Homework № 7

Kevin Mack (Worked with Alex DeWitt), Clarkson University 12/01/2017

Problem Statement:

In this assignment the task is create a model of the n-body problem, which is a famous problem in physics. In this case, the system is designated to be celestial bodies who's force of action is the gravitational force exerted by each body on each other. This is a problem that is notoriously difficult to solve analytically (it was actually shown by Poincarè that there is no analytical solution given by algebraic expressions and integrals), and it can also exhibit chaotic behavior. Due to these facts it is often solved with a numerical solution, also making it an appropriate problem to apply equation-free methods in order to extract the dynamics from the data. A database of the movement of celestial bodies would be an ideal place to gather data for equation-free modeling, and is available from NASA and other astronomical societies. In this work no specific system is modeled, however a general case of the n-body problem will be simulated using the ODE solver in Matlab, which will then be compared with equation-free models.

Solution to the N-Body Problem:

The n-body problem can be generalized with an equation of motion that can be applied to each body of the system. When all bodies are considered, the system will provide the motion of each body with the consideration of the gravitational forces from every other body. The equations of motion are given by,

$$m_i \frac{d^2\mathbf{q}_i}{dt^2} = \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{G m_i m_j (\mathbf{q}_j - \mathbf{q}_i)}{\|\mathbf{q}_j - \mathbf{q}_i\|^3} = \frac{\partial U}{\partial \mathbf{q}_i} \tag{1}$$

where $\mathbf{q_i}$ is the position vector of the i^{th} body, m_i is the mass of the i^{th} body, and G is the universal gravitational constant. Using (1) it can be seen that in three spatial dimensions, the number of differential equations used to solve the system is on the order of $6n$ (where n is the number of bodies), which can become increasing computationally expensive for sets of large particles. The ODE's, written in code as **n_body.m** (listing 1), is solved using Matlab's built-in ODE solver **ode113.m**.

Listing 1: Matlab code – n_body.m

```
1    function dx = n_body(~ , x)
2 %GM = 1; % Normalized G*mass
3 GM = 1e2; % Mass*G that "works well" for our initial distances and ↩
      velocities
```

```
%x = reshape(x, [2, 6]);
N = size(x,1)/6;
x = reshape(x, [6, N])'; % reshape from vector to matrix
%x = [x(1:6)'; x(7:end)'];
dx(:,1) = x(:,4); % set velocity in x
dx(:,2) = x(:,5); % set velocity in y
dx(:,3) = x(:,6); % set velocity in z
for i = 1:size(x, 1)
    dx(i,4:6) = 0;
    for j = 1:size(x, 1)
        if( j~= i)
            df = x(j,1:3) - x(i,1:3); % get difference between each pair of particles
            dx(i,4:6) =  dx(i,4:6) + GM*(df)/(df * df.').^(3/2); % compute gravitational effects
        end
    end
end
dx = reshape(dx', [N*6, 1]); % ODE must output a column vector
end
```

In this code, ode113 is used instead of ode45 due to its improvement in accuracy. When using ode45 to solve these equations, the round-off error in the implementation of the Runge-Kutta method was causing the system to lose energy artificially over time. The ode113 method is variable-step, variable order (VSVO) Adams-Bashforth-Moulton solver of orders 1 to 13, and has been noted for its ability to be used in celestial mechanics problems (which is similar, if not equivalent to this problem). Using this solver and the code in **n_body.m**, we generate the micro-scale model that is necessary to perform equation-free modeling.

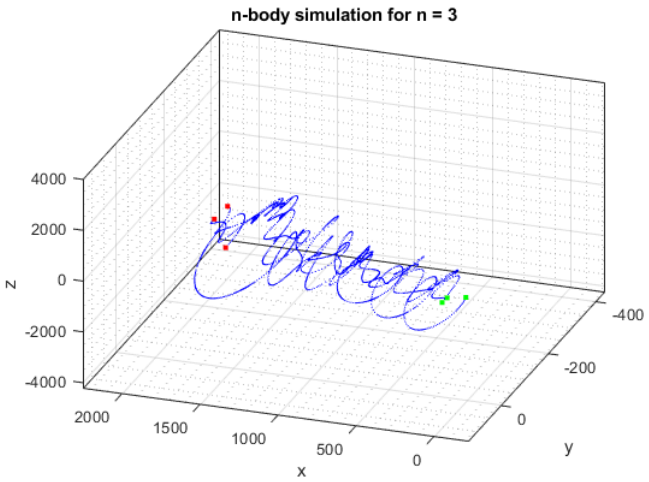
Listing 2: Matlab code – Micro-scale simulation

```
%% n-body problem simulation

N = 50; % number of bodies
pdim = 3; % number of spatial dimensions
vdim = 3; % number of velocity dimensions
vvar = 1e0; % variance for initial velocity of particles
max = 1e2; % max distance for inital positions of particles

positions = -max+(max+max).*rand(N, pdim);
velocities = sqrt(vvar)*randn(N, vdim);
ic = [positions, velocities];
ic = reshape(ic.', [1, N*6]);

% set time parameters
t_start = 0;
```

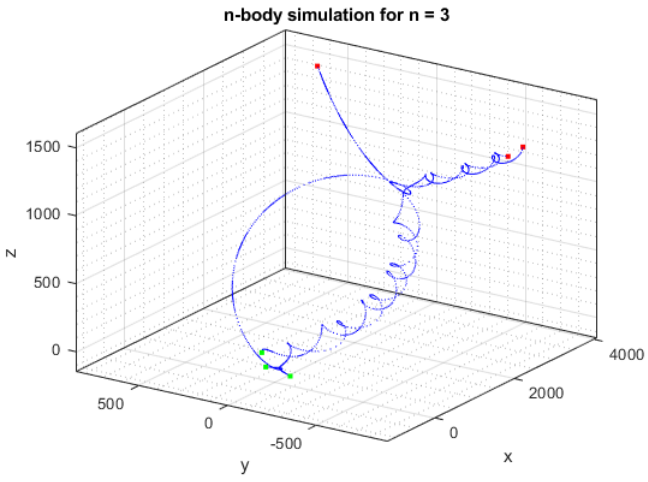
```
16 t_end = 1e4;
17 % t_step = 5e1;
18 t = [t_start t_end];
19
20 tic
21 [T,Y] = ode113(@n_body, t, [ic]); % Solve ODE's
22 toc
23
24 % get output in correct form to plot time-series
25 Y_plot = zeros(pdim+vdim, N, size(T,1));
26 for i = 1:size(T,1)
27     Y_plot(:,:,i) = reshape(Y(i,:).', [pdim+vdim, N]);
28 end
29
30 %% Plot output
31 figure
32 hold all
33 grid on
34 grid minor
35 box on
36
37 for i = 1:size(T,1)
38     if(i ~= 1 && i~= size(T,1))
39         plot3(Y_plot(1,:,i), Y_plot(2,:,i), Y_plot(3,:,i),'b.', '↩
              MarkerSize', 3) % plot particle path with blue marker
40     elseif( i == 1)
41         plot3(Y_plot(1,:,i), Y_plot(2,:,i), Y_plot(3,:,i),'g.', '↩
              MarkerSize', 10) % plot starting point with green marker
42     else
43         plot3(Y_plot(1,:,i), Y_plot(2,:,i), Y_plot(3,:,i),'r.', '↩
              MarkerSize', 10)% plot end point with red marker
44     end
45 end
46 xlabel('x')
47 ylabel('y')
48 zlabel('z')
49 title(['n-body simulation for n = ' num2str(N)])
```

Since the equations cannot be solved analytically, as stated above, a numerical solution in Matlab has been obtained. The solver will compute the position and velocity vectors for each particle in the system. Several different values for the number of particles will be considered, and several graphs will show the output from each solution. In the solver, the initial conditions for position of each particle in the system are generated from a uniform random distribution, while the initial velocities are generated from a Gaussian distribution. Further, each mass is of equal magnitude, and all have been normalized so that $m_i G = 1$ (where m_i is the mass of the i^{th}

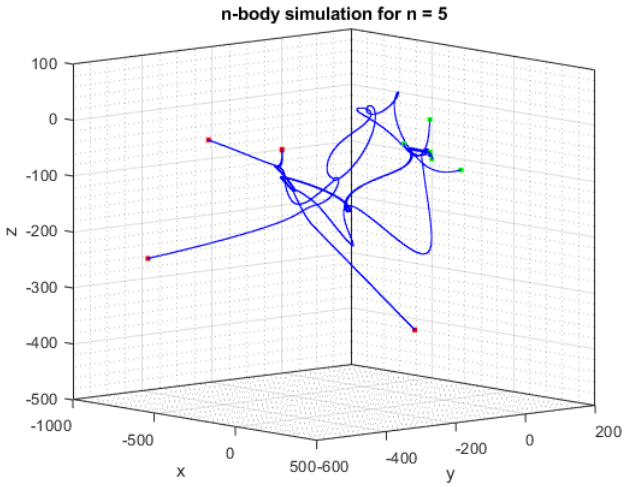
body, G is the universal gravitational constant). This is to limit the complexity of the simulation and keep the computation times as small as possible.



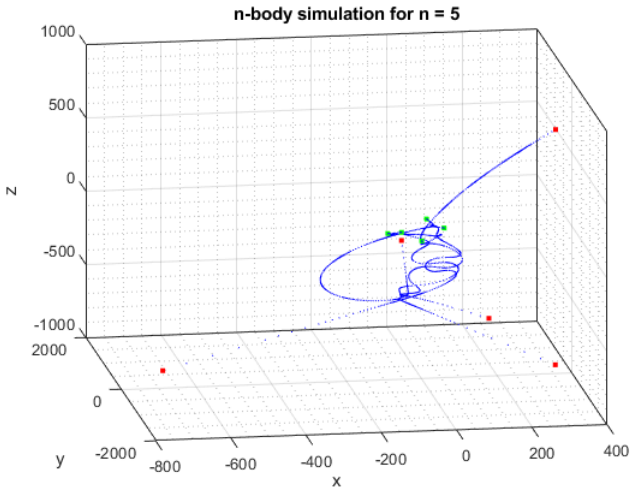
(a) n-body simulation for $n = 3$



(b) n-body simulation for $n = 3$



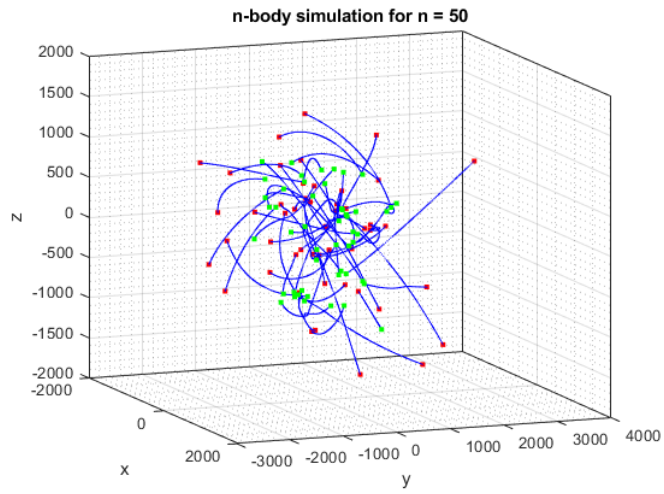
(c) n-body simulation for $n = 5$



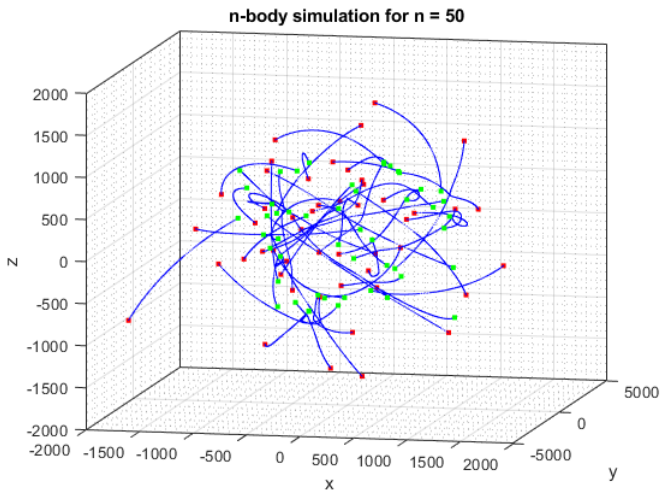
(d) n-body simulation for $n = 5$

Figure 1: Solutions to the n-body problem for n = 3, and n = 5 bodies. Each mass in the system is equal, with only the initial conditions being different. Two trials of each case are shown, and it is clear that the initial conditions play a large role in the manifested dynamics of the system.

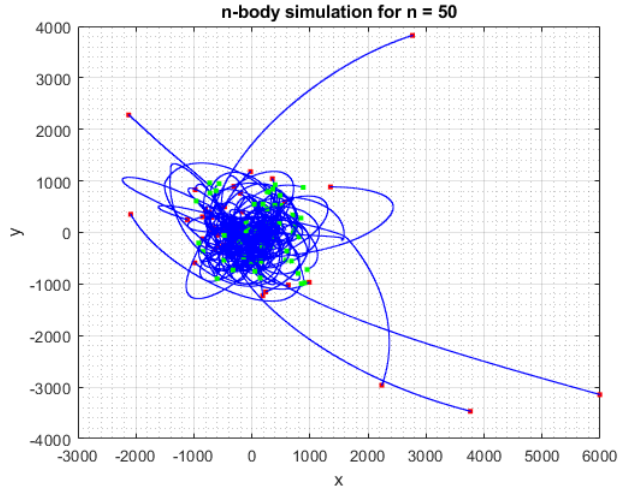
From Figure 1 it appears that the nbody simulator is working correctly for the $n = 3$ and $n = 5$ cases. It is now important to test situations where the number of bodies is significantly higher. Tests for $n = 50$ and $n = 100$ are given in Figures 2 and 3. These tests caused a large rise in computational complexity, this is due to the fact that the number of equations that are solved by the ODE solver in Matlab is on the order of $6n$ (n being the number of bodies). As the



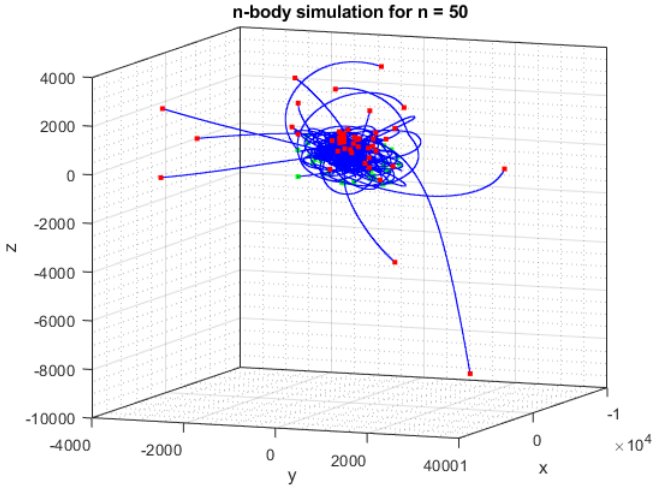
(a) Angle 1, n-body simulation for $n = 50$



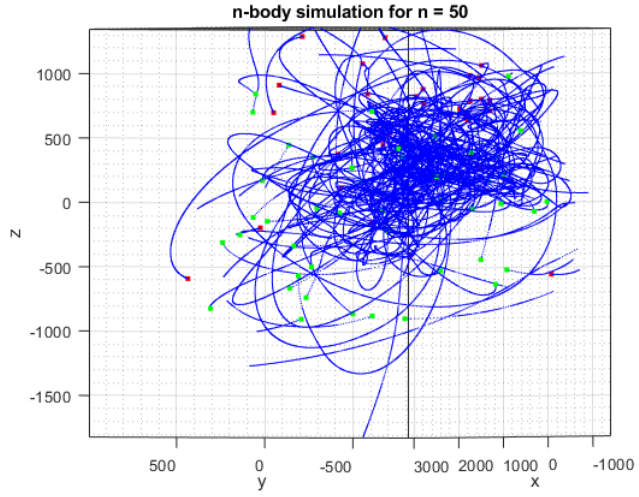
(b) Angle 2, n-body simulation for $n = 50$



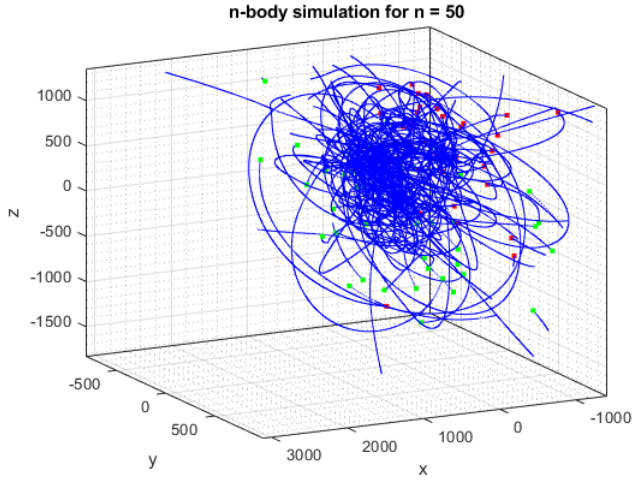
(c) Angle 1, n-body simulation for $n = 50$ (long time)



(d) Angle 2, n-body simulation for $n = 50$ (long time)

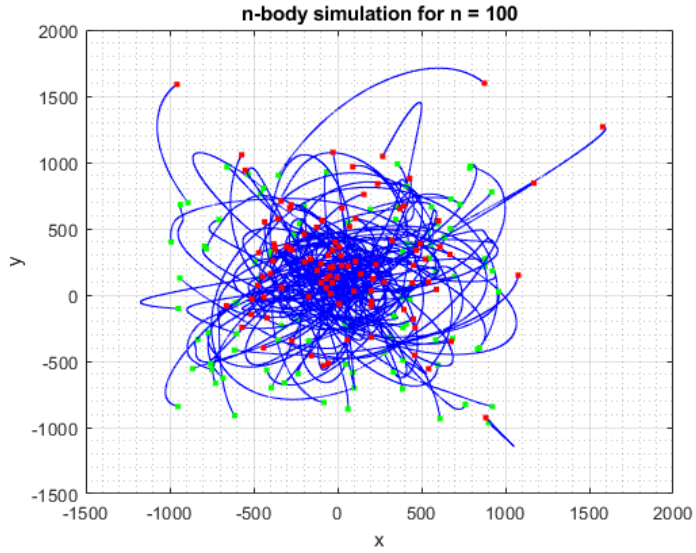


(e) Angle 3, n-body simulation for $n = 50$ (long time)

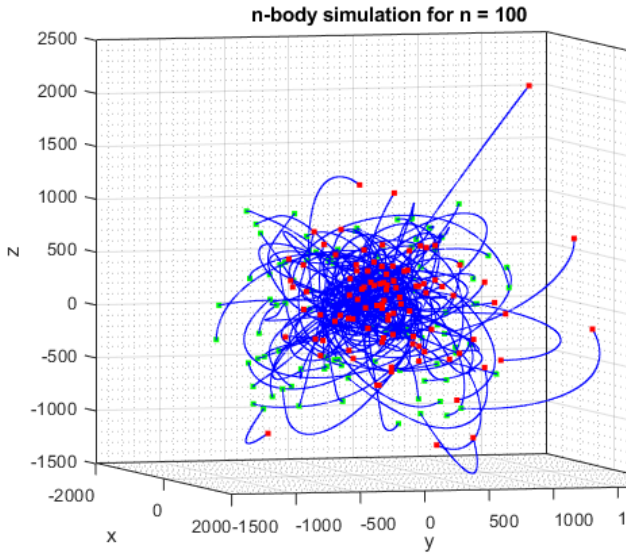


(f) Angle 4, n-body simulation for $n = 50$ (long time)

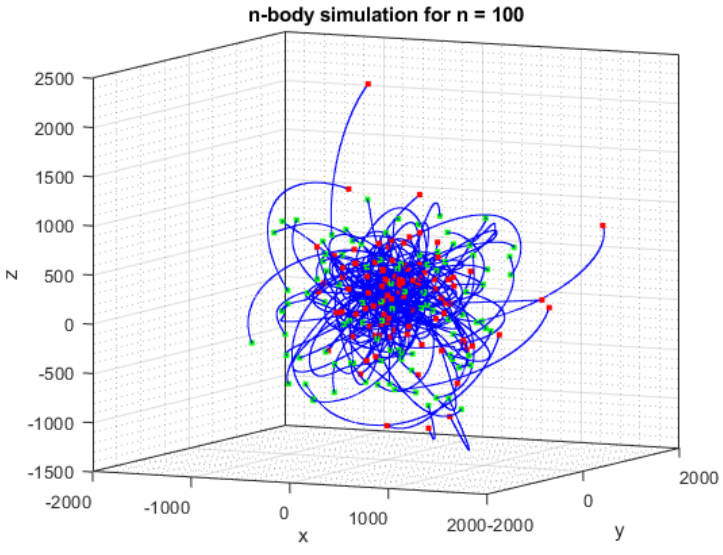
Figure 2: Solutions to the n-body problem for n = 50 bodies. Each mass in the system is equal, with only the initial conditions being different. Again, the starting positions of each mass is labeled with a green dot, with the final position shown as a red dot. Sub-figures (a) and (b) are from the same simulation, with different angles. Sub-figures (c),(d),(e), and (f) go together and show a longer time evolution.



(a) Angle 1, n-body simulation for $n = 100$



(b) Angle 2



(c) Angle 3

Figure 3: The images are similar to the previous n-body simulations, with only the number of particles being different. The system, as in the other figures, is heavily dependent on the initial conditions of particle position and velocity (on the micro-scale).

Equation Free Modeling:

Due to the fact that this model becomes computationally expensive for sets of large particles or over long periods of time, it is appropriate to consider an equation free modeling approach. The equation-free modeling algorithm consists of two main aspects, restricting and lifting. In restriction, a selection of macroscopic variables is made in order to generate a model of the system. The restriction operator converts the microscopic (fine) model into the macroscopic (course) model. The restriction operator generally completes the task of projecting the principal components of the microscopic variables, onto the chosen macroscopic variables (using methods such as SVD/PCA/POD). This is described in (2),

$$\mathbf{U} = \mathcal{M}\mathbf{u}, \tag{2}$$

where $\mathbf{U}$ is the system of variables for the macroscopic behavior, $\mathcal{M}$ is the restriction operator, and $\mathbf{u}$ is the system of variables for the microscopic behavior. It is important to note that $\mathbf{U} \in \mathbb{R}^M$ and $\mathbf{u} \in \mathbb{R}^m$, where generally $M << m$. Assuming that the microscopic system can be adequately modeled by the macroscopic variables $\mathbf{U}$, then the the remaining statistical characteristics at the microscopic scale should be properly approximated by functionals of the those selected for the restriction operation. If we say that the microscopic dynamics can be transformed to a new set of variables, this gives a system of differential equations,

$$\frac{d\mathbf{v}_1}{dt} = g(\mathbf{v}_1, \mathbf{v}_2), \tag{3}$$

where $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2)$ is partitioned into M and $m - M$ components. Then the variable $\mathbf{v}_1$ corresponds to with the macroscopic variable U so that,

$$\frac{d\mathbf{U}}{dt} = \frac{d\mathbf{v}_1}{dt} = g(\mathbf{v}_1, r(\mathbf{v}_1)) = F(\mathbf{U}). \tag{4}$$

Thus, the functions g, r, and F can be constructed systematically (not in closed form) to give the process its equation-free approach.

In this work the cumulative distribution is used as the restriction operator. This portion of the code computes the cumulative distributions of radial distance and radial velocity. The Lifting operation is performed to bring the simulation from a lower-dimensional macroscale simulation to a higher dimensional microscale model. The lifting operator generates initial conditions from the macroscale variables for the individual particles in the simulation. The overall process can be summarized in order by performing a Lift, evolving the microscopic variables in fine-scale time (Evolve), and then Restricting to the to the macroscopic variables for coarse-time evolution. The code for the three steps mentioned previously (Lift, Evolve, Restrict) can be found in listing 3 below. In this case, the lifting operator will be performing Dynamic Mode Decomposition (DMD) analysis (listing 4, **dmd.m**) on the chosen variables. This will also allow us to evolve the system in time.

Listing 3: Matlab code – Lift -> Evolve -> Restrict

```
%Alex DeWitt & Kevin Mack
%
%% n-body problem simulation
clear

N = 50; % number of bodies
pdim = 3; % number of spatial dimensions
vdim = 3; % number of velocity dimensions
vvar = 5e0; % variance for initial velocity of particles
max = 1e3; % max distance for inital positions of particles

positions = -max+(max+max).*rand(N, pdim);
velocities = sqrt(vvar)*randn(N, vdim);
ic = [positions, velocities].';
output_600_t = zeros(10+1,6,N);
output_600_t(1,:,:) = ic;
num_iters = 100;
start = 1;
quarter = floor(num_iters/4);
half = floor(num_iters/2);
three_quarter = floor(3*num_iters/4);

for iterator = 1:num_iters
ic  =  output_600_t(iterator,:,:);
ic = reshape(ic, [1, N*6]);
% set time parameters
t_start = 0;
t_end = 5e2;
t = [t_start:1:t_end];

tic
[T,Y] = ode113(@n_body, t, [ic]); % Solve ODE's
toc
Tc = size(T,1);
% get output in correct form to plot time-series
Y_plot = zeros(pdim+vdim, N, Tc);
for i = 1:Tc
    Y_plot(:,:,i) = reshape(Y(i,:).', [pdim+vdim, N]);
end

Y_r = zeros(Tc,N);
hboxs_r = linspace(0,4000,10);
for i = 1:Tc
Y_r(i,:) = Y_plot(1,:,i).^2+Y_plot(2,:,i).^2+Y_plot(3,:,i).^2;
Y_r(i,:) = sqrt(Y_r(i,:));
r_h(i,:) = hist(Y_r(i,:),hboxs_r);
end
```

```
48
49
50 Y_v = zeros(Tc,N);
51 hboxs_v = 0:1:10;
52 for i = 1:Tc
53 Y_v(i,:) = Y_plot(4,:,i).^2+Y_plot(5,:,i).^2+Y_plot(6,:,i).^2;
54 Y_v(i,:) = sqrt(Y_v(i,:));
55 v_h(i,:) = hist(Y_v(i,:),hboxs_v);
56 end
57
58 % DMD
59 X = [r_h,v_h].';
60 dt = (t_end-t_start)/(Tc-1);
61 t = [T;T(2:100)+T(end)];
62 [~,X_dmd,modes] = dmd(t,X,dt);
63 X_dmd = real(X_dmd);
64
65 %%%%%%% Lifting %%%%%%%%
66 xx = linspace(0,5000,1000);
67 dT = 5000/1000;
68 [pos_pdf] = csaps(hboxs_r,X_dmd(1:10,t(end)),.99,xx); % smooth with spline
69 pos_pdf(pos_pdf < 0) = 0;
70 pos_cdf = cumtrapz(xx,pos_pdf);
71 pos_cdf = pos_cdf./pos_cdf(end);
72
73 pos_r = zeros(N,1);
74 uni = rand(N,1);
75
76 for i = 1:N
77 pos_r(i) = find(pos_cdf>uni(i),1).*dT;
78 end
79
80 theta = rand(N,1)*2*pi;
81 phi   = rand(N,1)*pi;
82
83 oj = zeros(6,N);
84 oj(1,:) = pos_r.*sin(phi).*cos(theta);
85 oj(2,:) = pos_r.*sin(phi).*sin(theta);
86 oj(3,:) = pos_r.*cos(phi);
87
88 cdf_dummy = @cumtrapz;
89 %radial velocity
90 xx = linspace(0,10,1000);
91 dT = 10/1000;
92 [v_pdf] = csaps(hboxs_v,X_dmd(11:21,t(end)),.99,xx);%smooth with spline
93 v_pdf(v_pdf < 0) = 0;
94 v_cdf = cdf_dummy(xx,v_pdf);
```

```
95  v_cdf = v_cdf./v_cdf(end);
96
97  v_r = zeros(N,1);
98  uni = rand(N,1);
99  for i = 1:N
100 v_r(i) = find(v_cdf>uni(i),1).*dT;
101 end
102
103 % plot histograms and graphs
104 if(iterator == start || iterator == quarter || iterator == half || ↩
        iterator == three_quarter || iterator == num_iters)
105     bins_pos = floor(sqrt(size(pos_r, 1)));
106     bins_v = floor(sqrt(size(v_r, 1)));
107     figure
108     hold on
109     histogram(pos_r, bins_pos)
110     title({'Histogram of Radial Distances, iteration ' iterator })
111
112     figure
113     hold on
114     histogram(pos_r, bins_v)
115     title({'Histogram of Radial Velocities, iteration ' iterator })
116
117     figure
118     hold on
119     grid on
120     grid minor
121     box on
122     plot(v_pdf)
123     title({'CDF of Radial Velocity, iteration' iterator})
124 end
125
126 %radial velocity is in the direction as r
127 oj(4,:) = v_r.*sin(phi).*cos(theta);
128 oj(5,:) = v_r.*sin(phi).*sin(theta);
129 oj(6,:) = v_r.*cos(phi);
130
131 %tangential velocity is proportional to r
132 theta = rand(N,1)*2*pi;
133 phi = phi + pi/2;
134
135 %compute circular orbits
136 v_t = sqrt(100*N./pos_r);%
137 oj(4,:) = oj(4,:)' + v_t.*sin(phi).*cos(theta);
138 oj(5,:) = oj(5,:)' + v_t.*sin(phi).*sin(theta);
139 oj(6,:) = oj(6,:)' + v_t.*cos(phi);
140
```

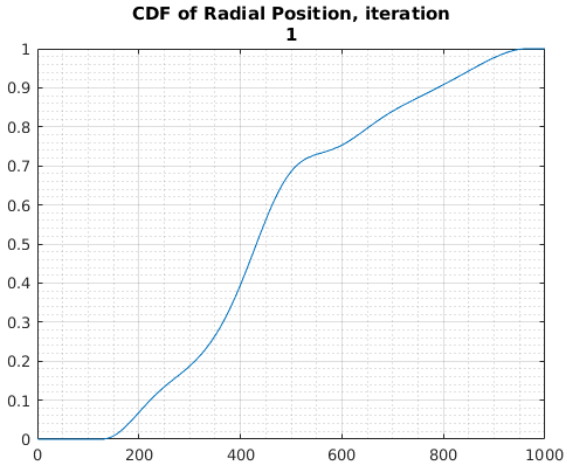
```
%%%%%%%%oj = new Y vector%%%%%%%%%%

output_600_t(iterator+1,:,:)  = oj;
end

markers = {'k.','b.','g.','r.','m.'};
plots = 0;
figure
hold on
% Plot bodies at different points in the simulation
for i=1:iterator
    if(i == start || i == quarter || i == half || i == three_quarter || i ==
        num_iters)
        plot3(squeeze(output_600_t(i,1,:)), squeeze(output_600_t(i,2,:)),
            squeeze(output_600_t(i,3,:)), markers{plots})
        axis([-5000,5000,-5000,5000,-5000,5000]);
    end
end
```

Listing 4: Matlab code – dmd.m

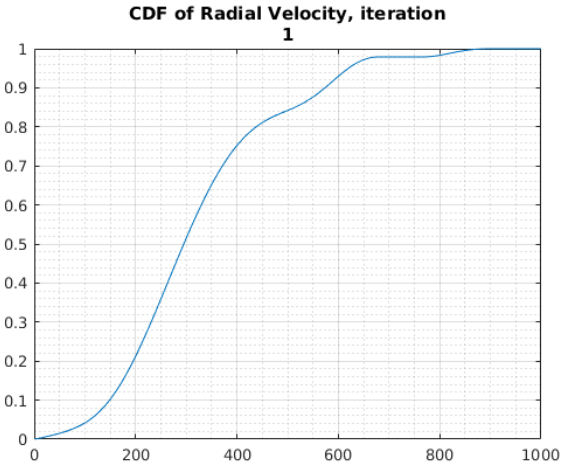
```
function [t,u_dmd,u_modes] = dmd(t,X,dt)
u  = X(:,1);         % create intial conditions
X1 = X(:,1:end-1);
X2 = X(:,2:end);

%perform dmd
[U,Sigma,V] = svd(X1,'econ');
Sigma = Sigma + diag(10^-10*ones(size(Sigma,1),1));
Sigma_inv = diag(1./diag(Sigma));
S =  U'*X2*V*Sigma_inv;
[ev,D] = eig(S);
mu = diag(D);
omega = log(mu)/(dt);
Phi = U*ev;

y0 = pinv(Phi)*u; %pseudo-inverse on initial conditions
u_modes = zeros(size(V,2), length(t));
for iter = 1:length(t)
    u_modes(:,iter) = (y0.*exp(omega*t(iter)));
end
u_dmd = Phi*u_modes; % return the dmd modes
```

In conclusion, the code above provides the macro-evolution of the system, with the aid from simulating small steps in the microscale evolution at regular time intervals. This allows for the

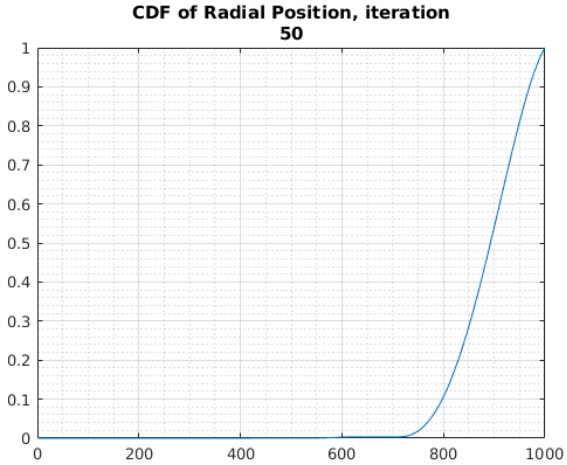
modeling of the behavior of the chosen macro-scale variables (radial distance and radial velocity) over comparatively large time scales to the microscale simulation. Figures 4 and 5 represent some of the histograms created for the restriction operator. As for the initial conditions of the simulation, the starting positions of the particles were computed from a uniform distribution, while the initial velocities were computed from a random normal distribution. For the lifting operator, the new initial conditions for microscale simulation were again generated from random distributions, but this time in compliance with the histograms of the macro-scale variables at that specific time. The overall result is the evolution of the macro-scale variables across coarse time (Figures 4 and 5). The graphs were generated through 100 iterations (meaning 100 instances of micro-scale iterations).



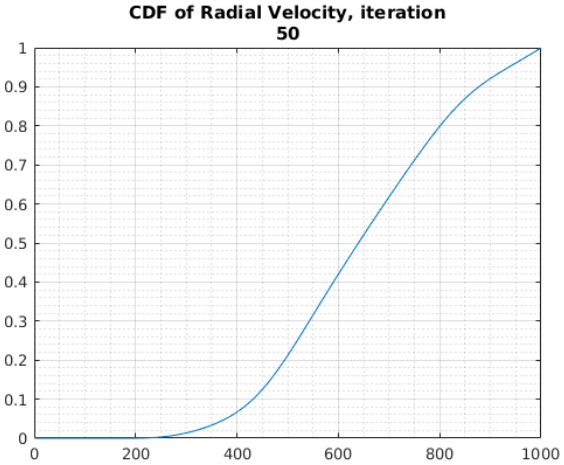
(a) CDF of radial position for iteration 1



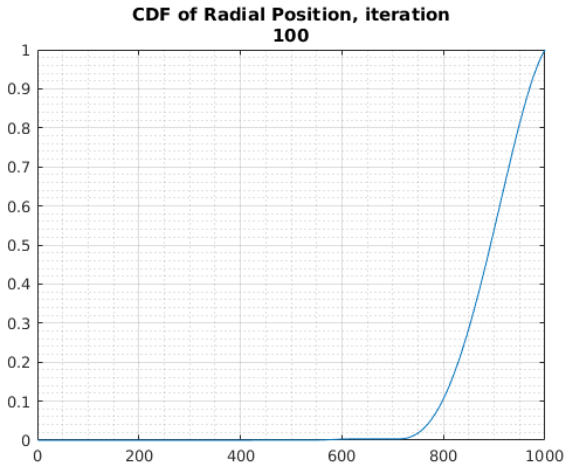
(b) CDF of radial velocity for iteration 1



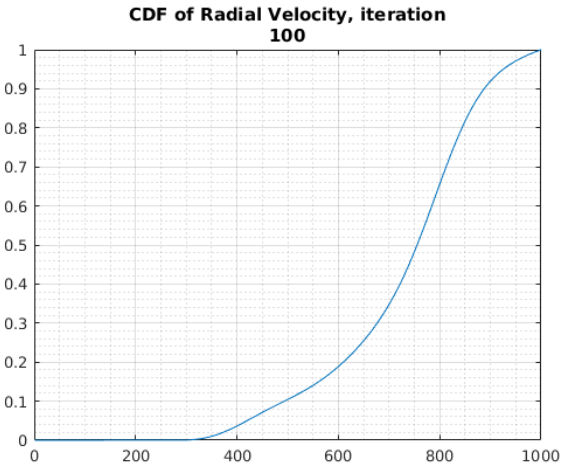
(c) CDF of radial position for iteration 50



(d) CDF of radial velocity for iteration 50

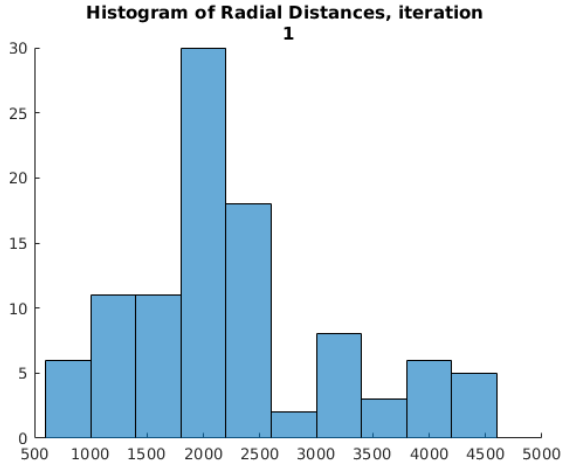


(e) CDF of radial position for iteration 100

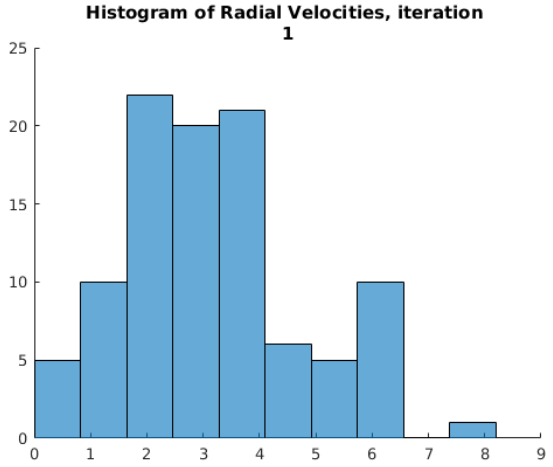


(f) CDF of radial velocity for iteration 100

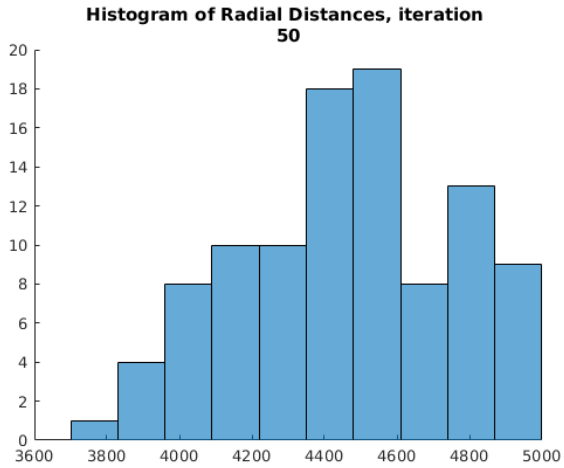
Figure 4: The figures in the left column represent the CDF of the radial position, with the right column being the distribution for the radial velocity for the particles in the simulation. The CDF is used in the restriction phase of the simulation.



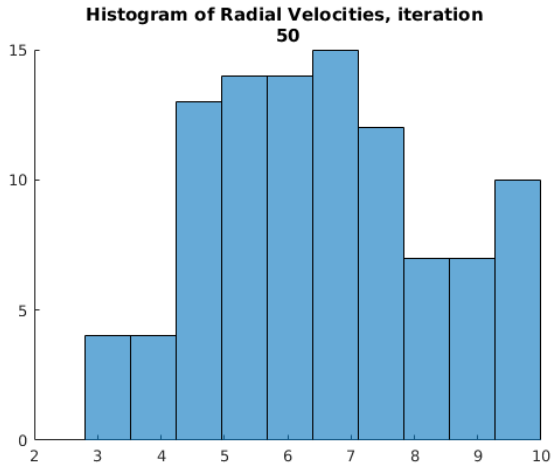
(a) Histogram of radial position for iteration 1



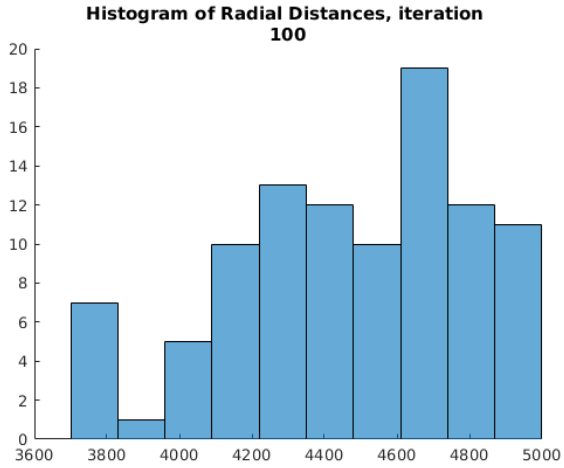
(b) Histogram of radial velocity for iteration 1



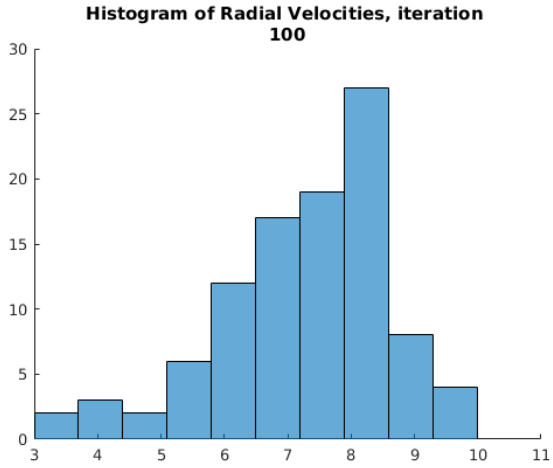
(c) Histogram of radial position for iteration 50



(d) Histogram of radial velocity for iteration 50



(e) Histogram of radial position for iteration 100



(f) Histogram of radial velocity for iteration 100

Figure 5: The histograms in the left column represent the radial distances of the particles at different points in the simulation, withe the histograms on the right representing the radial velocities. These histograms represent the evolution of the macro-scale variables over coarse-scale time.