Electrostatic Plasma Simulation: Ion Clump

APAM1601
Columbia University

Introduction

References:

Berkeley Plasma Simulation Group
Prof. Charles "Ned" Birdsall
Plasma Physics Via Computer Simulation, by Birdsall and Langdon, (Institute of Physics, 1991)

This notebook calculates a (very small) "particle-in-cell" (PIC) simulation of a one-dimensional plasma. Electrons and ions interact collectively by way of electrostatic forces. This is a very demanding computation. PIC simulations are usually performed on large supercomputers and require many hours (or days) of computation. Current research using these types of plasma codes are 3D, electromagnetic, and defined with complicated geometries.

Equations

The equations of motion for the ith particle are

$\partial_t\, x_i = v_i$

$\partial_t\, v_i = (q_i/m_i)\, E(x_i,\ t)$

The equation for the electric field

$E(x, t) = -\partial_x\, \Phi(x,\ t)$

$\partial_{x,x}\, \Phi = -\rho(x, t)\ /\epsilon_0$

where ρ is the total charge density, including both the electrons and the ions.

Physics

A plasma is characterized by a *length* scale called the "Debye length", a *time* scale by the (inverse) "plasma frequency", and a *velocity* scale associated with the "thermal velocity".

In MKS units,

$\omega_p\ =\ \sqrt{q_e^2\, n/\epsilon_0\, m_e}$, proportional to square-root of the density

$v_{\text{the}}\ =\ \sqrt{kT_e/m_e}$, proportional to the square-root of the temperature

$\lambda_D = v_{\text{the}}/\omega_p$, the ratio of the thermal speed to the plasma frequency

In order for us to properly simulate a plasma, we need a "box" much larger than $\lambda_D$and a simulation that lasts much longer than ω_p^{-1}. We also require the numerical time step, Δt, to be much *shorter* than ω_p^{-1}.

Numerics

For simplicity, we'll make the following assignments:

- $\Delta x = \Delta t = 1$ (the spatial and temporal grid points are integers)
- $(q_e/m_e) = -1$ for electrons
- $(q_i/m_i) = 1/25$ for ions. (The ions are *actually* much heavier!)

With $\Delta x = 1$, the total length of our "box" of particles must be $L >> 1$.

With $\Delta t = 1$ and with $q_e/m_e = 1$, then $\Delta t\, \omega_p = \sqrt{q_e\, n/\epsilon_0} << 1$. If we have 2000 electrons, then $n = 10^3/L \sim 1$. Thus, we need to set the parameter $q_e/\epsilon_0 << 1$.

With $v_{\text{the}} = \lambda_D/\omega_p$, then $v_{\text{the}} \sim 1$.

Basic Definitions

```
In[•]:= SetDirectory[NotebookDirectory[]]
Out[•]= /Users/mauel/Library/CloudStorage/Dropbox/Work/Courses/Plasma I/Plasma 1 (2023)

In[•]:= boxLength = 500; (* Length of plasma and number of grid points *)

In[•]:= numElectrons = 2000; (* also number of ions *)

In[•]:= qOverϵ = 4.0*^-4; (* qe/ϵ0 << 1 *)

In[•]:= ionMass = 25.0; (* me/mi = 1/25 << 1 *)

In[•]:= electronTemp = 0.25; (* kTe; vthe = √kTe *)

In[•]:= ionTemp = 0.1; (* kTi; vthi = √kTi/mi *)

In[•]:= Print["Average Density = ", nAvg = numElectrons / boxLength];
        Print["Electron Thermal Velocity = ", vthe = Sqrt[electronTemp]];
        Print["Ion Thermal Velocity = ", vthi = Sqrt[ionTemp / ionMass]];
        Print["Plasma Frequency = ", ωp = Sqrt[(numElectrons / boxLength) qOverϵ]];
        Print["Debye Length = ", λD = vthe / ωp];
        Print["Number of particles per cell = ", nAvg];
        Print["Number of Debye lengths in box = ", boxLength / λD];
        Print["Number of time steps per plasma oscillation = ", 1 / ωp];
```

```
Average Density = 4
Electron Thermal Velocity = 0.5
Ion Thermal Velocity = 0.0632456
Plasma Frequency = 0.04
Debye Length = 12.5
Number of particles per cell = 4
Number of Debye lengths in box = 40.
Number of time steps per plasma oscillation = 25.
```

Electrons

Each particle is represented by a coordinate, $\{x_i, v_i\}$.

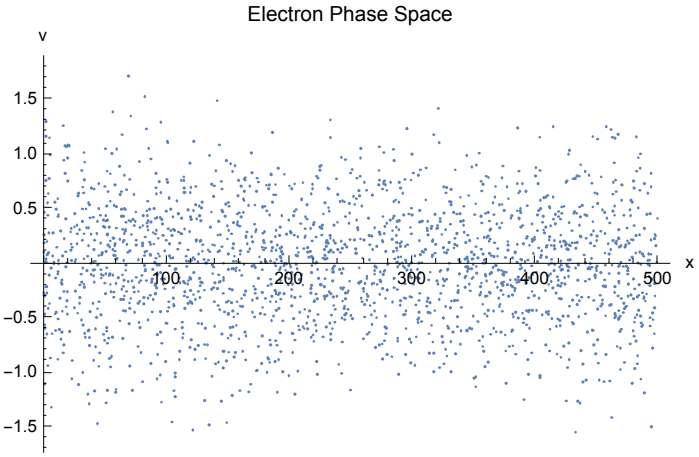
In[•]:=
```
electrons = Table[{RandomReal[{0, boxLength}],
    RandomVariate[NormalDistribution[0.0, vthe]]}, {numElectrons}];
```

In[•]:=
```
Dimensions[electrons]
```

Out[•]=
```
{2000, 2}
```

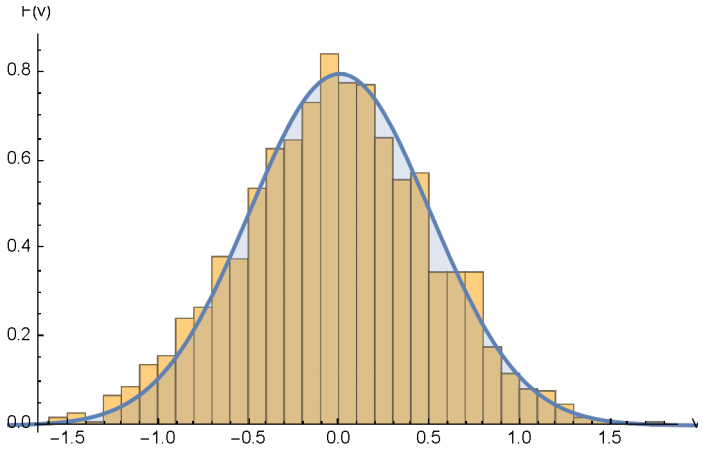
In[•]:=
```
ListPlot[electrons, AxesLabel → {"x", "v"}, PlotLabel → "Electron Phase Space"]
```

Out[•]=



In[•]:=
```
Plot[PDF[NormalDistribution[0, vthe], x], {x, -2, 2}, Filling → Axis];
Histogram[electrons[[All, 2]], Automatic, "PDF"];
pltElectDistribution = Show[%, %%, AxesLabel → {"v", "F(v)"}]
```

Out[•]=



The electron charge density is evaluated on the *grid points*. We use "triangular weighting" to map the electron's position (in-between the grid points) to the charge density at each point.

In[•]:=
```
rho0 = Table[0.0, {boxLength + 1}]; (* including 0...boxLength *)
rhoT = rho0;
```

In[•]:=
```
addρ[x_Real] := Module[{loc},
  loc = Floor[x];
  rhoT[[loc + 1]] += (loc + 1 - x);
  rhoT[[loc + 2]] += (x - loc);]
```

In[•]:=
```
getρ[e_List] := Module[{},
  rhoT = rho0;
  addρ /@ First[Transpose[e]];
  rhoT[[1]] += rhoT[[boxLength + 1]];
  Drop[rhoT, -1]]
```

In[•]:=
```
rhoElectron = getρ[electrons]; // Timing
```

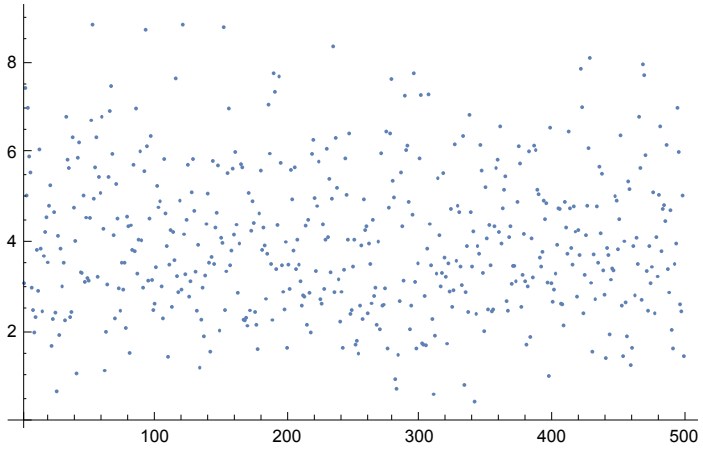
Out[•]=
```
{0.016681, Null}
```

In[•]:=
```
Plus @@ rhoElectron (* How many electrons? *)
```

Out[•]=
```
2000.
```

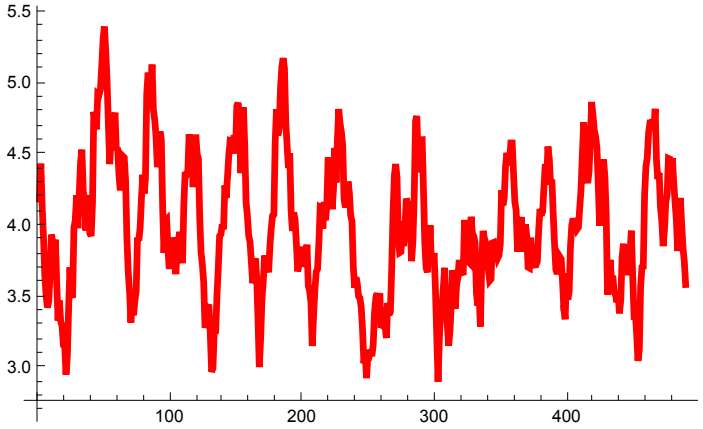
In[ ]:=
```
pr1 = ListPlot[rhoElectron]
```

Out[ ]=



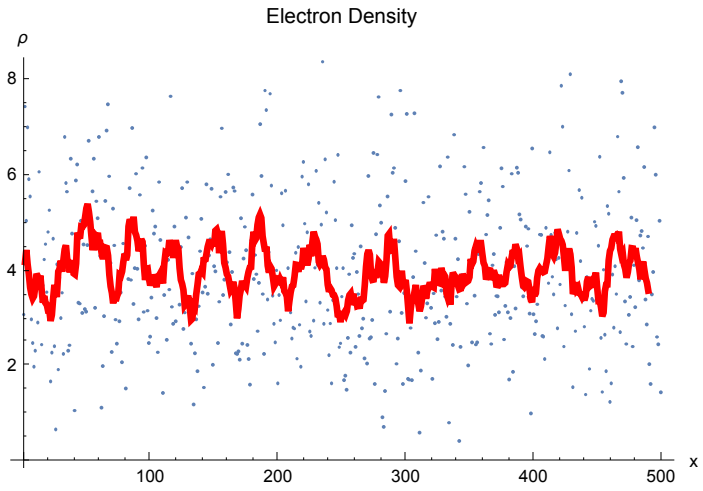
In[ ]:=
```
pr2 = ListPlot[MovingAverage[rhoElectron, Floor[λD]],
  Joined → True, PlotStyle → {Red, Thickness[0.01]}]
```

Out[ ]=



In[•]:=
```
Show[pr1, pr2, PlotLabel → "Electron Density",
 AxesLabel → {"x", "ρ"}, PlotRange → {0, 2 nAvg}]
```

Out[•]=



Notice large fluctuations in the electron density exists. Why? Answer: the amount of "noise" in our simulation depends upon the number of particles. On average, we have only a few particles for cell. The (random) statistical fluctuations scale as $\pm\sqrt{n}$, where n is the average number of particles per cell. For $n = 4$, we get ± 50% noise!

Ions

Like the electrons, each ion is represented by a coordinate, $\{x_i, v_i\}$.

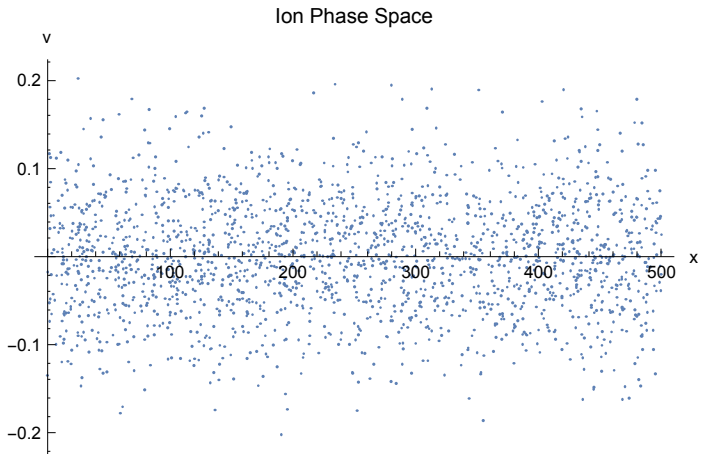
In[•]:=
```
ions = Table[{RandomReal[{0, boxLength}],
    RandomVariate[NormalDistribution[0.0, vthi]]}, {numElectrons}];
```

In[•]:=
```
Dimensions[ions]
```

Out[•]=
```
{2000, 2}
```

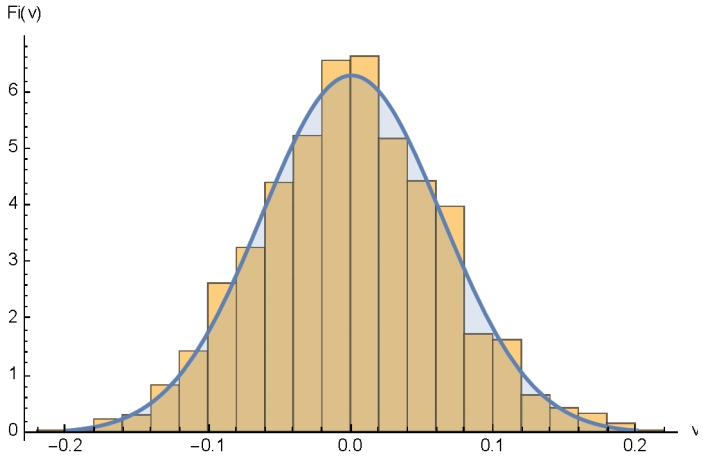
In[•]:=
```
ListPlot[ions, AxesLabel → {"x", "v"}, PlotLabel → "Ion Phase Space"]
```

Out[•]=



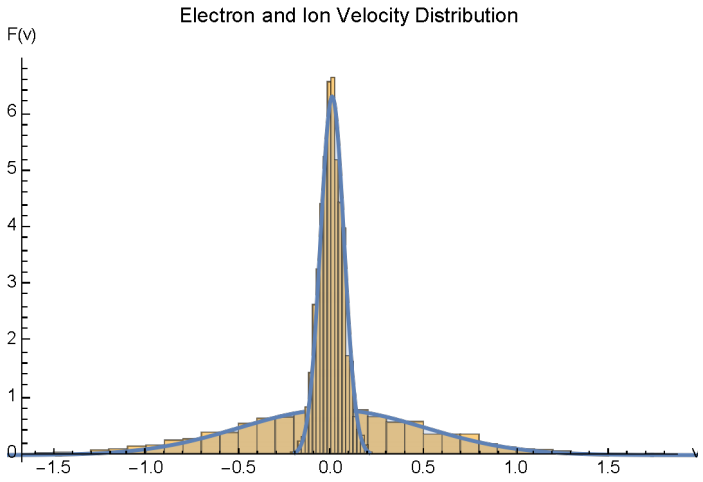
In[•]:=
```
Plot[PDF[NormalDistribution[0, vthi], x], {x, -.2, .2}, Filling → Axis];
Histogram[ions[[All, 2]], Automatic, "PDF"];
pltIonDistribution = Show[%, %%, AxesLabel → {"v", "Fi(v)"}]
```

Out[•]=



In[•]:=
```
Show[pltElectDistribution, pltIonDistribution,
 PlotLabel → "Electron and Ion Velocity Distribution"]
```

Out[•]=



In[•]:=
```
rhoIon = getρ[ions]; // Timing
```

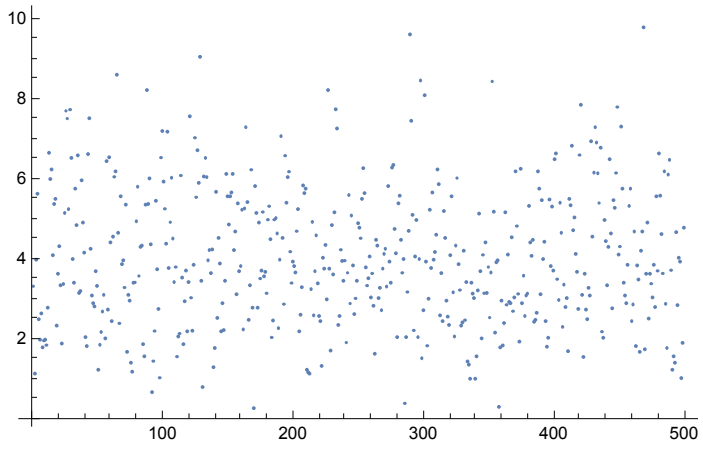
Out[•]=
```
{0.013949, Null}
```

In[•]:=
```
Plus @@ rhoIon (* How many ions? *)
```

Out[•]=
```
2000.
```

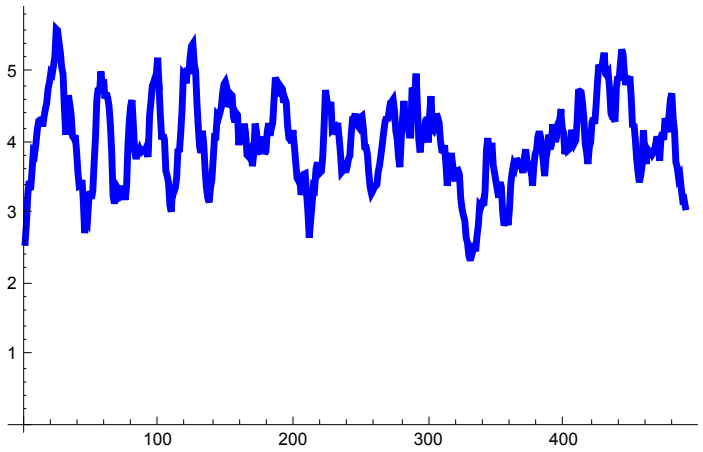
In[•]:=
```
pr1 = ListPlot[rhoIon]
```

Out[•]=



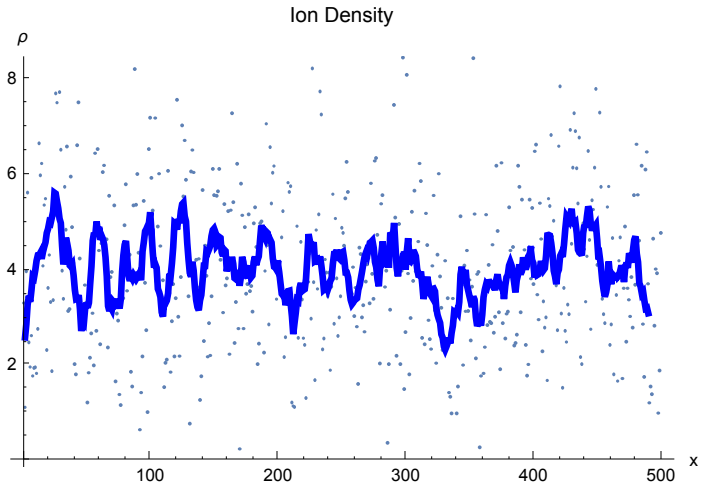
In[ ]:=
```
pr2 = ListPlot[MovingAverage[rhoIon, Floor[λD]],
  Joined → True, PlotStyle → {Blue, Thickness[0.01]}]
```

Out[ ]=



In[ ]:=
```
Show[pr1, pr2, PlotLabel → "Ion Density",
 AxesLabel → {"x", "ρ"}, PlotRange → {0, 2 nAvg}]
```

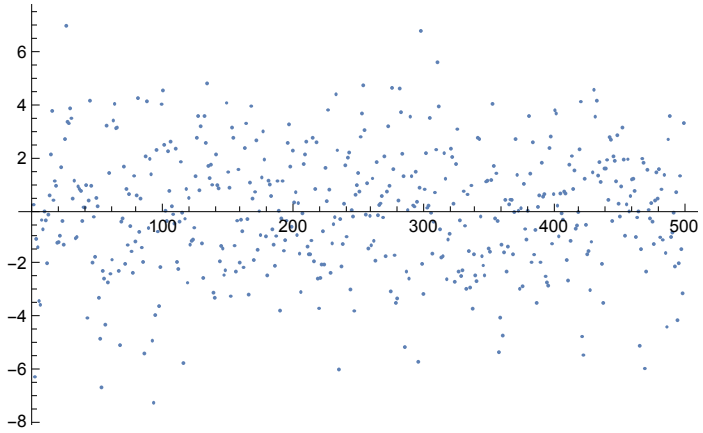
Out[ ]=



Finding the Electric Field

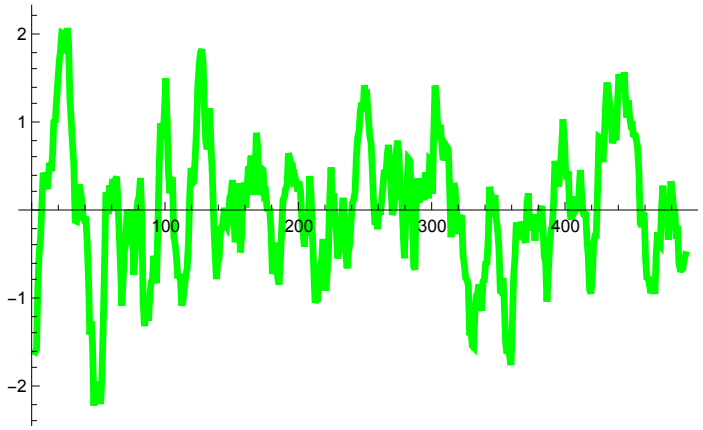
In[ ]:=
```
pr1 = ListPlot[rhoIon - rhoElectron]
```

Out[ ]=



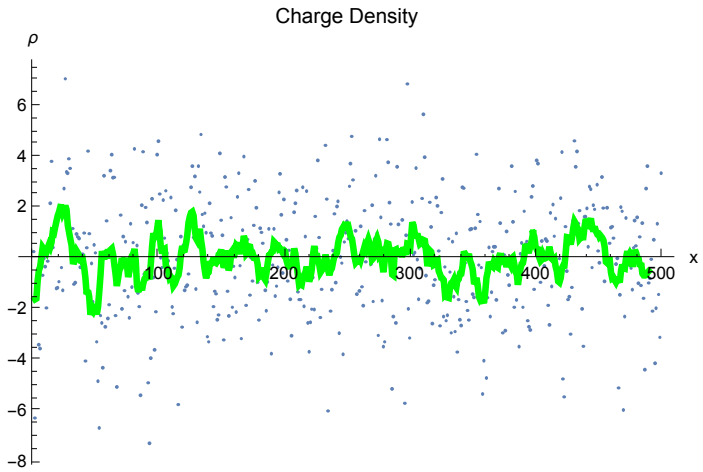
In[ ]:=
```
pr2 = ListPlot[MovingAverage[rhoIon - rhoElectron, Floor[λD]],
  Joined → True, PlotStyle → {Green, Thickness[0.01]}]
```

Out[ ]=



In[•]:=
```
Show[pr1, pr2, PlotLabel → "Charge Density", AxesLabel → {"x", "ρ"}]
```

Out[•]=

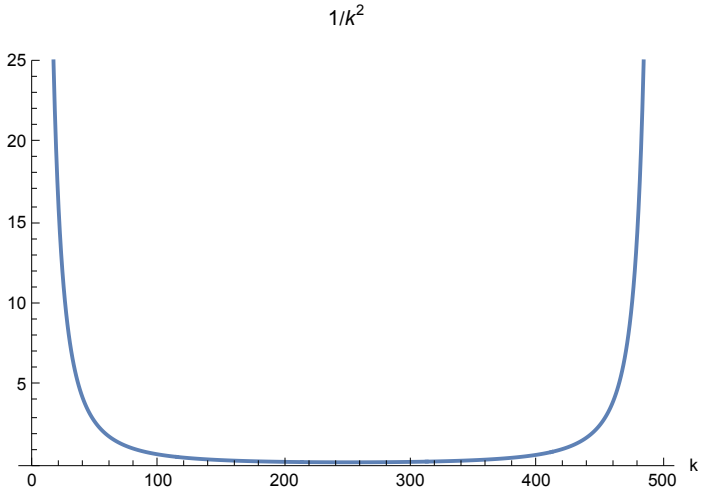


In[•]:=
```
kFourier = Table[2 Sin[2.0 π n / (2 boxLength)], {n, 0, boxLength - 1}];
(* This represents the "effective" wavenumber
 used to invert Poisson's equation with the FFT. *)
```

In[•]:=
```
kFourier2 = kFourier^2;
```

In[•]:=
```
ListPlot[1/Drop[kFourier2, 1], PlotRange → {0, 25},
 PlotLabel → "1/k^2", AxesLabel → {"k", ""}, Joined → True]
```

Out[•]=

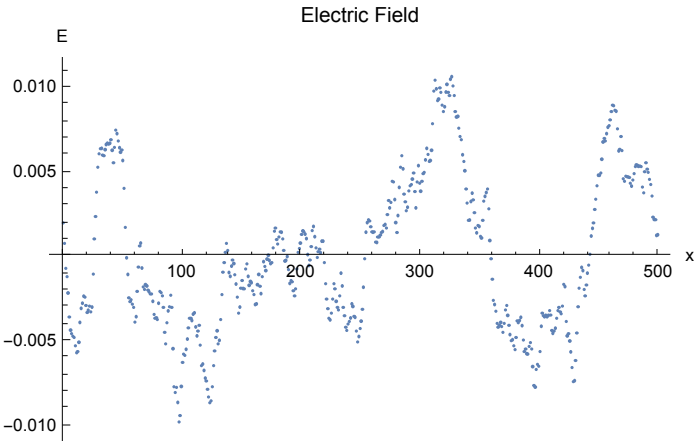


For a given electron and ion density, this computes the electric field using an FFT to invert Poisson's equation...

In[•]:=
```
getE[rhoE_, rhoI_] := Module[{Φ},
  Φ = {0.0} ~Join~ (Drop[Fourier[rhoI - rhoE], 1] / Drop[kFourier2, 1]);
  Φ = Re[InverseFourier[qOverϵ Φ]];
  -(RotateLeft[Φ] - RotateRight[Φ]) / 2]
```

In[•]:=
```
eField = getE[rhoElectron, rhoIon]; // Timing
```
Out[•]=
```
{0.002274, Null}
```

In[•]:=
```
ListPlot[eField, PlotLabel → "Electric Field", AxesLabel → {"x", "E"}]
```
Out[•]=



Of course, with a "noisy" charge density, we also get a "noisy" electric field!

Electron and Ion Step

In this section, we use the "Euler-Cromer" method to advance the position and velocity of a particle for a given electric field.

In[•]:=
```
stepE[eField_List][{x_, v_}] := Module[{loc, locp, force, vNew},
    loc = Floor[x];
    locp = Mod[loc + 1, boxLength] + 1;
    force = eField[[loc + 1]] (loc + 1 - x) + eField[[locp]] (x - loc);
    (* advance velocity *)
    vNew = v - force;
    {Mod[x + vNew, boxLength], vNew}];
```

In[•]:=
```
stepI[eField_List][{x_, v_}] := Module[{loc, locp, force, vNew},
    loc = Floor[x];
    locp = Mod[loc + 1, boxLength] + 1;
    force = eField[[loc + 1]] (loc + 1 - x) + eField[[locp]] (x - loc);
    (* advance velocity *)
    vNew = v + force / ionMass;
    {Mod[x + vNew, boxLength], vNew}];
```

In[•]:=
```
(electrons1 = ParallelMap[stepE[eField][#] &, electrons];) // Timing
```
Out[•]=
```
{1.24184, Null}
```

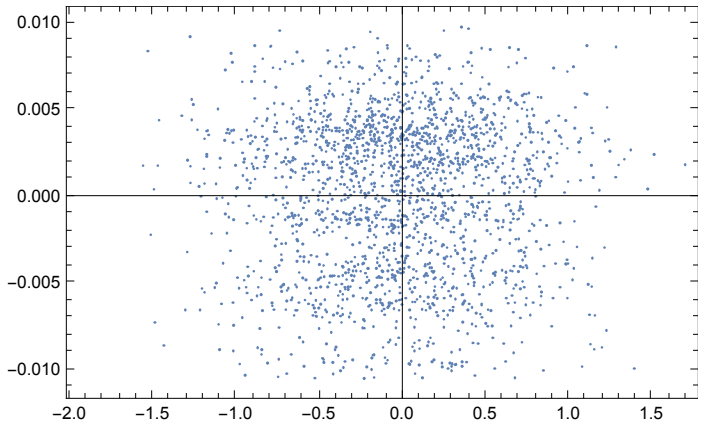
In[•]:=
```
(ions1 = ParallelMap[stepI[eField][#] & , ions];) // Timing
```

Out[•]=
```
{0.097837, Null}
```

View of the change in the particles' phase-space positions...

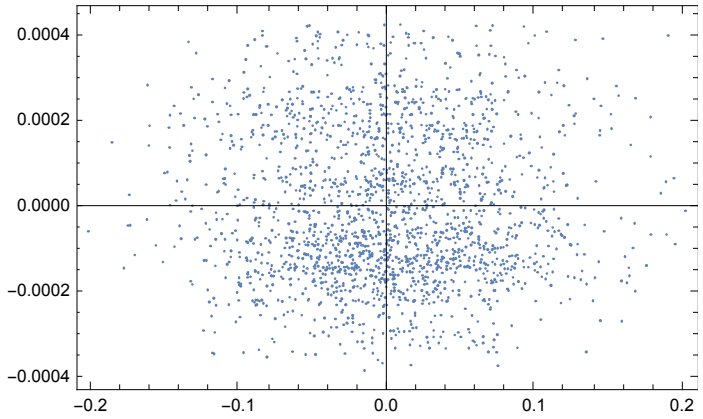
In[•]:=
```
ListPlot[electrons1 - electrons, Frame → True]
```

Out[•]=



In[•]:=
```
ListPlot[ions1 - ions, Frame → True]
```

Out[•]=



Many Steps

Now, we can repeat these calculations for many times steps. Be warned: as we increase the number of particles, this becomes a *very long* computation...

In[•]:=
```
numSteps = 60;
```

In[ ]:=
```
pInd1 = 0; ProgressIndicator[Dynamic[pInd1], {1, numSteps}]
Timing[Do[
   Do[rhoElectron = getρ[electrons];
    rhoIon = getρ[ions];
    eField = getE[rhoElectron, rhoIon];
    electrons = ParallelMap[stepE[eField][#1] &, electrons];
    ions = ParallelMap[stepI[eField][#1] &, ions];, {5}];
   pInd1 = i;
   ePlot1[i] = GraphicsRow[{ListPlot[eField, PlotLabel → "Electric Field",
       AxesLabel → {"x", "E"}, PlotRange → {-0.02, 0.02}, Joined → True],
      Show[ListPlot[electrons, AxesLabel → {"x", "v"}, PlotRange →
         {{1, boxLength}, {-1.5, 1.5}}], ListPlot[ions, AxesLabel → {"x", "v"},
        PlotStyle → Red, PlotRange → {{1, boxLength}, {-1.5, 1.5}}],
       PlotLabel → "Phase Space"]}];, {i, numSteps}];]
```

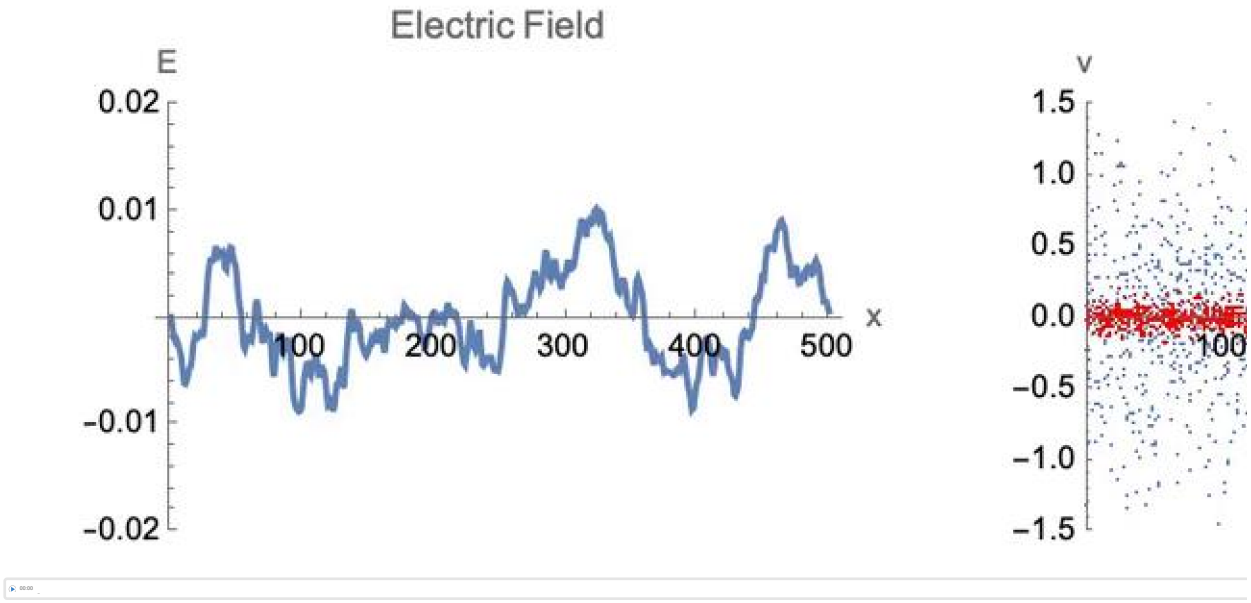
Out[ ]=

[redacted]

Out[ ]=
```
{84.1194, Null}
```

In[ ]:=
```
AnimationVideo[Show[ePlot1[i]], {i, 1, numSteps, 1}]
Export["Efield-1.mov", %, "QuickTime"]
```

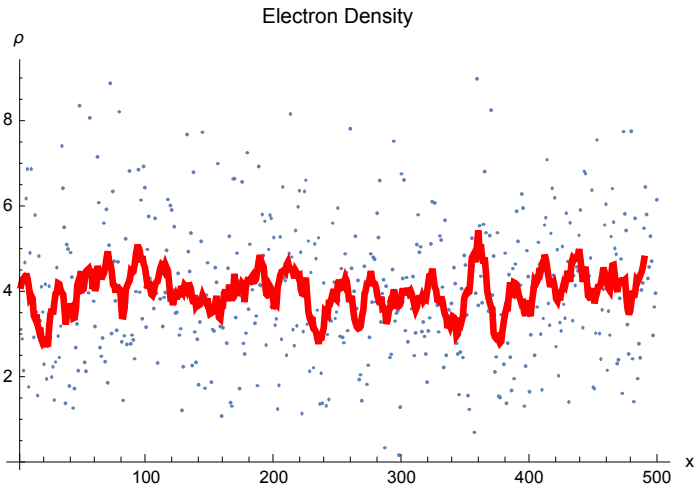
Out[ ]=



Out[ ]=
```
Efield-1.mov
```

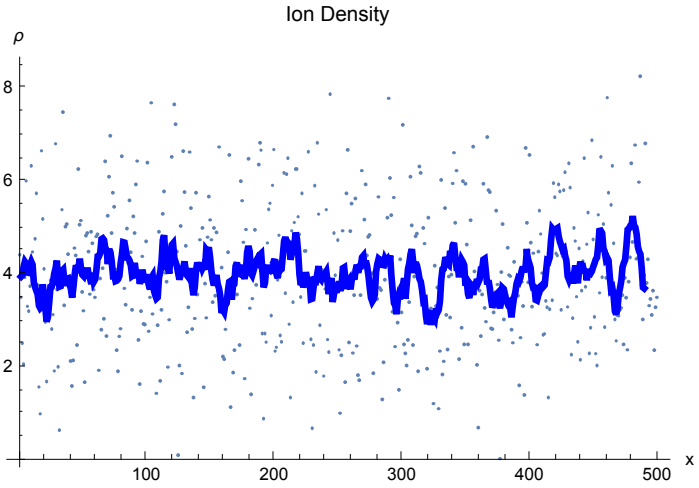
In[•]:=
```
pr1 = ListPlot[rhoElectron];
pr2 = ListPlot[MovingAverage[rhoElectron, Floor[λD]],
   Joined → True, PlotStyle → {Red, Thickness[0.01]}];
Show[pr1, pr2, PlotLabel → "Electron Density", AxesLabel → {"x", "ρ"}]
```

Out[•]=



In[•]:=
```
pr1 = ListPlot[rhoIon];
pr2 = ListPlot[MovingAverage[rhoIon, Floor[λD]],
   Joined → True, PlotStyle → {Blue, Thickness[0.01]}];
Show[pr1, pr2, PlotLabel → "Ion Density", AxesLabel → {"x", "ρ"}]
```

Out[•]=



The mean-squared average velocity of the electrons and ions...

In[•]:=
```
(Plus @@ (Sqrt[#^2] & /@ Last[Transpose[electrons]])) / numElectrons
```

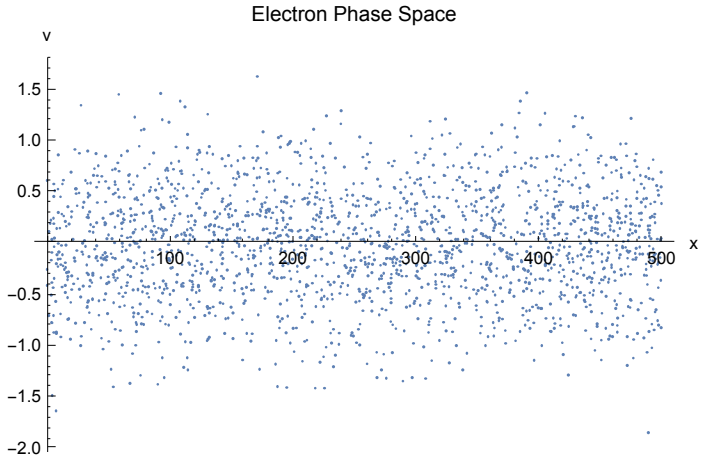
Out[•]=
```
0.420365
```

In[•]:=
```
(Plus @@ (Sqrt[#^2] & /@ Last[Transpose[ions]])) / numElectrons
```

Out[•]=
```
0.0511519
```

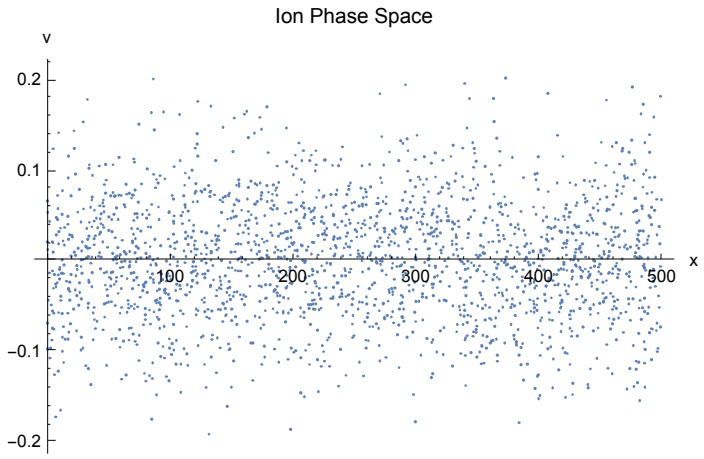
In[ ]:=
```
ListPlot[electrons, AxesLabel → {"x", "v"}, PlotLabel → "Electron Phase Space"]
```

Out[ ]=



In[ ]:=
```
ListPlot[ions, AxesLabel → {"x", "v"}, PlotLabel → "Ion Phase Space"]
```

Out[ ]=



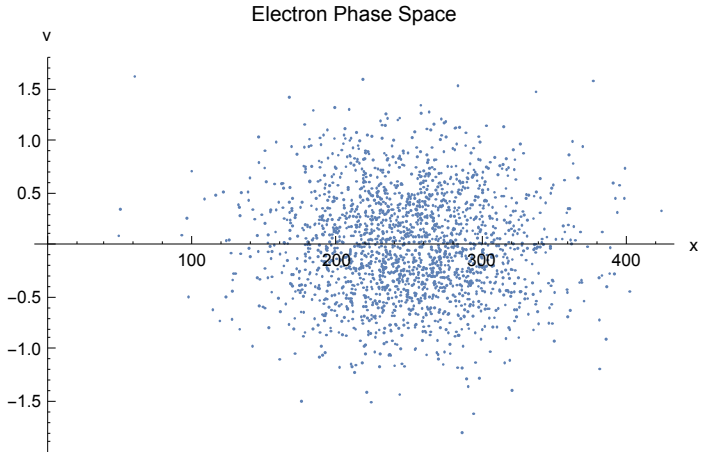
An Initial "Clump"

In[ ]:=
```
spaceDistElectron = NormalDistribution[boxLength / 2.0 , boxLength / 10.0];
spaceDistIon = NormalDistribution[boxLength / 2.0, boxLength / 10.0];
vElectDistribution = NormalDistribution[0.0, vthe];
vIonDistribution = NormalDistribution[0.0, vthi];
```

In[ ]:=
```
electrons = Table[{Mod[Random[spaceDistElectron], boxLength],
    Random[vElectDistribution]}, {numElectrons}];
```

In[•]:=
```
ListPlot[electrons, AxesLabel → {"x", "v"}, PlotLabel → "Electron Phase Space"]
```

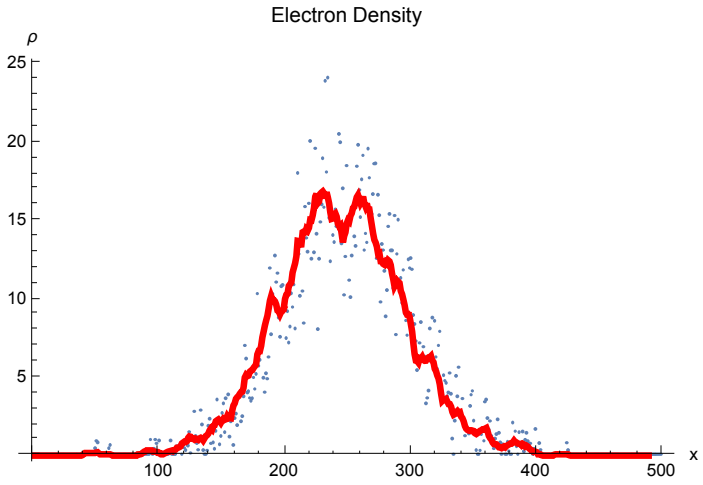
Out[•]=



In[•]:=
```
rhoElectron = getρ[electrons];
```

In[•]:=
```
pr1 = ListPlot[rhoElectron];
pr2 = ListPlot[MovingAverage[rhoElectron, Floor[λD]],
    Joined → True, PlotStyle → {Red, Thickness[0.01]}];
ρe0Plot = Show[pr1, pr2, PlotLabel → "Electron Density", AxesLabel → {"x", "ρ"}]
```

Out[•]=

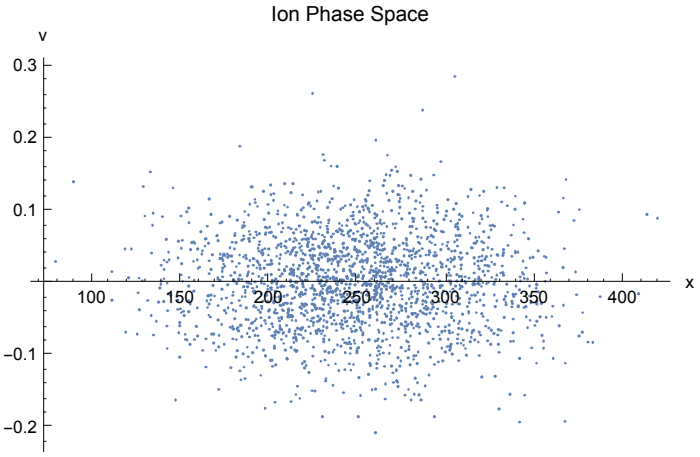


In[•]:=
```
ions = Table[{Mod[Random[spaceDistIon], boxLength],
    Random[vIonDistribution]}, {numElectrons}];
```

In[•]:=
```
rhoIon = getρ[ions];
```

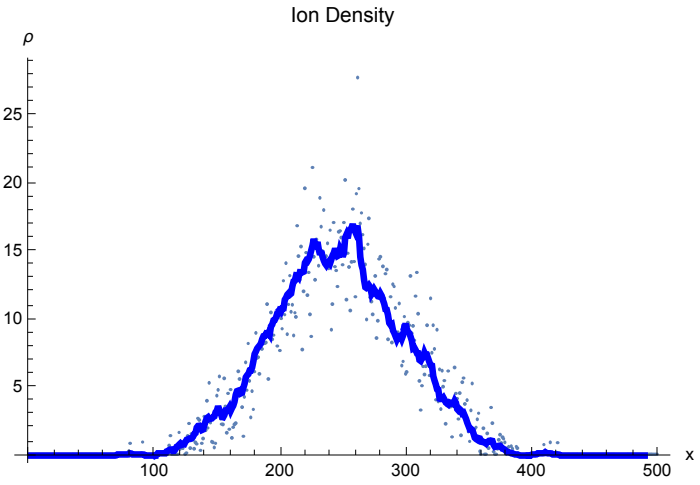
In[ ]:=
```
ListPlot[ions, AxesLabel → {"x", "v"}, PlotLabel → "Ion Phase Space"]
```

Out[ ]=



In[ ]:=
```
pr1 = ListPlot[rhoIon];
pr2 = ListPlot[MovingAverage[rhoIon, Floor[λD]],
    Joined → True, PlotStyle → {Blue, Thickness[0.01]}];
ρi0Plot = Show[pr1, pr2, PlotLabel → "Ion Density", AxesLabel → {"x", "ρ"}]
```

Out[ ]=



Many Steps

In[ ]:=
```
numSteps = 100;
```

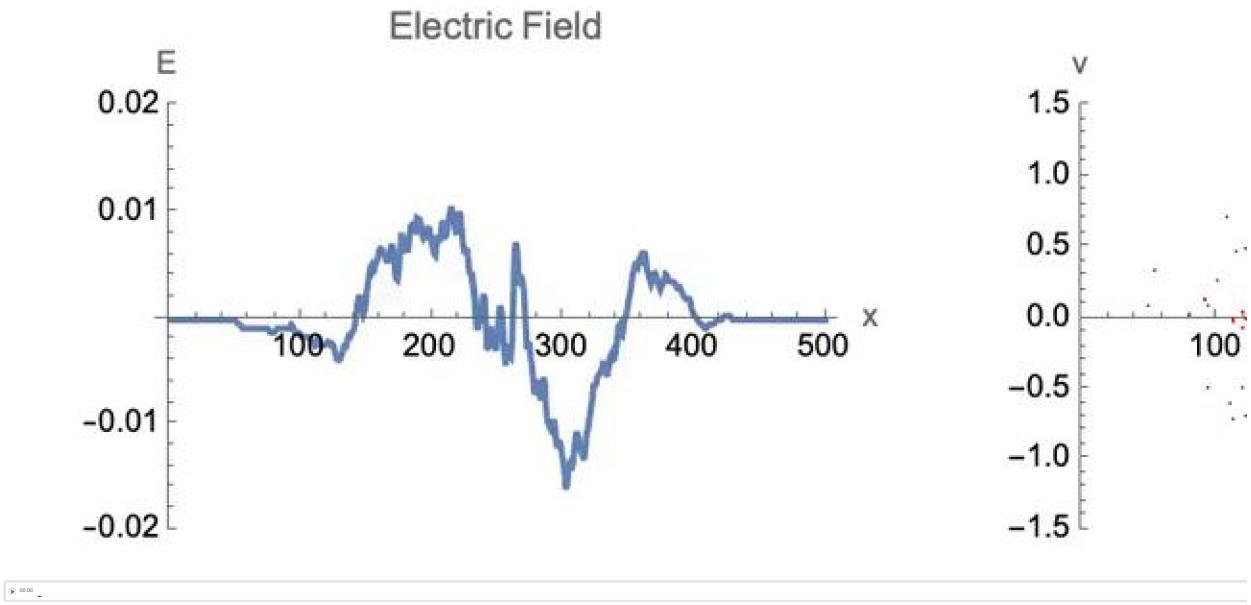
In[ ]:=
```
pInd3 = 0; ProgressIndicator[Dynamic[pInd3], {1, numSteps}]
Timing[Do[
   Do[rhoElectron = getρ[electrons];
    rhoIon = getρ[ions];
    eField = getE[rhoElectron, rhoIon];
    electrons = ParallelMap[stepE[eField][#1] &, electrons];
    ions = ParallelMap[stepI[eField][#1] &, ions];, {10}];
   pInd3 = i;
   ePlot3[i] = GraphicsRow[
     {ListPlot[eField, PlotLabel → "Electric Field", AxesLabel → {"x", "E"},
       PlotRange → {-0.02, 0.02}, Joined → True], Show[ListPlot[electrons,
        AxesLabel → {"x", "v"}, PlotRange → {{1, boxLength}, {-1.5, 1.5}}],
       ListPlot[ions, AxesLabel → {"x", "v"}, PlotStyle → Red,
        PlotRange → {{1, boxLength}, {-1.5, 1.5}}],
       PlotLabel → "Phase Space"]}], {i, numSteps}];]
```

Out[ ]=

Out[ ]=
```
{226.506, Null}
```

In[ ]:=
```
AnimationVideo[Show[ePlot3[i]], {i, 1, numSteps, 1}]
Export["Efield-3.mov", %, "QuickTime"]
```

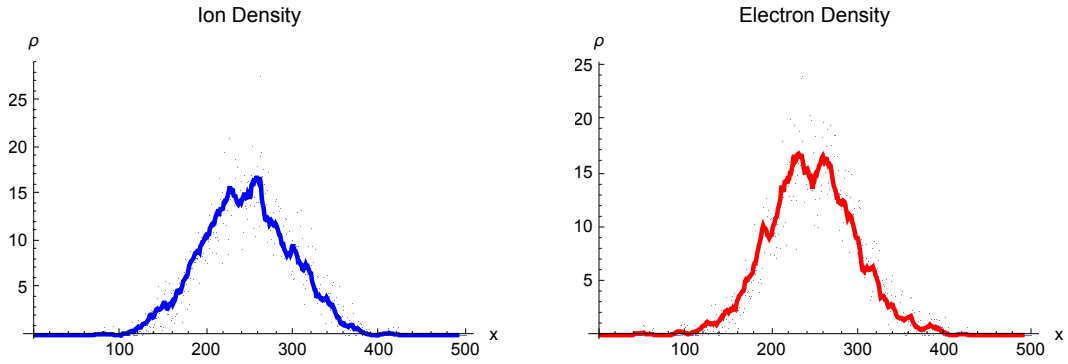
Out[ ]=



Out[ ]=
```
Efield-3.mov
```

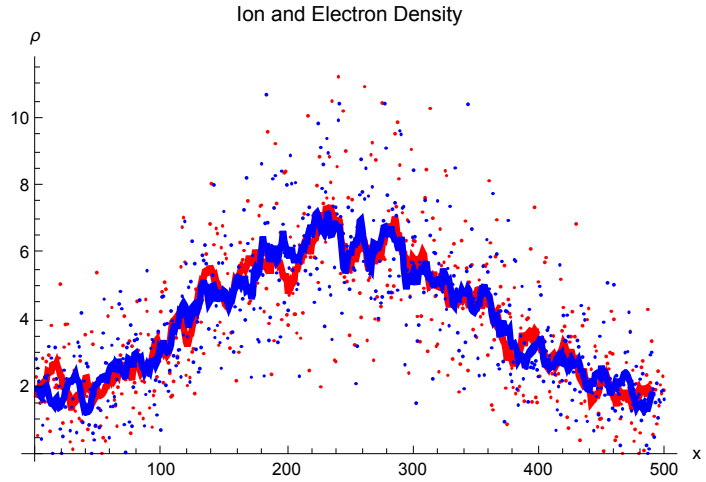
In[•]:=
```
Show[GraphicsRow[{ρi0Plot, ρe0Plot}]]
```

Out[•]=



In[•]:=
```
pr1 = ListPlot[rhoElectron, PlotStyle → Red];
pr2 = ListPlot[MovingAverage[rhoElectron, Floor[λD]],
    Joined → True, PlotStyle → {Red, Thickness[0.01]}];
pr3 = ListPlot[rhoIon, PlotStyle → Blue];
pr4 = ListPlot[MovingAverage[rhoIon, Floor[λD]],
    Joined → True, PlotStyle → {Blue, Thickness[0.01]}];
Show[pr1, pr2, pr3, pr4,
 PlotLabel → "Ion and Electron Density", AxesLabel → {"x", "ρ"}]
```

Out[•]=



Summary

A one-dimensional "particle-in-cell" simulation of a plasma was defined using *Mathematica.* PIC simulations represent a direct model of a plasma allowing computation of dynamical and statistical properties of plasmas. We examined three particularly easy examples determined by the initial conditions of the distributions of ions and electrons. (Many more examples are possible–even with this simple model!) In one example, we looked at "noise" in a *stable* plasma. In the second example, we looked at the evolution of counter-streaming electrons that excite the so-called "two-stream" instability. The

initial kinetic energy of the electrons excites growing electrostatic oscillations. Particle energy is converted into electric-field energy. In the third example, we looked at a "clump" of plasma and observed "ambipolar" electron confinement and the plasma Debye length.