HW 2 - ASTR510

Daniel George

a) Particle-in-Cell code

Initialization

Parameters

In[1]:=
```
q1 = -1.; q2 = -1.;
m1 = 1.; m2 = 1.;
L = 2. Pi ;
ng = 64;
dxg = L / ng;
dt = dxg / 4;
zc = Range[dxg / 2., L, dxg];
wp = Sqrt[2 # ng / L] &;
T = 16 Pi / wp@# &;
```

Function to initialize positions

In[10]:=
```
xF = Function[{r, k}, Module[{x0, x1, x2, np},
    np = r ng;
    x1 = # + .01 L Sin[2 Pi # k / L] &@Range[L / np / 2., L, L / np];
    x2 = x1;
    Join[x1, x2]]];
```

Function to initialize velocities

In[11]:=
```
vF = Function[r, Module[{v1, v2, np}, np = r ng;
    v1 = Table[4., np];
    v2 = -v1;
    Join[v1, v2]]];
```

Function to initialize charge and mass of each particle

In[12]:=
```
qmF = Function[r, Join @@ (Table[#, r ng] & /@ {q1 / m1, q2 / m2})];
```

Function to find the zone of a particle

In[13]:=
```
zone = Mod[Floor[# ng / L], ng] + 1 &;
```

Function to assign densities (CIC)

In[14]:=
```
dCIC = # - Mean@# &[Plus @@ ({q1, q2} * (Total[
            SparseArray[
               Function[l, zone[l] → Abs[l - L / ng Floor[# ng / L + .5]] ng^2 / L^2]
                /@ {# - .5 L / ng, # + .5 L / ng}, {ng}] & /@ #] & /@
          {Take[#, Length[#] / 2], Take[#, -Length[#] / 2]}))] &;
```

This function takes positions of particles and computes densities on the grid. Each particle assigns density to a sparse array which are all summed up at the end.

Function to solve Poisson equation using matrix inversion

In[15]:=
```
Module[{ϕ, solF}, ϕ[ng + 1] = ϕ[1] = 0; ϕ[0] = ϕ[ng];
 solF = LinearSolve[
   CoefficientArrays[Table[ - (ϕ[i - 1] + ϕ[i + 1] - 2 ϕ[i]) / dxg^2 == ρ[i], {i, 2, ng}],
     Table[ϕ[i], {i, 2, ng}]][[2]]];
 phiF = Prepend[solF[#[[2 ;;]]], 0] &;]
```

This function takes densities and returns potential.

Function to interpolate acceleration (equivalent to CIC)

In[16]:=
```
accF = ListInterpolation[
    Append[#, #[[1]]] &@ (RotateLeft[#] - RotateRight[#]) / (2 dxg) &@- phiF[#],
    Range[.5 dxg, L + .5 dxg, dxg], InterpolationOrder → 1, PeriodicInterpolation → True] &;
```

This function takes densities and returns an interpolating function for acceleration.

Iterating with LeapFrog method

Time evolution operator

In[17]:=
```
H = Function[r, {Mod[#1 + dt * #2 + .5 * dt^2 #3, L], #2 + dt #3} &[
     #[[1]], #[[2]], qmF[r] * accF[dCIC[#]][#] &[#[[1]] + .5 dt #[[2]]]] &];
```

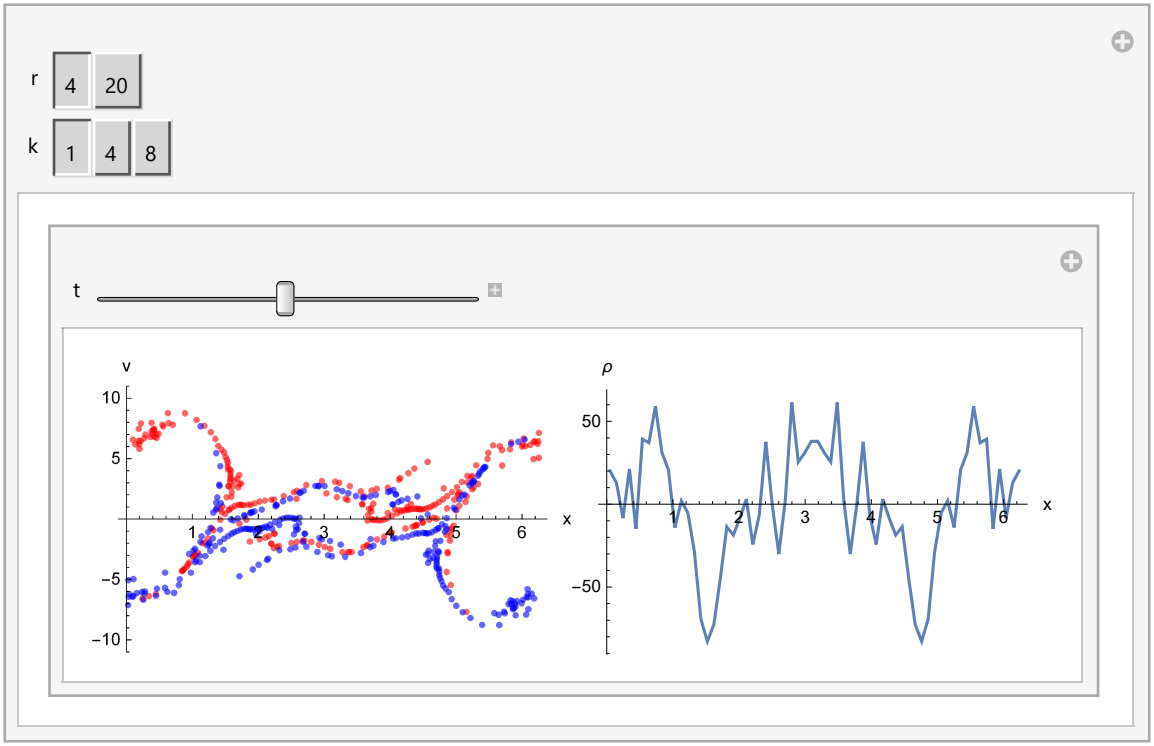
This operator takes positions and momenta as input and returns their values at the next time step.

b)

Interactive plots

In[22]:=
```
Manipulate[w = NestList[H[r], {xF[r, k], vF[r]}, T[r] / dt // Floor];
Manipulate[Row@{ListPlot[
    Take[Transpose[{w[[n, 1]], w[[n, 2]]}], #] & /@ (Length[w[[1, 1]]] / 2 {1, -1}),
    PlotRange → {Min@#, Max@#} &@w[[-1, 2]], PlotStyle → {{Red, ##}, {Blue, ##}} &[
     Opacity[.6], PointSize[.015]], ImageSize → 250, AxesLabel → {"x", "v"}],
   ListLinePlot[Transpose[{zc, dCIC@w[[n, 1]]}], ImageSize → 250,
    AxesLabel → {"x", "ρ"}]}, {{n, 1, "t"}, 1, Length[w], 1}],
 {{r, 4}, {4, 20}}, {{k, 1}, {1, 4, 8}}, SynchronousUpdating → False]
```

Out[22]=



c)