Term Project:

Evolving Soft Robots (Phase A)

Submitted by: Yaoyao Xu (UNI: yx2725)

Ruoyao Zhao (UNI: rz2535)

Course number: MECSE4510

Instructor: Hod Lipston

Submitted time: 11/08/2021

Grace Hour Used: 1 h

Grace Hour Left: 98 h

**Results**

**1 Breathing cube**

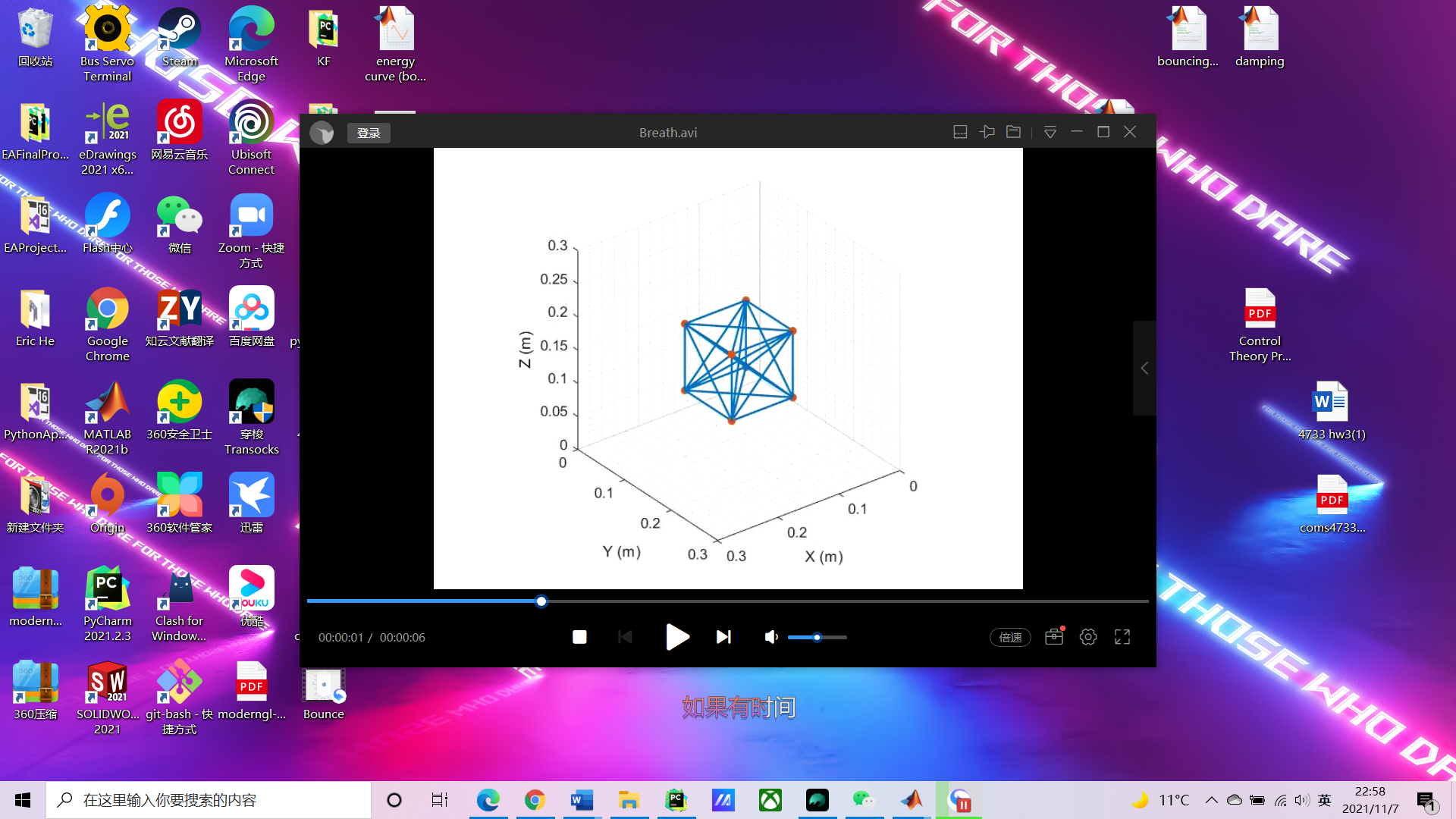


Figure 1 Breathing cube

Video link: <https://youtu.be/gbxYZxEyqx4>

**2 Bouncing cube**

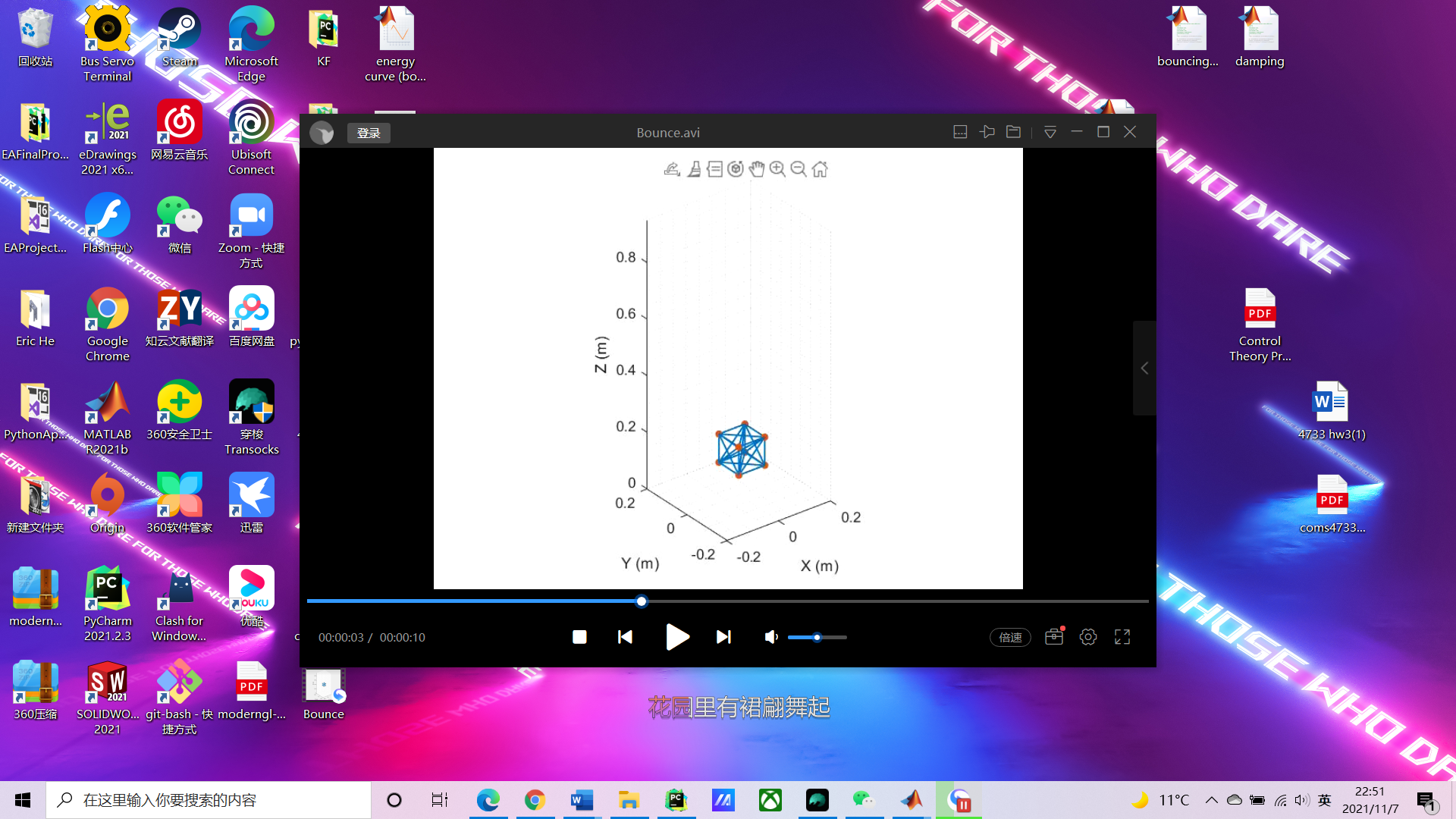


Figure 2 Bouncing cube

Video link: <https://youtu.be/RwjVdspIrbY> (without damping)

<https://youtu.be/38rZLXyWfec> (with damping)

**Methods**

In phase A, we create a physical simulator, which consists of a cube connected using springs and the flat ground. The environmental conditions require the gravity acceleration of 9.81m/s^2 to be applied to the cube for bouncing problem. The time step increment (dt) is set to be 1e-3. The cube has 8 point masses, each has a mass of 0,1kg. The spring constant is 500 N/m.

**Breathing Cube**

We applied the sine function to the vertex of the cube to simulate the “breathing” of the cube. The function is expressed as A\*sin(𝜔*T*), where A is the amplitude of breathing, 𝜔 is the frequency, and in this assignment, we set A to be 5e-4 and 𝜔 to be 10. In every time step, the vertex of the cube will add or minus the distance changed depending on its location. In this case, since the potential energy will not change and the kinetic change will not change much, we just compute the energy of the springs using , which is conservative due to the output plots.

**Bouncing Cube**

Firstly, we apply the same environmental conditions to the cube as breathing, and we set the total time to be 3 seconds. Secondly, a loop is created for calculating the displacements, reaction forces, velocities, accelerations, and energies.

For the bouncing cube, the motion, position and acceleration are calculated through integral, where its position ‘s’ is, velocity ‘v’ is and acceleration is 𝑔 when the cube does not touch the ground. When the cube touches the ground, the lower masses of the cube are supposed to be rebounded without losing energy, and the upper masses of the cube are supposed to keep the original motion. By using the motion of the mass point, the compression or stretch of springs can be found, and the force of spring is separated into x y and z directions. Also, the motion, velocity, and acceleration of springs are calculated through integral.

When we plot the energy, there are some noises. This is because the dt is large and k is large as well. In order to make our model work, we choose dt to be 1e-3 and k to be 500 to balance the performance and efficiency.

By plotting the potential energy (sum of gravity and spring), kinetic energy and summation of all energy, we could tell that the total energy is close to a constant without damping. However, the total energy will decrease when we set the damping ratio to be 0.9. The small increase of total energy is caused by the integral function and large dt, which will introduce more error when velocity is higher. To solve this problem, decreasing the ‘dt’ and increasing the spring constant could highly decrease the error, but the error cannot be eliminated due to the limitation of this model.

**Performance plots**

Energy curves for bouncing cube and “breathing” cube are as shown.

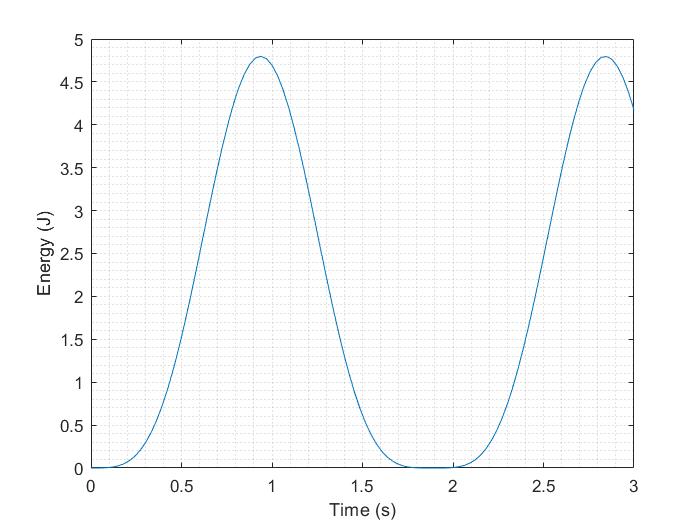


Figure 3 Energy curve for “breathing” cube

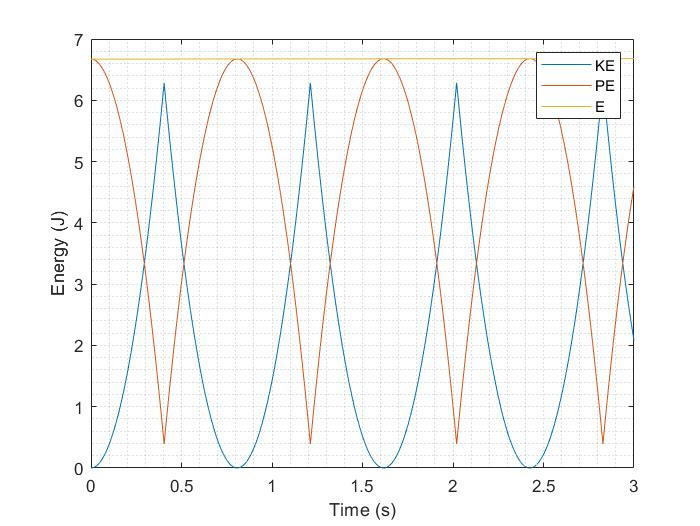


Figure 4 Energy curve for bouncing cube without damping

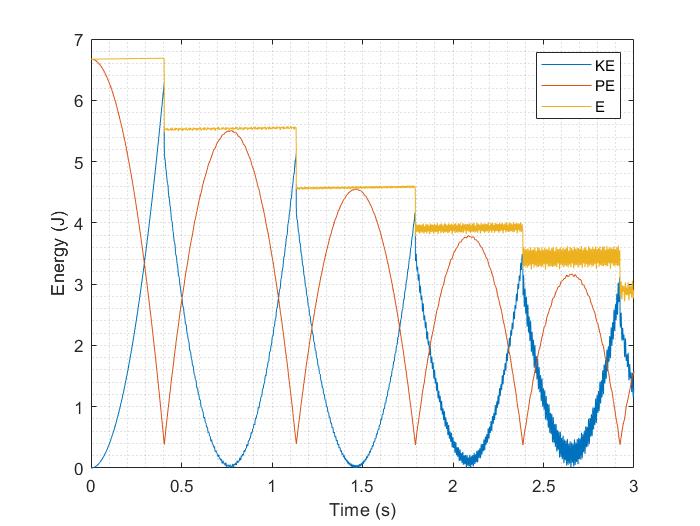


Figure 5 Energy curve for bouncing cube with damping

|  |  |
| --- | --- |
| Model | Number of Evaluations |
| Bouncing with Dampening | 1000 / sec |
| Bouncing without Dampening | 1000/sec |
| Breathing | 300/sec |

Table 1: Number of Evaluations vs. Different Models

**Appendix**

Breathing cube:

clear all

close all

clc

%% Initialization:

% Define global variables:

global g

g = [0,0,-9.81]; % in m/s^2

global dt

dt = .01;

global T;

T = 0;

A = .0005; % amplitude of breathing

w = 10; % frequency of breathing

k=500;

L0 = .1; % original length

Vo = [0 0 0; 0 L0 0; L0 L0 0; L0 0 0; 0 0 L0; 0 L0 L0; L0 L0 L0; L0 0 L0]; % vertices matrix

V = [Vo(:,1)+.1,Vo(:,2)+.1,Vo(:,3)+.1]; % move the position in advantage of good vision

posx = V(:,1);

posy = V(:,2);

posz = V(:,3);

n = 0; % initialize the frame count

while (1)

for j = 1:length(V)

if j == 1 || j == 2 || j == 5 || j == 6 % left side of the cube

posx(j) = posx(j) - A\*sin(w\*T);

else % right side of the cube

posx(j) = posx(j) + A\*sin(w\*T);

end

if j == 1 || j == 4 || j == 5 || j == 8 % front side of the cube

posy(j) = posy(j) - A\*sin(w\*T);

else % back side of the cube

posy(j) = posy(j) + A\*sin(w\*T);

end

if j == 5 || j == 6 || j == 7 || j == 8 % top side of the cube

posz(j) = posz(j) + A\*sin(w\*T);

else % bottom side of the cube

posz(j) = posz(j) - A\*sin(w\*T);

end

end

n = n+1;

dl(n) = abs(posx(1,1)-posx(4,1))-L0; % calculate the displacement between two vertices

V = [posx,posy,posz];

E = [1 2; 2 3; 3 4; 4 1; 5 6; 6 7; 7 8; 8 5; 1 5; 2 6; 3 7; 4 8;...

1 8; 4 5;...

2 5; 1 6;...

3 8; 4 7;...

2 7; 3 6;...

1 3; 2 4;...

5 7; 6 8;...

1 7; 2 8;...

3 5; 4 6];

%% Visualization:

figure(1)

plot3(V(:,1),V(:,2),V(:,3),'.','Markersize',20,'color',[0.8500 0.3250 0.0980]);

%title('Breathing Cube');

xlabel('X (m)');

ylabel('Y (m)');

zlabel('Z (m)');

grid on

grid minor

hold on

for i = 1:size(E,1)

V1 = V(E(i,1),:);

V2 = V(E(i,2),:);

line([V1(1) V2(1)],[V1(2) V2(2)],[V1(3) V2(3)],'LineWidth',1.5);

end

axis equal

axis([0 L0+.2 0 L0+.2 0 L0+.2])

set(gca,'XDir','reverse')

set(gca,'YDir','reverse')

set(gcf,'doublebuffer','on');

T = T+dt;

hold off

M(n) = getframe(gcf);

if n >= 200

break;

end

end

%% Plot Energy:

E = 12\*.5\*k.\*dl.^2 + 12\*.5\*k.\*(sqrt(2)\*dl).^2 + 4\*.5\*k.\*(sqrt(3)\*dl).^2;

figure(2)

plot(linspace(0,3,length(dl)),E);

grid minor

%title('Energy Plot of Breathing Cube');

xlabel('Time (s)');

ylabel('Energy (J)');

%% Generate video:

video = VideoWriter('Breath.avi');

open(video);

writeVideo(video,M);

close(video);

Bouncing cube:

clear all

close all

clc

%% Initialization (All the components of this code are the same as bouncing part, except for adding dampening):

T = 3;

dt = 1e-3;

vz=zeros(1,T/dt);

vdz=zeros(1,T/dt);

vx=zeros(1,T/dt);

vy=zeros(1,T/dt);

h=zeros(1,T/dt);

s=zeros(1,T/dt);

dx=zeros(1,T/dt);

dy=zeros(1,T/dt);

dz=zeros(1,T/dt);

g=-9.81;

t=0;

l=0.1;

h(1)=.8;

s(1)=.8;

k=500;

m=0.1;

az=g;

ax=0;

ay=0;

adz=0;

damp = .9; % This is dampening coefficient

%% Reaction calculation:

for i =2 : (T/dt)

s(i)=s(i-1)+vz(i-1)\*dt;

h(i)=s(i);

vz(i)=az\*dt+vz(i-1);

dx(i)=dx(i-1)+2\*vx(i-1)\*dt;

dy(i)=dy(i-1)+2\*vy(i-1)\*dt;

dz(i)=dz(i-1)+2\*vdz(i-1)\*dt;

ly=l-dy(i);

lx=l-dx(i);

lz=l-dz(i);

fx=-k\*dx(i);

fy=-k\*dy(i);

fz=-k\*dz(i);

fxy=-k\*(l\*2^0.5-(ly^2+lx^2)^0.5);

fxz=-k\*(l\*2^0.5-(lz^2+lx^2)^0.5);

fyz=-k\*(l\*2^0.5-(ly^2+lz^2)^0.5);

fxyz=-k\*(l\*3^0.5-(ly^2+lz^2+lx^2)^0.5);

ax=fx/m\*4+fxy\*lx/(lx^2+ly^2)^0.5/m\*4+fxz\*lx/(lx^2+lz^2)^0.5/m\*4+fxyz\*lx/(lx^2+ly^2+lz^2)^0.5/m\*4;

ay=fy/m\*4+fxy\*ly/(lx^2+ly^2)^0.5/m\*4+fyz\*ly/(ly^2+lz^2)^0.5/m\*4+fxyz\*ly/(lx^2+ly^2+lz^2)^0.5/m\*4;

adz=fz/m\*4+fxz\*lz/(lz^2+lx^2)^0.5/m\*4+fyz\*lz/(lz^2+ly^2)^0.5/m\*4+fxyz\*lz/(lx^2+ly^2+lz^2)^0.5/m\*4;

vx(i)=ax\*dt+vx(i-1);

vy(i)=ay\*dt+vy(i-1);

vdz(i)=adz\*dt+vdz(i-1);

if s(i)<=0

h(i)=0;

dz(i)=dz(i-1)-s(i);

s(i)=0;

vz(i)= (((0.5\*vz(i-1)^2+(-g\*s(i-1))/2)\*2-(g\*s(i)/2))^0.5)\*damp;

else

az=g ;

end

Ex=.5\*k\*dx(i)^2;

Ey=.5\*k\*dy(i)^2;

Ez=.5\*k\*dz(i)^2;

Exy=.5\*k\*(l\*2^0.5-(ly^2+lx^2)^0.5)^2;

Exz=.5\*k\*(l\*2^0.5-(lz^2+lx^2)^0.5)^2;

Eyz=.5\*k\*(l\*2^0.5-(ly^2+lz^2)^0.5)^2;

Exyz=.5\*k\*(l\*3^0.5-(ly^2+lz^2+lx^2)^0.5)^2;

Espring(i) = (Ex+Ey+Ez+Exy+Exz+Eyz+Exyz)\*4;

t=t+dt;

end

%% Visualization:

E=[ 1 2;1 3;1 4;1 5;

1 6;1 7;1 8;2 3;

2 4;2 5;2 6;2 7;

2 8;3 4;3 5;3 6;

3 7;3 8;4 5;4 6;

4 7;4 8;5 6;5 7;

5 8;6 7;6 8;7 8];

n = 0;

for loop=1: T/dt/300: T/dt

V=[-dx(loop)/2 -dy(loop)/2 h(loop);

-dx(loop)/2 l+dy(loop)/2 h(loop);

l+dx(loop)/2 l+dy(loop)/2 h(loop);

l+dx(loop)/2 -dy(loop)/2 h(loop);

-dx(loop)/2 -dy(loop)/2 h(loop)+l+dz(loop);

-dx(loop)/2 l+dy(loop)/2 h(loop)+l+dz(loop);

l+dx(loop)/2 l+dy(loop)/2 h(loop)+l+dz(loop);

l+dx(loop)/2 -dy(loop)/2 h(loop)+l+dz(loop);];

figure(1)

plot3(V(:,1),V(:,2),V(:,3),'.','Markersize',18,'color',[0.8500 0.3250 0.0980]);

hold on;

for i=1:size(E,1)

V1=V(E(i,1),:);

V2=V(E(i,2),:);

line([V1(1) V2(1)],[V1(2) V2(2)],[V1(3) V2(3)],'LineWidth',1.2);

end

axis equal

axis([-0.2 0.2 -0.2 0.2 0 .95])

grid minor

ylabel('Y (m)')

xlabel('X (m)')

zlabel('Z (m)')

%title('Bouncing Cube w/ Dampening Coefficient = 0.9')

hold off

n = n+1;

M(n)=getframe(gcf);

end

vtotal = sqrt(vx.^2 + vy.^2 + vz.^2 + vdz.^2);

KE = .5\*8\*m.\*vtotal.^2;

PE = -(8\*m\*g.\*s + 4\*m\*g.\*(l-dz));

E = KE + PE + Espring;

figure(2)

plot(linspace(0,3,length(h)),KE);

hold on

grid minor

plot(linspace(0,3,length(h)),PE);

plot(linspace(0,3,length(h)),E);

legend('KE','PE','E');

%title('Energy Plot w/ Dampening Coefficient = 0.9');

ylabel('Energy (J)');

xlabel('Time (s)');

%% Generate video:

video = VideoWriter('Bounce\_Damp.avi');

open(video);

writeVideo(video,M);

close(video);