = 0.004 on each particle.

$\text{Ncell} := 50$ The number of electrons in a cell.
Ncell determines the level noise in the charge density calculation.

$Q := \frac{\Delta x}{\text{Ncell}}$ $Q = 0.004$ The charge on one electron required to have a beam of unit density.

$\text{dNdt} := \frac{1}{Q}$ $\text{dNdt} = 250$ The number of electrons to inject per unit time for unit charge density

$\text{Ninject} := \text{floor}(\text{dNdt}\cdot\Delta t)$ $\text{Ninject} = 25$ The number of particles to inject in one time step, rounded to be an integer.

$\text{dNdt} := \frac{\text{Ninject}}{\Delta t}$ $Q := \frac{1}{\text{dNdt}}$ $Q = 0.004$ Q adjusted to be consistent with the rounded value of Ninject.

Initial electron positions to test the E field finder

In order to test our programs to find the charge density, electric field, and the potential, we will create a trial set of electron positions that puts the beam inside of the simulation domain.

$\text{kmax} := \text{jmax}\cdot\text{Ncell}$

$\text{kmax} = 1000$

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

We define a uniformly spaced set of electron positions with locations s_k: $k := 0 .. \text{kmax} - 1$ $s_k := \left(\frac{k + 0.5}{\text{kmax}}\right)\cdot L$

The locations s_k are defined so that the electrons are uniformly spaced between the boundaries and the zeroth and last electrons are not on the boundaries. $s_0 = 0.002$ $s_{\text{kmax}-1} = 3.998$

Charge density calculation

The first step in finding the electric field E is to find the particle density. The program loop below determines electron density by counting electrons between grid points. Particles outside the box are not counted. If the particle is 10% of the way between a first and second grid point, 90% of the charge is assigned to the first grid point and 10% is assigned to the second. This 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. This program loop was explained in more detail in the previous PIC code exercise.

```
NumberDensity(s) :=
    $N_{jmax} \leftarrow 0$
    for  $k \in 0 .. \text{rows}(s) - 1$
        continue   if  $s_k \le 0 \vee s_k \ge L$
        $\text{jlower} \leftarrow \text{floor}\left(\frac{s_k}{\Delta x}\right)$
        $N_{jlower+1} \leftarrow \left(\frac{s_k - x_{jlower}}{\Delta x}\right) + N_{jlower+1}$
        $N_{jlower} \leftarrow \left(\frac{x_{jlower+1} - s_k}{\Delta x}\right) + N_{jlower}$
    $N_0 \leftarrow 2 \cdot N_0$
    $N_{jmax} \leftarrow 2 \cdot N_{jmax}$
    $n \leftarrow \frac{N}{\Delta x}$
```

Note that the **continue** statement omits the counting of particles on or outside of the boundaries. The number density in the first and last cells is doubled because the volume contributing is only on one side of the grid points, not both sides.

This line calls the NumberDensity function.

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

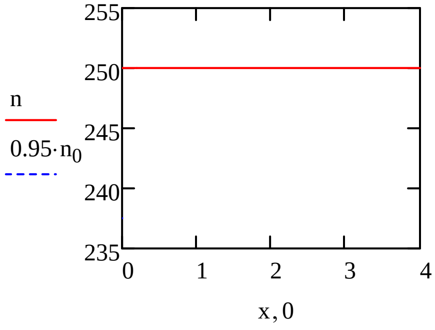
The average number density n0 is the total number of particles kmax divided by the length of the simulation domain L. Recall that the charge density n0*Q is unity.

$$n0 := \frac{kmax}{L} \qquad Q \cdot n0 = 1$$

If n0 is not an integer, for example if the average number of particles in a cell is 50.5, then some cells will have 50 particles and some will have 51, hence a plot of n_k will not necessarily show a straight line.

The plot of number density includes the point ($0.95n_0$,0) in order to have the graph show a reasonable range of y values.

Plot of number density



Electric field at grid points with grounded boundaries

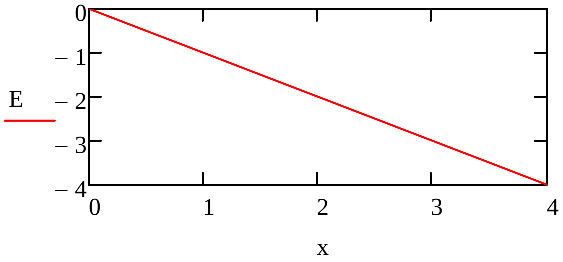
Recall that the electric field is found using: $E_k \leftarrow E_k - n \cdot Q \cdot \Delta x$

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

```
E := | E_jmax ← 0
     | for  j ∈ 1 .. jmax
     |   E_j ← E_(j−1) − 0.5·(n_j + n_(j−1))·Q·Δx
     | E
```

The average density in a cell is obtained by averaging the values at the adjacent grid points.

Plot of trial solution for E at grid points



This is called a trial solution because there is a constant of integration that has not been evaluated.

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 linear interpolation.

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

The solution for φ in finite difference form is:

$$\phi_j = \phi_{j-1} - \left(\frac{E_j + E_{j-1}}{2}\right)\Delta x$$

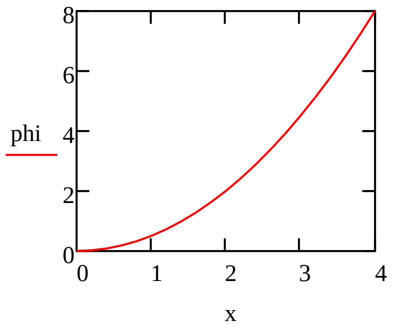
This program loop creates an array with the φ values at the grid points:

```
φ(E) := | $\varphi_{jmax} \leftarrow 0$
        | for  $j \in 1 .. jmax$
        |    $\varphi_j \leftarrow \varphi_{j-1} - 0.5\cdot(E_j + E_{j-1})\cdot\Delta x$
        | $\varphi$
```

This line calls the function: $phi := \varphi(E)$

A plot of the potential shows a parabolic profile which is one possible solution for a uniform charge density. We can add any potential that varies linearly, such as φ = ax +b where a and b are coefficients, and still have a valid solution. The boundary condition of φ = 0 at the left boundary indicates that we should assign b = 0. The coefficient a is found by dividing the potential at the right boundary by L. The linear function ax is then subtracted from the initial result for φ to obtain the final result that satisfies both boundary conditions.

Plot of a first guess for φ

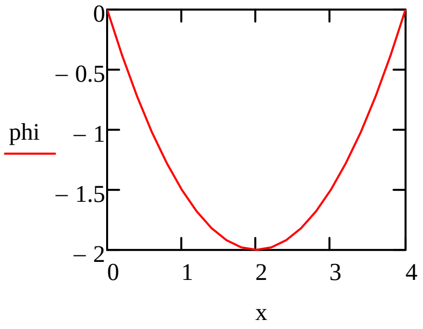


$$a := \frac{\varphi(E)_{jmax}}{L}$$

This is the constant electric field (constant of integration) that shoud be subtracted from our trial solution for E.

This line "fixes" φ so that it satisfies the boundary condition: $phi := phi - a\cdot x$

Plot of the corrected φ



The plot at right shows a parabolic depression in the potential that is what we expect for a uniform cloud of electrons.

The depth of the depression is -L^2/8 which is what we calculated at the outset.

$$\frac{-L^2}{8} = -2 \qquad \min(phi) = -2$$

The minimum value of phi is the expected value.

Add a bias potential φbias on the right hand boundary

In an experiment, the right hand boundary may be biased relative to the left hand boundary. To include this possibility, we add a constant of integration to E that is defined as φbias/L.

$\varphi\text{bias} := 1$

This is a test value for φbias we use to determine that the Efield function is working correctly.

This program loop finds the E that satisfies the boundary conditions with an applied bias. The function Efield(n,Q) returns a vector containing E. We will use this function in our PIC code. The function also has argument Q because Q may be changed later in this exercise. If we did not make Efield a function of Q, then the Q value used would be the value at the place in the program where Efield was defined.

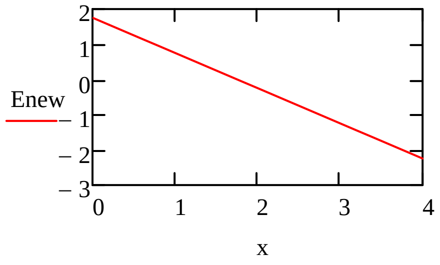
```
$\text{Efield}(n, Q, \varphi\text{bias}) :=$
    $E_0 \leftarrow 0$
    for $j \in 1 .. \text{jmax}$
        $E_j \leftarrow E_{j-1} - 0.5 \cdot (n_j + n_{j-1}) \cdot Q \cdot \Delta x$
    $\varphi \leftarrow -0.5 \cdot \Delta x \sum_{j=1}^{\text{jmax}} (E_{j-1} + E_j)$
    $E \leftarrow E + \frac{\varphi - \varphi\text{bias}}{L}$
```

$\text{Enew} := \text{Efield}(n, Q, \varphi\text{bias})$

$\text{PhiNew} := \varphi(\text{Enew})$

The plot at right shows that the new electric field is not symmetric about the midplane as a result of φbias.

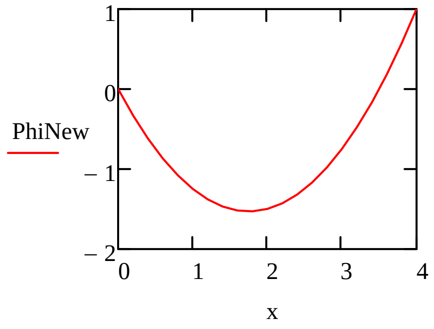
Plot of the electric field



Now we will check that the potential on the left boundary is zero and the potential on the right boundary is φbias:

Next we will change φbias to zero for further testing of the PIC code:

Plot of φ(x) with bias



$\varphi\text{bias} := 0$

This value may be changed to alter the results of the PIC code.

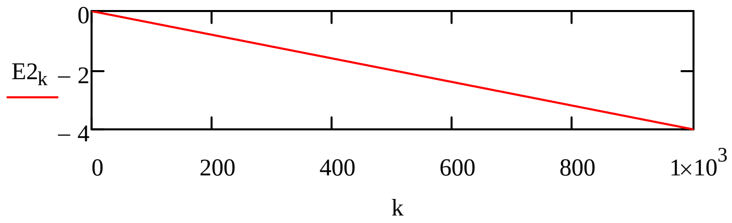
Electric field at particle locations

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 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**.

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

$E2_k := \text{interp}\left(\text{coeffs}, x, E, s_k\right)$ This interpolation routine finds the electric field at the particle locations.

E at particle locations



The plot shows the the interpolator is working correctly.

Number of iterations

$\text{Ncrossings} := 3$

We will follow the electrons for sufficient time for the beam to have crossed the simulation box Ncrossings times, with Ncrossings = 3. The number of iterations we need is then:

$$\text{iters} := \frac{\text{Ncrossings}\cdot L}{V\cdot\Delta t}$$

$\text{iters} = 120$

Recycling of electrons

The electrons will be "parked" at S = 0 until they are launched. The parked electrons will not contribute to the electric field until they enter the system.

An electron that has reached the end of the simulation box can be injected and used again. This has the advantage that the subscripted variable S_k, which contains the electron positions, does not need to have a dimension sufficient for all the particles that are injected. The program FindNext below searches the locations of the particles, starting with k = 0, and finds the first one that is parked at zero. This value of k is then used when the next electron is injected.

```
FindNext(S) := | k ← 0
               | while  $S_k > 0 \wedge S_k < L$
               |    k ← k + 1
               | k
```

Choosing electrons from a distribution function by the rejection method

The program Vselect(b) will select an initial velocity for each new electron. The velocity is selected randomly from a specified distribution. The distribution we will use is the distribution of particles that would escape into space through a hole in a space station (for example) that had a Maxwellian distribution of particles inside. The Maxwellian is weighted by an extra factor of v because the flux through a hole is proportional to v.

$vt := \frac{2}{\sqrt{\pi}}$ vt has the role of a thermal velocity. Increasing vt increases the average velocity of the emitted electrons. This particular value of vt gives an average velocity of 1.

This is the distribution function we select from. $f(v) := \frac{\pi}{2} \cdot v \cdot e^{-\left(\frac{v}{vt}\right)^2}$

$\int_0^\infty f(v)\,dv = 1$ This verifies that our distribution is normalized to unity.

$Vaverage := \frac{\int_0^\infty v \cdot f(v)\,dv}{\int_0^\infty f(v)\,dv}$ $Vaverage = 1$ This verifies that the average velocity is unity.

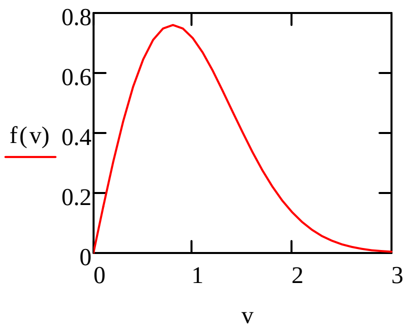
$v := 0, 0.1 .. 3$ A range of v values for a plot.

There is no need to create particles with v > 3 because f(v) gets small.

This program loop chooses a v randomly on the interval 0 to 3 and chooses a test value from 0 to 1. 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 being kept is proportional to f(v), the selected values of v have a probability given by f(v). This procedure is the "rejection method."

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" to override Vselect being evaluated only once.

Plot of the electron distribution function



```
Vselect(b) :=  v ← 0
               y ← 1
               while  y > f(v)
                   v ← 3·rnd(1)
                   y ← rnd(1)
               v
```

Test of the function Vselect for f(v)

We will call the function Vselect(b) many times, put the v values in a vector Vtest, and h**istogram** the values. Note that b can be any number.

```
Vtest := | hmax ← 5000
         | $G_{hmax}$ ← 0
         | for  h ∈ 0 .. hmax
         |   $G_h$ ← Vselect(2)
         | G
```

To scale the histogram we need:

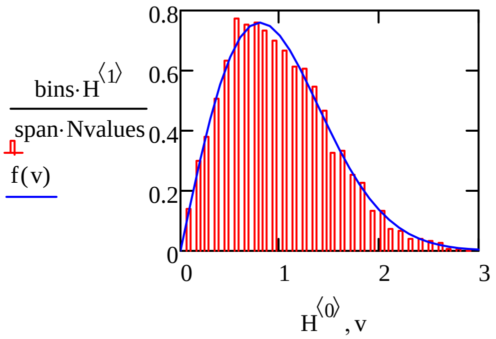
$bins := 30$

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

$Nvalues := rows(Vtest)$

$H := histogram(bins, Vtest)$

Our program Vtest is working correctly because the scaled histogram of the velocities agrees with f(v). The scale factor is found from the span of values, the number of bins, and the number of values being histogrammed.



Launching multiple particles per time step

Recall the we want to inject Ninject electron per time step. Electrons injected on the previous time step will have moved a mean distance V*Δt leaving this region relatively free of particles. Hence we will inject the new particles evenly spaced between 0 and V*Δt. Note that V is the mean velocity and v is the velocity of the velocity of the injected particle.

The program fragment at right launches Ninject electrons on each iteration. The factor (kk/Ninject) varies from just above zero to unity as the loop is executed, and this places the electrons at evenly spaced positions.

```
for  kk ∈ 1 .. Ninject
  | kinject ← FindNext(S)
  | v ← Vselect(2)
  | $S_{kinject} \leftarrow V \cdot \Delta t \cdot \frac{kk}{Ninject}$
  | $\Delta S_{kinject} \leftarrow v \cdot \Delta t$
```

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.

The program loop

We wll use the timer function to measure the elapsed time: $\text{StartTime} := \text{time}(2)$

```
$M :=$
  $E_{jmax} \leftarrow 0$
  $M_{kmax-1,\,iters} \leftarrow 0$
  $\Delta S_{kmax-1} \leftarrow 0$
  $S_{kmax-1} \leftarrow 0$
  for $i \in 0 .. iters$
    for $kk \in 1 .. Ninject$
      $kinject \leftarrow \text{FindNext}(S)$
      $v \leftarrow \text{Vselect}(2)$
      $S_{kinject} \leftarrow V \cdot \Delta t \cdot kk \div Ninject$
      $\Delta S_{kinject} \leftarrow v \cdot \Delta t$
    $n \leftarrow \text{NumberDensity}(S)$
    $E \leftarrow \text{Efield}(n, Q, \varphi bias)$
    $vs \leftarrow \text{pspline}(x, E)$
    for $k \in 0 .. kmax - 1$
      continue  if $S_k \le 0 \vee S_k \ge L$
      $\Delta S_k \leftarrow \Delta S_k - \Delta t \Delta t \cdot \text{interp}(vs, x, E, S_k)$
      $S_k \leftarrow S_k + \Delta S_k$
      $S_k \leftarrow 0$  if $S_k \le 0 \vee S_k \ge L$
    $M^{\langle i \rangle} \leftarrow S$
  $M$
```

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

These lines inject new electrons.

These line advance the particle positions S_k.

The **continue** statement omits from calculation the parked electrons.
If S is outside the box, it is moved to the boundary at 0 so that plots are nicer.

$\text{EndTime} := \text{time}(2)$

The time to run the loop is: $\text{EndTime} - \text{StartTime} = 0.578$ seconds

Note that the **time( )** function needs a "dummy" argument. This is because **time( )** must be recognized as a function in order to be evaluated every time it is called. This "work around" was used for our function **Vselect( )** that uses the random number generator that we want to call each time we need a velocity.

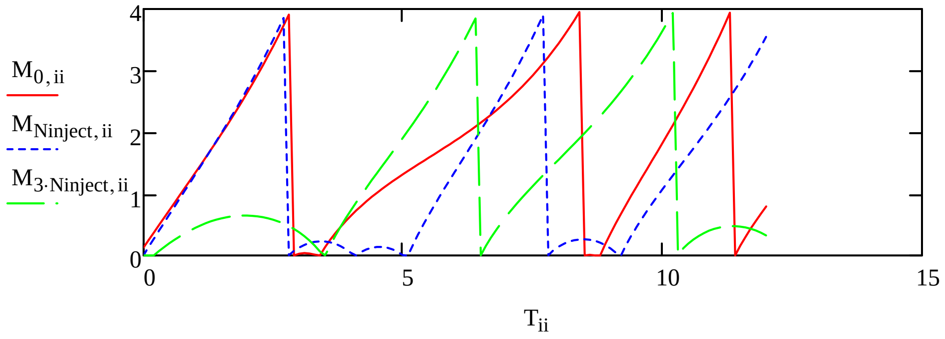
The answer matrix M has columns that are the positions of the kmax particles.

$\mathrm{Mcols} := \mathrm{cols}(M)$ $\mathrm{Mcols} = 121$ This is the number of time steps in M.

$\mathrm{Mrows} := \mathrm{rows}(M)$ $\mathrm{Mrows} = 1000$ This is the maximum number of particles in M.

This line defines the time T for each time step that is useful for plots. $ii := 0 .. iters$ $T_{ii} := ii \cdot \Delta t$

Plot of several electron trajectories showing recycling



How many electrons are in the simulation domain at any time?

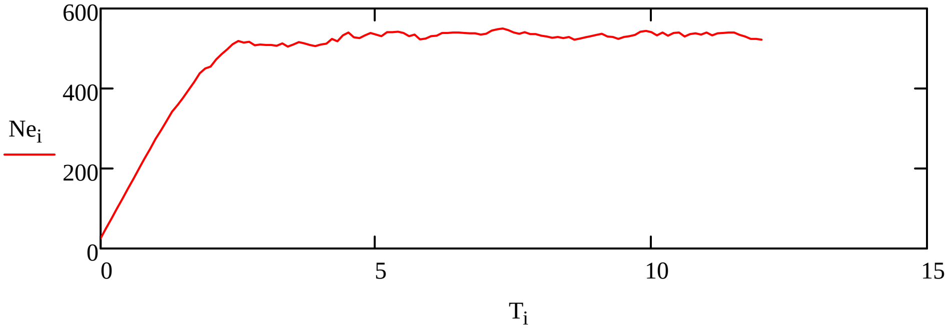
The program loop below looks at all entries in the matrix M (that will be substituted for the generic matrix X) and counts the particles that are in the simulation domain at each time step.

```
Nelectrons(X) :=  | ncols ← cols(X) − 1
                  | nrows ← rows(X) − 1
                  | $counts_{ncols}$ ← 0
                  | for  i ∈ 0 .. cols(X) − 1
                  |   for  k ∈ 0 .. nrows
                  |     $counts_i$ ← $counts_i$ + 1  if  $X_{k,i} > 0 \wedge X_{k,i} < L$
                  | counts
```

The function Nelectrons is called to find the number of electrons that are active at each of the sampled time steps.

$Ne := \mathrm{Nelectrons}(M)$ $i := 0 .. \mathrm{rows}(Ne) - 1$ An index for the plot below.

Plot of the number of electrons as a function of time



Reduce the size of the matrix of answers

The maximum number of electrons in the box at one time was:

$\text{Nmax} := \max(\text{Ne})$

$\text{Nmax} = 550$

Inspection of M shows that the rows in the bottom of the matrix are not used because the dimension of S was guessed using a number that was too large (unit density filling the box). The program below starts with the bottom row of M, finds the largest S value on the row, and moves upward if the largest value is zero indicating that the row was not used. In this way the last used row, klast, is found. Then the matrix is made smaller using the submatrix command so that the last row is klast.

```
klast := | MT ← M^T
         | for  k ∈ kmax − 1 .. 0
         |    (break)  if  max(MT^⟨k⟩) ≠ 0
         | k
```

$\text{rows}(M) = 1000$ This is the number of rows that were put in M.

$\text{klast} = 574$ This is the number of rows actually used.

$\max\left[\left(M^T\right)^{\langle \text{klast} \rangle}\right] = 0.147$ $\max\left[\left(M^T\right)^{\langle \text{klast}+1 \rangle}\right] = 0$ This shows that the row klast contained data and the row klast + 1 did not.

$M := \text{submatrix}(M, 0, \text{klast}, 0, \text{iters})$ Now we shorten the matrix M.

$\text{rows}(M) = 575$ This line shows that it is shorter.

Function to further reduce the size of the matrix of answers

We will not want to plot the position of every particle, hence we will create a matrix Msmaller that has fewer rows.

$Nfewer := 12$ The number of particles to be plotted is reduced by a factor of Nfewer.

SmallerMatrix is a function that can be used to reduce the number of rows of any matrix. The program first transposes the matrix so that rows have become columns, then reduces the number of columns, then transposes the reduced matrix. Mathcas does not have a command to select rows.

```
$SmallerMatrix(X, Nfewer) :=$
    $XT \leftarrow X^{T}$
    $NrowsSmall \leftarrow floor\left(\frac{cols(XT) - 1}{Nfewer}\right)$
    for $i \in 0 .. NrowsSmall$
        $XTsmaller^{\langle i \rangle} \leftarrow XT^{\langle Nfewer \cdot i \rangle}$
    $Xsmaller \leftarrow XTsmaller^{T}$
    $Xsmaller$
```

XT is the transpose of matrix X.

This command reduces the size of our answer matrix: $Msmall := SmallerMatrix(M, Nfewer)$

A matrix that has 100 rows and columns usually makes a nice plot: $rows(Msmall) = 48$ $cols(Msmall) = 121$

Particle positions for plotting

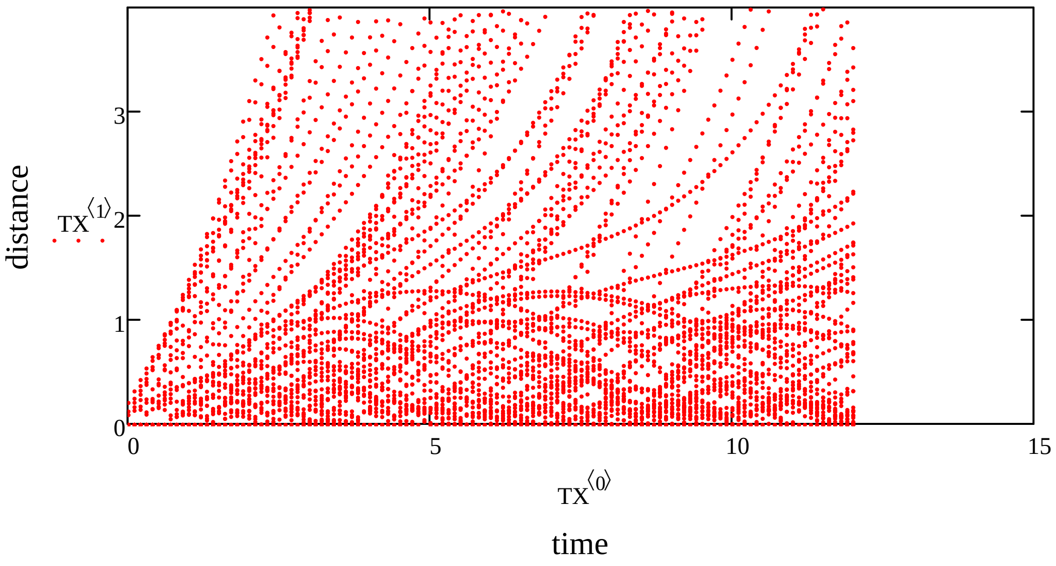
We will make a plot of points that are the particle positions as a function of time. To do this, we will create a matrix consisting of two columns. The first column will contain the time and the second column will contain the position. We will take the rows of the matrix Msmall (which contain particle positions) and **stack** them one on top of the other to get a long list of particle positions. These positions will be the second column of a matrix. The first column will contain the corresponding times. These are also obtained by stacking. To read the rows of the matrix, we will first transpose the matrix so that we can use the Mathcad function that selects columns. The **augment** command places the X positions in the column to the right of the list of times.

```
$Points(X) :=$
    $XT \leftarrow X^{T}$
    $Points \leftarrow augment\left(T, XT^{\langle 0 \rangle}\right)$
    for $i \in 1 .. cols(XT) - 1$
        $Points \leftarrow stack\left(Points, augment\left(T, XT^{\langle i \rangle}\right)\right)$
    $Points$
```

$TX := \text{Points}(\text{Msmall})$ The points to plot are created from the smaller matrix Msmall.

$\text{rows}(TX) = 5808$ These are the number of points to be plotted.

Plot of particle positions as a function of time



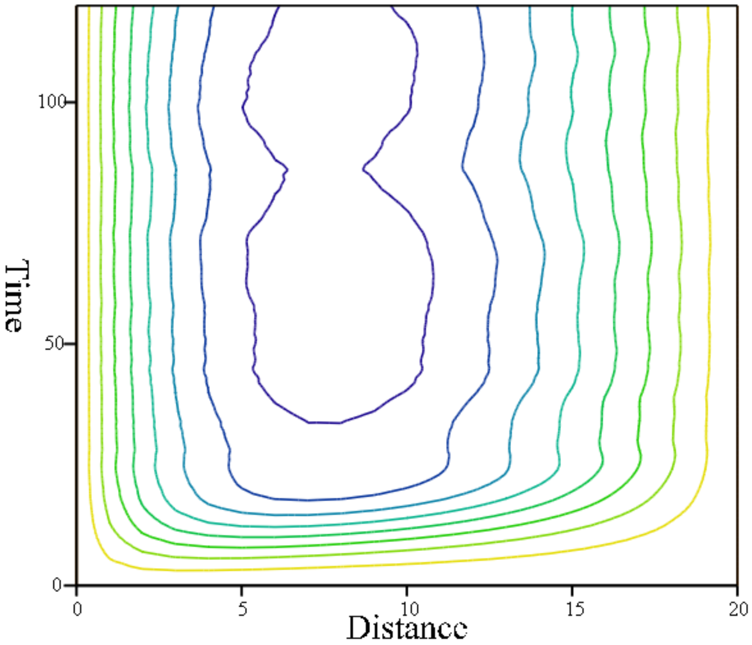
The plot shows that the first electron crosses the simulation domain and that later electrons are repelled by the electrons in the box and return to the starting position. The electron cloud seems to have an oscillating boundary and some electrons return to x = 0 after being launched. These oscillations are the virtual cathode oscillations.
The plot is a plot of points, but the points appear to the eye as particle trajectories because of the density of points. The sampling parameter **Nfewer** will adjust the number of points plotted. About 500 time points will create continuous lines from points.

Program for the potential as a function of time

This program loop takes the full matrix M and calculates the potential at the grid points at each time step. The columns of Phi contain the values of E at grid points.

```
Phi :=  $Phi_{jmax, cols(M)-1} \leftarrow 0$
        for  $i \in 0 .. cols(M) - 1$
            $s \leftarrow M^{\langle i \rangle}$
            $n \leftarrow NumberDensity(s)$
            $E \leftarrow Efield(n, Q, \varphi bias)$
            $Phi^{\langle i \rangle} \leftarrow \varphi(E)$
        Phi
```

Contour plot of φ(x) as a function of time

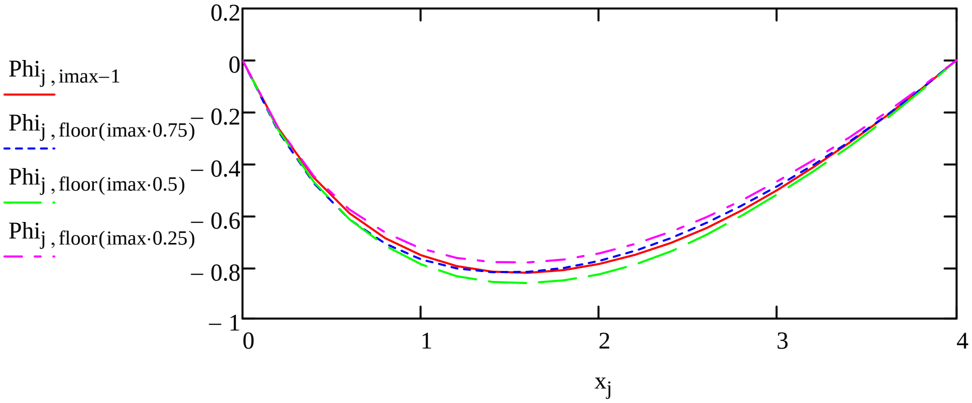


Phi

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

$imax := cols(Phi)$

Plot of the potential at several times



Improved line plot

When particles are recycled, they drop from the top of the plot x = L to the bottom x = 0 which creates unwanted lines in a line plot. The following program takes the list of points TX and adds new points so that a particle that drops from the top of the box to the bottom is first moved to the top right hand boundary so that the drop is at the end of the graph, not in the middle somewhere. Similarly, if a trajectory terminates at the right boundary, points are added so that the particle moves downward along the right boundary then to the left along the bottom boundary to the starting position of the next particle.

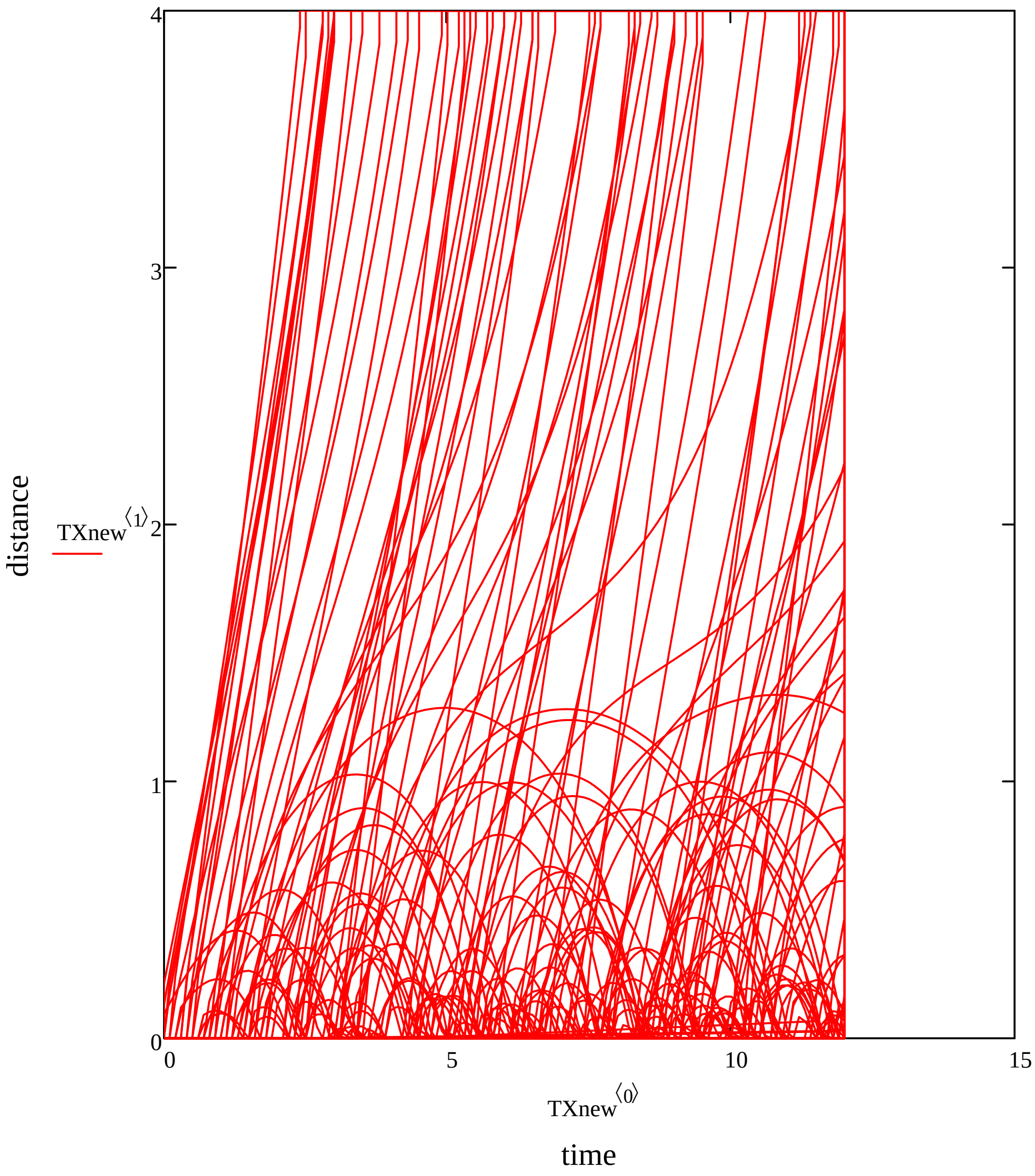
```
$TXnew :=$
    $Tend \leftarrow iters \cdot \Delta t$
    $TXnew_{rows(TX),1} \leftarrow 0$
    $k \leftarrow -1$
    for $j \in 0 .. rows(TX) - 2$
        continue if $\left(TX_{j+1,1} = 0 \wedge TX_{j,1} = 0\right)$
        $k \leftarrow k + 1$
        $TXnew_{k,0} \leftarrow TX_{j,0}$
        $TXnew_{k,1} \leftarrow TX_{j,1}$
        if $\left(TX_{j,1} - TX_{j+1,1}\right) > 0.9 \cdot L \vee TX_{j,0} = Tend$
            $TXnew_{k+1,0} \leftarrow TX_{j,0}$
            $TXnew_{k+1,1} \leftarrow L$
            $TXnew_{k+2,0} \leftarrow Tend$
            $TXnew_{k+2,1} \leftarrow L$
            $TXnew_{k+3,0} \leftarrow Tend$
            $TXnew_{k+3,1} \leftarrow 0$
            $TXnew_{k+4,0} \leftarrow TX_{j+1,0}$
            $TXnew_{k+4,1} \leftarrow 0$
            $k \leftarrow k + 4$
    $TXnew \leftarrow submatrix(TXnew, 0, k, 0, 1)$
    $TXnew$
```

The submatrix command at the end of the program loop is used to delete the zeroes in the list of points that arise from the dimension of S being more than the number of particles used in the program.

$rows(TXnew) = 5541$

The particle positions to be plotted. Note that the variable Nfewer can be used to alter the number of lines in the plot.

Plot of particle positions as a function of time



References :

"Space-Charge Instabilities in Electron Diodes and Plasma Converters," C. K. Birdsall and W. B. Bridges, Journal of Applied Physics 32(12), pp. 2611-2618, December 1961.

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