HW 6 - Q2

-- Daniel George

Initialization

Defining functions and constants

In[109]:=
```
np = 100 000; (*Number of particles to be initialized*)
G = 1.; M = 1.; R = 1.;
a = 4.; (*Side length of the box*)
ne = 5; (*The parameter n given in the question*)
ϕ[r_] = -G M (3 R^2 - r^2) / (2 R^3); (*Potential energy inside the sphere*)
vm[r_] = Sqrt[-2 ϕ[r]]; (*Maximum allowed velocity for a particle at r*)
f[e_] = If[e > 0, F e^ (ne - 3 / 2), 0] // Simplify;
(*Distribution function in terms of relative energy*)
e[v_, r_] = -v^2 / 2 - ϕ[r];
(*Relative energy as a function of phase space coordinates*)
```

Defining distribution function in terms of v and r

Analytically integrating to find normalization constant F:

```
sF = Solve[Integrate[(4 Pi)^2 r^2 v^2 f[e[v, r]], {r, 0, R}, {v, 0, vm[r]}] == 1, F]
{{F → 0.05586560778}}
```

Substituting the value of F in the distribution function:

```
f[v_, r_] = f[e[v, r]] /. sF[[1]];
```

This is the normalized distribution function as a function of the magnitudes of velocities and radial distances.

Distributing initial positions uniformly inside a sphere

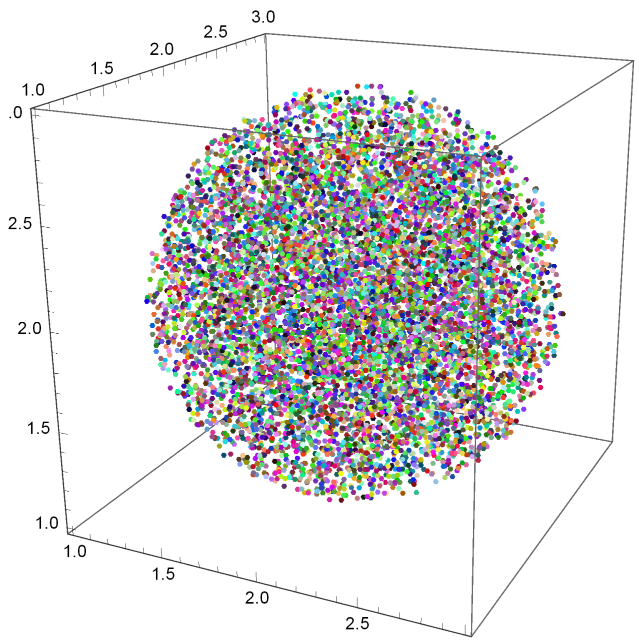
```
x0 = RandomPoint[Ball[{a/2, a/2, a/2}, R], np]; (*This gives the
 initial positions of particles uniformly distributed inside a sphere*)
```

The radial distance of each point is given by:

```
xi = Norm[# - {a/2, a/2, a/2}] & /@ x0;
(*This gives a list containing the magnitude of velocity of each particle*)
```

Plot of initial distribution of positions:

```
ListPointPlot3D[x0[[1 ;; np/10]], BoxRatios → {1, 1, 1}, ImageSize → Medium,
 ColorFunction → Function[{x, y, z}, RGBColor @@ RandomReal[1, 3]]]
```



Calculating the probability distribution function for velocity:

Renormalizing the distribution function at each value of r:

```
pf[v_, r_] =
  v^2 f[v, r] / Integrate[v^2 f[v, r], {v, 0, vm[r]}, Assumptions → r^2 < R^2];
```

Function to calculate the overall velocity distribution:

```
fv[v_] := NIntegrate[r^2 pf[v, r], {r, 0, R}] / Integrate[r^2, {r, 0, R}];
(*Conditional probability of v given a uniform distribution of r*)
```

Inverse transform sampling to find the magnitude of velocities

Finding the CDF at each r analytically:

```
cpv[x_, r_] =
  Integrate[pf[v, r], {v, 0, x}, Assumptions → {0 < x < vm[r], 0 <= r ≤ R}];
```

Numerically finding the inverse of the CDF using the secant method:

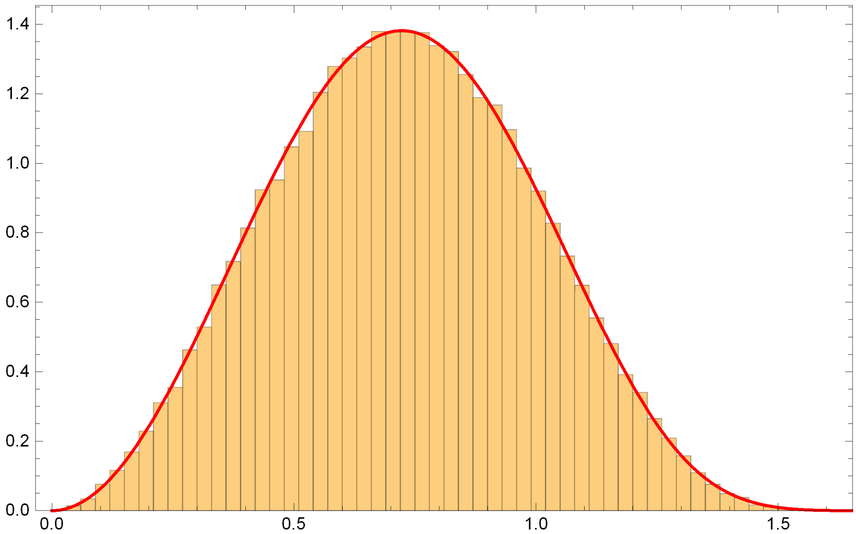
```
rv[r_] := FindRoot[cpv[x, r] == RandomReal[], {x, 0.0001, vm[r] - .0001}][[1]][[2]]
(*The function rv takes a value of|r|
 and returns a value of|v| using inverse transform sampling*)
```

Applying this function to every particle to calculate its speed:

```
vi = rv /@ xi;
```

Comparing the obtained velocities with the expected distribution:

```
vplot = Plot[fv[v], {v, 0, vm[0]}, PlotStyle → Red];
Show[Histogram[vi, {.03}, PDF, Frame → True], vplot]
```



Thus the obtained distribution of velocities closely matches the expected distribution.

Finding intial velocity vectors

Generating random unit vectors for directions for velocities:

```
n0 = RandomPoint[Sphere[], np];
(*Gives random points on the surface of a unit sphere*)
```

Multiplying the unit vectors with the speeds to get the initial velocities:

```
v0 = vi * n0; (*Initial velocities of particles*)
```

Subtracting center of mass velocity:

```
vmean = Total[v0] / np;
v0 = (# - vmean) & /@ v0;
```

Writing data to files

```
Export["A:\\NBody\\x0np100k.csv", x0, "Table"];
Export["A:\\NBody\\v0np100k.csv", v0, "Table"];
```

Computation

Defining functions and constants

Constants:

```
In[64]:= nz = 128 (*Number of zones = nz^3*);
L = 6. (*Length of the box*);
G = 1.; M = 1.; R = 1.;
h = L / nz;  (*Zone width*)
dt = .01 * (Pi / 2 * R^ (3 / 2)) / (Sqrt[2 G] * Sqrt[M]); (*Time step = 0.01 tff*)
nP = 50 000;
(*Number of particles to be simulated*)
```

Extracting intialization data from files:

```
In[47]:= x1 = Take[Import["A:\\NBody\\n5x0np100k.csv", "Table"], nP];
(*Initial positions*)
v1 = Take[Import["A:\\NBody\\n5v0np100k.csv", "Table"], nP];
(*Initial velocities*)

x1 = # + {L/2 - 2, L/2 - 2, L/2 - 2} & /@ x1;
(*Shifting coordinates to move particles to center of box*)
```

Calculating the Green's function (parallelized):

```
In[49]:= cGf = Compile[{{x, _Integer}, {y, _Integer},
     {z, _Integer}, {n, _Integer}, {l, _Real}, {g, _Real}},
    If[x == y == z == 1, 0.0, -l^2 g/n^2/(Min[2 n + 1 - x, -1 + x]^2 +
          Min[2 n + 1 - y, -1 + y]^2 + Min[2 n + 1 - z, -1 + z]^2)^0.5]];
(*Compiled function to calculate Green's function values in real space*)

FmG = Fourier[ParallelArray[cGf[#1, #2, #3, nz, L, G] &, {2 nz, 2 nz, 2 nz}]];
(*This is the Green's function array in fourier space*)
```

Function to find the zone of a particle:

```
In[69]:= zone = Floor[# nz / L] + 1 & /@ # &;
(*Adding 1 since arrays are indexed starting from 1 in Mathematica*)
```

Function to assign densities according to the NGP scheme:

Creating a sparse array for each particle with the value 1 assigned to its zone and finally taking the total sum.

```
In[70]:= dNGP :=
  M / nP/(L / nz)^3 Total[SparseArray[# → 1.0, {2 nz, 2 nz, 2 nz}] & /@ zone[#]] &;
```

Function to assign densities according to the CIC scheme:

Assuming each particle to be a cube with side length = L/nz, we can assign to each of the 8 nearest zones a weightage equal to the fraction of the volume occupied by the particle in that zone:

```
In[71]:= fCIC = Function[r,
    zone[r + #] → (nz / L)^3 Times @@ Abs[r + # - L / nz Floor[r nz / L + 1/2]] & /@
     Tuples[L/2/nz {1, -1}, 3]];
```

Now creating sparse arrays for the density of each particle and then summing them up:

```
In[72]:= dCIC := M / nP/(L / nz)^3 Total[SparseArray[fCIC[#], {2 nz, 2 nz, 2 nz}] & /@ #] &;
```

Function to compute the potential from density:

In[55]:=
```
phi = (2 nz) ^1.5 Take[Re[InverseFourier[Fourier[dNGP[#]] * FmG]], nz, nz, nz] &;
(*NGP scheme*)
phiC = (2 nz) ^1.5 Take[Re[InverseFourier[Fourier[dCIC[#]] * FmG]], nz, nz, nz] &;
(*CIC scheme*)
```

Function to calculate the gradient at a zone:

In[57]:=
```
grad[m_] =
  1 / (2 h) {m[[#[[1]] + 1, #[[2]], #[[3]]]] - m[[#[[1]] - 1, #[[2]], #[[3]]]], m[[#[[1]], #[[2]] + 1, #[[3]]]] -
     m[[#[[1]], #[[2]] - 1, #[[3]]]], m[[#[[1]], #[[2]], #[[3]] + 1]] - m[[#[[1]], #[[2]], #[[3]] - 1]]} &;
```

Function to compute the force from the potential:

In[58]:=
```
force = -grad[phi[#[[1]] + dt * #[[2]] / 2]] /@ zone[#[[1]] + dt * #[[2]] / 2] &;
(*NGP scheme*)
forceC = -grad[phiC[#[[1]] + dt * #[[2]] / 2]] /@ zone[#[[1]] + dt * #[[2]] / 2] &;
(*CIC scheme*)
```

Function to remove the particles which leave the grid:

In[60]:=
```
kill = Transpose[Select[Transpose[#],
    Min[(#[[1]] + dt * #[[2]] / 2)] > L / nz && Max[(#[[1]] + dt * #[[2]] / 2)] < L - L / nz &]] &;
```

Time evolution operator for the leapfrog method:

In[61]:=
```
H = {#1 + dt * #2 + (1 / 2) * dt^2 #3, #2 + dt #3} &[#[[1]], #[[2]], force[#]] &[kill[#]] &;
(*NGP scheme*)
HC = {#1 + dt * #2 + (1 / 2) * dt^2 #3, #2 + dt #3} &[#[[1]], #[[2]], forceC[#]] &[kill[#]] &;
(*CIC scheme*)
```

Iterating

Applying H repeatedly to initial distribution:

Storing values after every 5 time steps:

In[98]:=
```
w = NestList[Nest[HC, #, 5] &, {x1, v1}, 1]; // AbsoluteTiming
```

Out[98]=
```
{26.41779756, Null}
```

Appending values for more time steps:

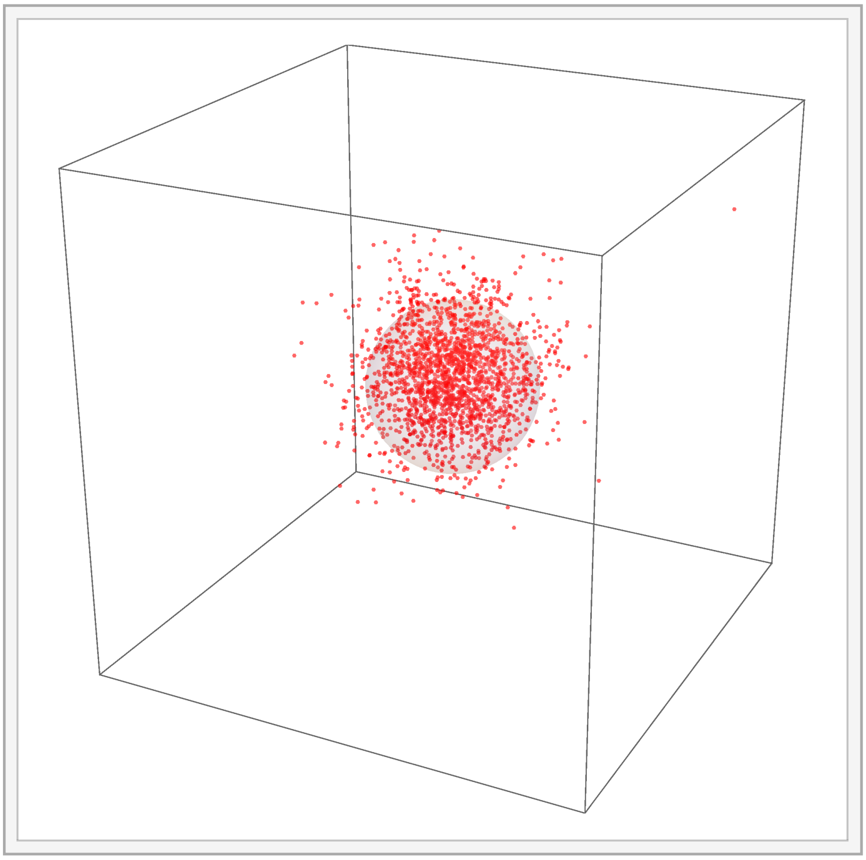
In[101]:=
```
Clear[wx]; w = Import["A:\\NBody\\Cn5np10000nz128L6.mx"];
```

In[71]:=
```
Dynamic@{Length[w], ByteCount[w] / 2.^30}
```
Out[71]= {502, 0.2238589972}

In[72]:=
```
While[Length[w] ≤ 5 / .01 && Length[w[[-1, 1, All]]] > nP / 10,
   w = Join[w[[1 ;; -2]], NestList[Nest[H, #, 5] &, w[[-1]], 5]]]; // AbsoluteTiming
```
Out[72]= {4071.482549, Null}

Visualization

In[112]:=
```
Manipulate[Show[ListPointPlot3D[w[[i, 1, a ;; a + 2000]],
    BoxRatios → {1, 1, 1}, Axes → False, ImageSize → {400, 400},
    PlotRange → {#, #, #} &[{0, L}], ViewPoint → {Pi, Pi / 2, 1.5},
    PlotStyle → {Red, PointSize → .006, Opacity[0.6]}],
  Graphics3D[{Opacity[0.15], Sphere[{1, 1, 1} L / 2, R]}]],
 {{i, -1}, 1, Length[w], 1(*,Appearance→{"Labeled"}*)},
 {a, 1, Length[w[[-1]][[1]]] - 2000, 1}]
```

Out[112]=


Writing data to files

In[73]:=
```
Export["A:\\Cn5np10000nz128L6.mx", w];
```

Creating animated GIF:

In[74]:=
```
Export["A:\\Google Drive\\Acads\\Cn5np10000nz128L6.gif",
  Table[Show[ListPointPlot3D[w[[i, 1, 1 ;; 1 + 2000]],
     BoxRatios → {1, 1, 1}, Axes → False, ImageSize → {600, 600},
     PlotRange → {#, #, #} &[{0, L}], ViewPoint → {Pi + 2 i / Length[w], Pi / 2, 1.5},
     PlotStyle → {Red, PointSize → .005, Opacity[0.8]}], Graphics3D[
     {Opacity[0.15], Sphere[{1, 1, 1} L / 2, R]}]], {i, 1, Length[w], 1}]];
```

Results

For n=5, L=6, 50000 particles, 128^3 zones, NGP scheme

Importing data:

In[1]:=
```
Clear[wx]; wx = Import["n5np50000nz128L6.mx"]; nP = Length[wx[[1, 1, All]]];
```

Percentage of particles that left the grid:

In[2]:=
```
(100 - Length[wx[[-1, 1, All]]] / 50 000 * 100.) "%"
```

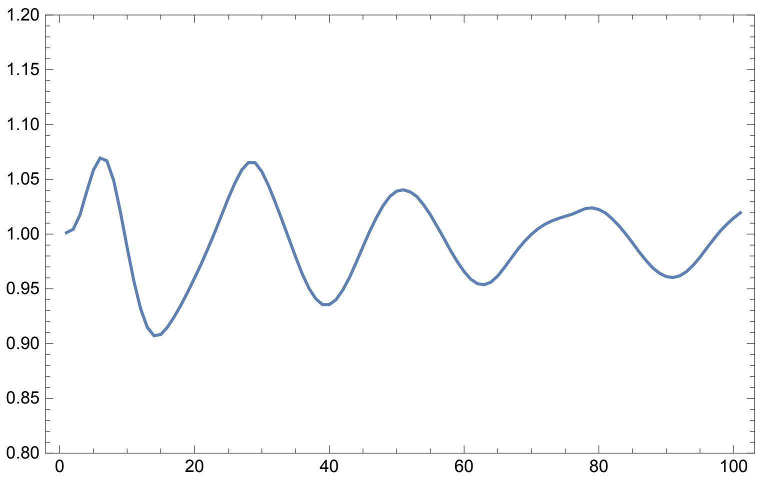
Out[2]= 0.428 %

Plotting virial ratio vs iteration:

In[43]:=
```
vr5 = Table[2 Total[Norm[#]^2 & /@ wx[[i, 2, All]]] / Total[
      Extract[-phi[wx[[i, 1, All]]], zone[wx[[i, 1, All]]]]], {i, 1, Length[wx], 5}];
```

In[174]:=
```
ListLinePlot[vr5, Frame → True, PlotRange → {.8, 1.2}]
```

Out[174]=



The system starts from a virial ratio of ~1 and undergoes minor fluctuations before settling down to a final equilibrium state with virial ratio =1.

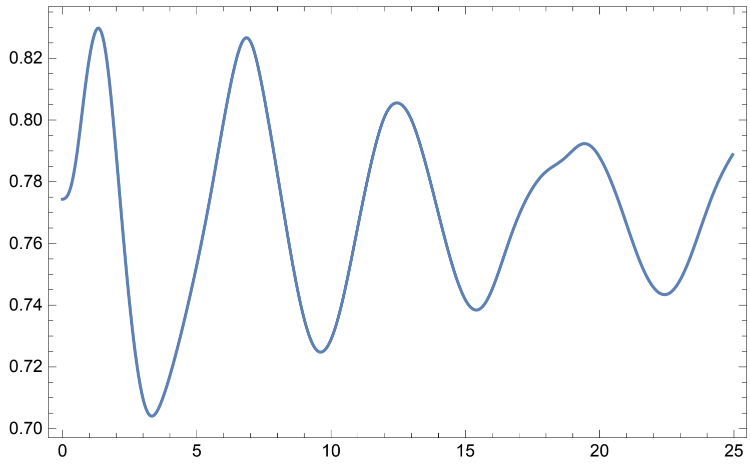
The rate of evolution of the system can be characterized by the frequency of oscillations which is about 4 cycles in 25 t_{ff}

Velocity dispersion vs time (in units of t_{ff})

In[19]:=
```
vd5 = Table[Norm@StandardDeviation[wx[[i, 2, All]]], {i, 1, Length[wx], 1}];
```

In[25]:=
```
ListLinePlot[
 Transpose[{Range[0, 25 - 25 / Length[vd5], 25 / Length[vd5]], vd5}], Frame → True]
```

Out[25]=



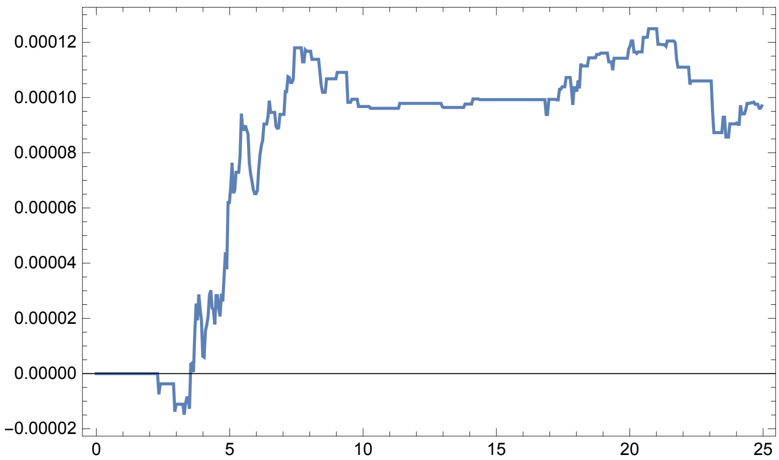
The velocity dispersion also exhibits damped oscillatory behavior about the value of .77

Mean velocity vs time (in units of t_{ff})

In[36]:=
```
vm5 = Table[Norm@Mean[wx[[i, 2, All]]], {i, 1, Length[wx], 1}]; vm5 = vm5 - vm5[[1]];
```

In[37]:=
```
ListLinePlot[
 Transpose[{Range[0, 25 - 25 / Length[vm5], 25 / Length[vm5]], vm5}], Frame → True]
```

Out[37]=



The magnitude of mean velocity of the system remains close to zero at all times thus the system is mostly anisotropic with a final mean velocity (possibly due to numerical errors) equal to:

In[38]:=
```
vm5[[-1]]
```

Out[38]=
```
0.00009698865662
```

Radial density distributions:

Grouping zones within spherical shells of width L/n :

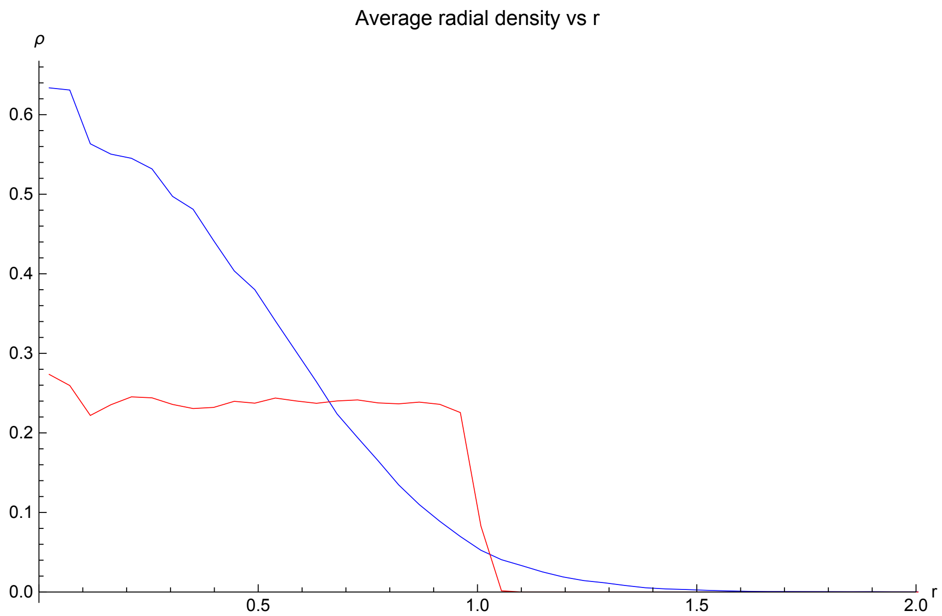
In[39]:=
```
nz = 128; grid = GatherBy[Tuples[Range[nz], 3], Floor[Norm[# - .5 - nz / 2]] &];
```

Averaging the initial potential over all the zones within each group:

```
avd = With[{dm = dCIC[wx[[-1, 1, All]]]}, Reverse[Mean /@ (Extract[dm, #] & /@ grid)]];
(*Final radial density*)
avdi = With[{dm = dCIC[wx[[1, 1, All]]]}, Reverse[Mean /@ (Extract[dm, #] & /@ grid)]];
(*Initial radial density*)
```

In[113]:=
```
Show[ListLinePlot[Transpose[{(Range[1, Length[avd]] - .5) L / nz, avd}],
  PlotStyle → {Blue, Thickness[.001]}, AxesLabel → {HoldForm["r"], HoldForm["ρ"]},
  PlotLabel → HoldForm["Average radial density vs r"], PlotStyle →
   {PointSize[.009], RGBColor @@ RandomReal[1, 3]}, PlotRange → {{0, 2}, All}],
 ListLinePlot[Transpose[{(Range[1, Length[avdi]] - .5) L / nz, avdi}],
  PlotStyle → {Red, Thickness[.001]},
  AxesLabel → {HoldForm["r"], HoldForm["ϕ"]}, PlotRange → {{0, 2}, All}]]
```

Out[113]=



Blue is the final density profile and red is the initial density profile. Thus over time most of the mass becomes concentrated towards the center. Also from the 3D animations we can see that the system remains approximately spherically symmetric.

For n=4, L=6, 50000 particles, 128^3 zones, NGP scheme

Importing data:

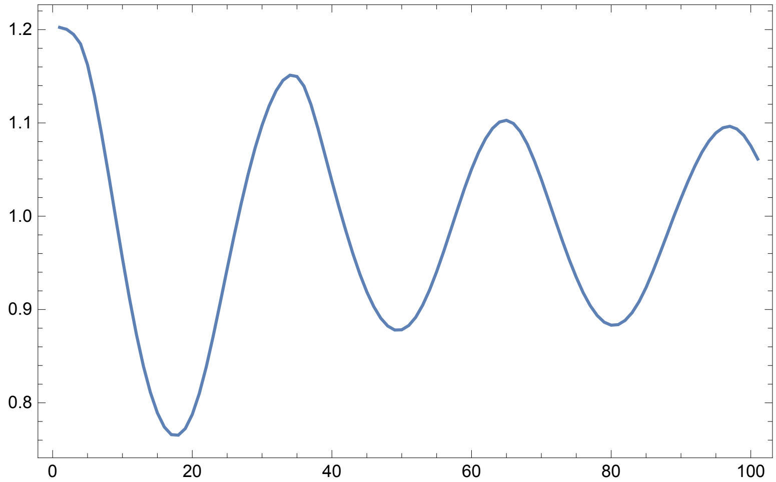
```
Clear[wx]; wx = Import["n4np50000nz128L6.mx"]; nP = Length[wx[[1, 1, All]]];
```

Percentage of particles that left the grid:

In[46]:=
```
(100 - Length[wx[[-1, 1, All]]] / 50000 * 100.) "%"
```
Out[46]= 3.142 %

Plotting virial ratio vs iteration:

In[47]:=
```
vr4 = Table[2 Total[Norm[#]^2 & /@ wx[[i, 2, All]]] / Total[
    Extract[-phi[wx[[i, 1, All]]], zone[wx[[i, 1, All]]]]], {i, 1, Length[wx], 5}];
```

In[77]:=
```
ListLinePlot[vr4, Frame → True]
```
Out[77]=


The system starts from a virial ratio of ~1.2 and undergoes a sequence of collapses and expansions before settling down to a final equilibrium state with virial ratio =1.

The rate of evolution of the system can be characterized by the frequency of oscillations which is about 3 cycles in 25 t_{ff}

For n=5, L=6, 10000 particles, 128^3 zones, CIC scheme

Importing data:

In[167]:=
```
Clear[wx]; wx = Import["A:\\NBody\\Cn5np10000nz128L6.mx"];
nP = Length[wx[[1, 1, All]]];
```

Percentage of particles that left the grid:

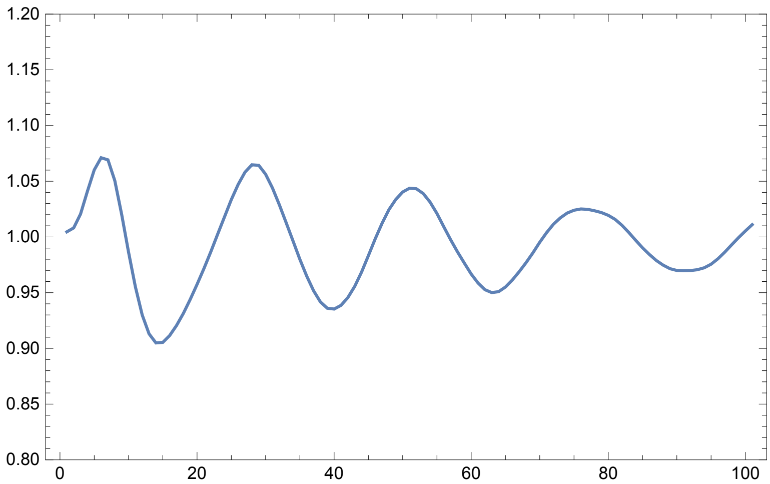
In[169]:=
```
(100 - Length[wx[[-1, 1, All]]] / nP * 100.) "%"
```
Out[169]= 0.51 %

Plotting virial ratio vs iteration:

```
In[171]:= vr5C = Table[2 Total[Norm[#]^2 & /@ wx[[i, 2, All]]] / Total[
              Extract[-phiC[wx[[i, 1, All]]], zone[wx[[i, 1, All]]]]], {i, 1, Length[wx], 5}];
```

```
In[175]:= ListLinePlot[vr5C, Frame → True, PlotRange → {.8, 1.2}]
```

Out[175]=



Again this system starts from a virial ratio of ~1 and undergoes minor fluctuations before settling down to a final equilibrium state with virial ratio =1.

The rate of evolution of the system can be characterized by the frequency of oscillations which is about 4 cycles in 25 t_{ff}

Tests

Testing Poisson Solver with the initial density

Grouping zones within spherical shells of width L/n :

```
grid = GatherBy[Tuples[Range[nz], 3], Floor[Norm[# - .5 - nz / 2]] &];
```

Averaging the initial potential over all the zones within each group:

```
Vr = Block[{phi0 = phiC[x0]}, Reverse[Mean /@ (Extract[phi0, #] & /@ grid)]];
```

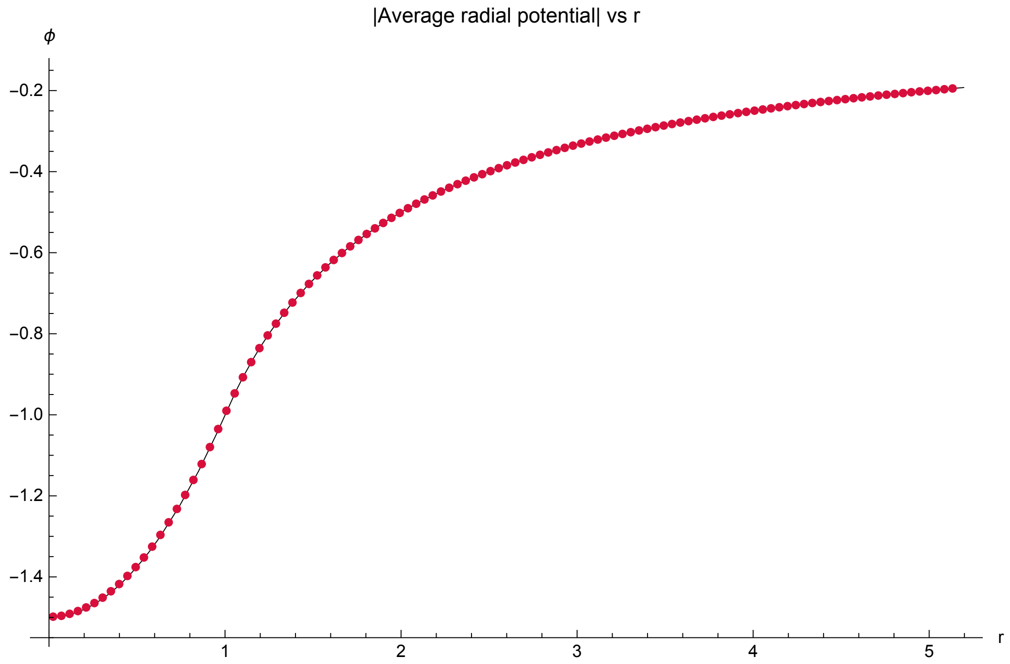
Analytical solution:

```
V[r_] := If[Abs[r] < R, -G M (3 R^2 - r^2) / (2 R^3), -G M / Abs[r]]
```

The computed potential matches the analytic potential of a uniform density

sphere as expected:

```
Show[Plot[V[x], {x, 0, Sqrt[3] L/2}, PlotStyle → {Black, Thickness[.001]},
  AxesLabel → {HoldForm["r"], HoldForm["ϕ"]},
  PlotLabel → HoldForm["|Average radial potential| vs r"]],
 plt[nz] = ListPlot[Transpose[{(Range[1, Length[Vr]] - .5) L/nz, Vr}],
   PlotStyle → {PointSize[.009], RGBColor @@ RandomReal[1, 3]}]]
```



Thus the Poisson solver appears to be working correctly as we are getting the correct initial potential.

Testing Leap Frog with Kepler potential

In[103]:=
```
dtK = .001; (*Time step size*)
```

In[104]:=
```
forceK := -G M (# - {L/2, L/2, L/2}) / Norm[# - {L/2, L/2, L/2}]^3 & /@
    (#[[1]] + dtK * #[[2]] / 2) &; (*Central force*)
```

In[105]:=
```
killK = Transpose[Select[Transpose[#],
      (Min[(#[[1]] + dt * #[[2]] / 2)] > L / nz && Max[(#[[1]] + dt * #[[2]] / 2)] < L - L / nz &&
         Norm[(#[[1]] + dtK * #[[2]] / 2) - {L/2, L/2, L/2}] > 0.1 &)]] &;
```

Time evolution operator:

In[106]:=
```
HK = {#1 + dtK * #2 + (1/2) * dtK^2 #3, #2 + dtK #3} &[#[[1]], #[[2]], forceK[#]] &;
(*Time evolution operator*)
```

Iterating for 100 particles:

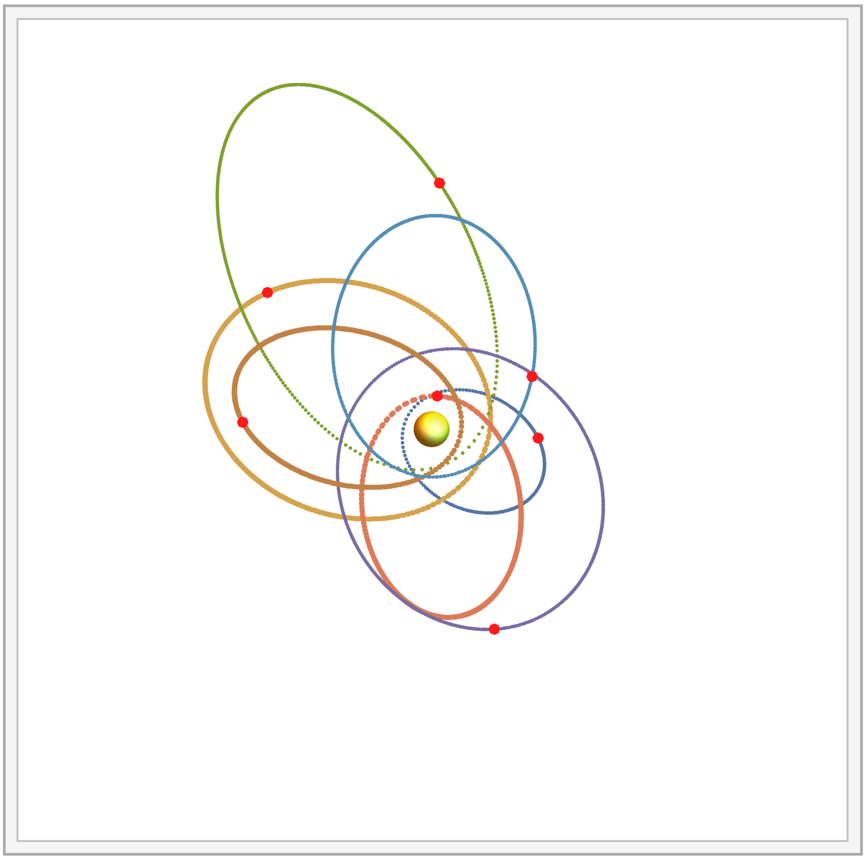
In[107]:=
```
wK0 = {x1[[1 ;; 100]], v1[[1 ;; 100]]}; wK0[[1, All, 3]] = L/2; wK0[[2, All, 3]] = 0;
(*Confining initial positions and velocities in the x-y plane*)
```

In[109]:=
```
wK = NestList[HK[killK[#]] &, wK0, 10 000];
```

Visualization:

In[110]:=
```
Manipulate[
 Show[ListPointPlot3D[Transpose[wK[[i - 5000 ;; i ;; 10, 1, a ;; a + n]], {2, 1, 3}],
   Boxed → False, BoxRatios → {1, 1, 1}, Axes → False,
   ImageSize → {400, 400}, PlotRange → {#, #, #} &[{L / 4, 3 L / 4}],
   ViewPoint → {0, 0, Infinity}, PlotStyle → {PointSize → .005, Opacity[0.8]}],
  ListPointPlot3D[wK[[i, 1, a ;; a + n]],
   PlotStyle → {Red, PointSize → .015, Opacity[0.9]}], Graphics3D[
   {Opacity[0.9], Specularity[White, 5], Yellow, Sphere[{1, 1, 1} L/2, .07 R]}]],
 {{i, 6000}, 4001, Length[wK], 1, Appearance → {"Labeled"}},
 {{n, 5}, 0, Length[wK[[-1, 1]]] - 1, 1},
 {{a, 5}, 1, Length[wK[[-1, 1]]] - n, 1}]
```

Out[110]=



Thus the Leap Frog routine appears to be working correctly as we are getting stable elliptical orbits satisfying Kepler’s laws.

HW – 6

Q1) a) $$\phi(r) = -\int d^3r'\,\rho(r')\,\frac{G e^{-\alpha(r-r')}}{|r-r'|}$$

$$-\nabla\phi(r) = G\int \frac{e^{-\alpha(r-r')}}{|r-r'|^2}\left[1+\alpha(r-r')\right]\rho(r')\,d^3r'$$

$$-\nabla\cdot\nabla\phi(r) = -\nabla^2\phi$$

$$= G\int d^3r'\,\rho(r')\,\alpha^2\frac{e^{-\alpha(r-r')}}{|r-r'|} - G\int d^3r'\,\rho(r')\left[\frac{-2e^{-\alpha(r-r')}}{|r-r'|^3} - \frac{2\alpha e^{-\alpha(r-r')}}{|r-r'|^2}\right]$$

$$\Rightarrow \nabla^2\phi = \alpha^2\phi + G\int d^3r'\,\rho(r')\left[4\pi\,\delta(r-r')\right]$$

$$\Rightarrow (\nabla^2-\alpha^2)\phi = 4\pi G\rho(r) \quad //$$

This is the analogue of Poisson's equation

b) Substitute $\nabla^2 \rightarrow \nabla^2-\alpha^2$ in the derivation of Jean's wave number

$\Rightarrow k^2 \rightarrow k^2+\alpha^2$ in Fourier space

Following the same steps as in the notes

$$\Rightarrow \frac{k^2+\alpha^2}{k_J^2} = 1 - \frac{\sqrt{\pi}\,\gamma}{k\alpha\sqrt{2}}\exp\left(\frac{\gamma^2}{2k^2\alpha^2}\right)\left[1-\mathrm{erf}\left(\frac{\gamma}{k\alpha\sqrt{2}}\right)\right]$$

$$\left(k_J = 4\pi G\rho_0/\alpha^2\right)$$

c) The Jean's swindle isn't necessary for this potential since $\nabla\phi = 0 \Rightarrow \nabla^2\phi = 0$ but this doesn't require ρ to be 0 because of the additional term α^2 //