Special Topics: Coin Project

Kornkit Kitsomsub

November 26th 2019

Abstract

This project aims to simulate a spinning coin on the ground with some initial angular momentum. The coin is treated as a rigid body. The ground is modeled a system of springs, where any point below the ground will experience a force in the z direction. The ground will also have a coefficient of sliding friction. This project studies the coin with angular momentum perpendicular to the face of the coin as well as on one of the vectors on the face of the coin.

Introduction

The motivation for this project is to simulate a spinning coin on a table top. The idea came from when I was younger, where I believed that the tilting of the coin might have an effect on the distribution of outcomes- heads vs tails- ever slightly in my favor -something that will not be tested for in this project but is definitely an idea moving forward.

Physics of Rigid Body Motion

The coin, being a rigid body, means that it is an object where there are little to no deformities. Thus, we can describe the motion of a rigid body by describing all of the points of the coin relative to the center of mass of the coin,

$$\mathbf{X}_{cm} = \frac{1}{M_{total}} \sum_{i=1}^{n} m_i \mathbf{x}_i$$

Where M_{total} is the mass of the coin,and m_i and x_i are the mass and position of each point of the coin, respectively. We also have that:

$$\mathbf{U}_{cm} = \frac{\mathbf{X}_{cm}}{dt}$$

where $\mathbf{U}_{cm}$ is the velocity of the center of mass.

Being a spinning object, the coin experiences a angular momentum,$\mathbf{L}$. With the Moment of Inertia Tensor, $\mathbf{I}$, known, we can calculate the angular velocity, ω, as follows:

$$\mathbf{L} = \mathbf{I}\omega$$

where $\mathbf{I}$ is calculated as follows:

$$\mathbf{I} = \sum m_i(||\tilde{\mathbf{X}}^2||E - \tilde{\mathbf{X}}^T\tilde{\mathbf{X}})$$

where $\tilde{\mathbf{X}}$ is the relative position of each point to the center of mass and E is the 3x3 identity matrix.

Physics of the Ground

Recall that a spring produces a force that is opposite and proportional to the direction of displacement of a point (Hooke's Law). In other words:

$$F_{spring} = -kx$$

where k is the spring constant, and x is the displacement.

Furthermore, the ground exerts a frictional force. This force is opposite to the direction of velocity and is proportional to the normal force:

$$F_{friction} = -\mu F_{normal}$$

Where μ is the coefficient of friction.

For further notes on the derivation of concepts in Physics, refer to Professor Peskin's notes on Dynamics of Rigid Body Motion, and Interaction of Dynamic Structures with the Ground: Non-Penetration and Sliding Frictional Forces, found here:

https://math.nyu.edu/faculty/peskin/modsim_lecture_notes/index.html

Initial Setup

Before we start the time loop, the coin is first constructed on the YZ plane using the following code:

```
% Generating points for the ball
for num = 1:num_points
    theta = num * 2 * pi / num_points;
    X(num,:) = [0,radius*cos(theta),radius*sin(theta)];
    M(num) =  M_total / (num_points);
end
```

So, $\mathbf{X}$ is a $n \times 3$ matrix where n is the number of points used in the circle, which is arbitrarily set to 120 points. After constructing the coin, it is then rotated by some angle,θ, along the Y axis. This is done by multiplying a rotation matrix,

$$R(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

to $\mathbf{X}$. Then, we take the height of the coin, and shift it up by that height so that the coin starts in above the ground, which lies on the XY plane with $z = 0$. The shift in height is $r\cos(\theta)$, where r is the radius of the coin.

Time Loop

The State of System, i.e the objects that we need to keep track of in each time step, includes $\mathbf{X}_{cm}, \mathbf{U}_{cm}, \mathbf{X}, \mathbf{U}, \mathbf{L}$. In order to calculate these quantities, we do the following in each time step in order:

1. Using $\tilde{X}$ from the previous time step, we calculate the Moment of Inertia tensor, $\mathbf{I}$.

   This is done using the following snippet of code:

   ```
   % Compute I
   I = zeros(3, 3);
   for k = 1:num_points
       I = I + M(k).*( (norm(Xrel(k,:))^2).*eye(3) - Xrel(k,:)'*Xrel(k,:) );
   end
   ```

2. With the Moment of Inertia Tensor, we can then update ω using the fact that $\omega = \frac{\mathbf{I}}{\mathbf{L}}$.

3. Update $\tilde{X}$ using ω, which will be used to later to update $\mathbf{X}$ of the State of the system.

4. Update $\tilde{U}$ using $\tilde{X}$, which will be used to update $\mathbf{U}$

5. Update $\mathbf{X}_{cm}$ from $\mathbf{U}_{cm}$ from the previous the time step using the fact that $\mathbf{X}_{cm}^{n+1} = \mathbf{X}_{cm}^{n} + dt * \mathbf{U}_{cm}^{n}$, where n denotes the nth time step.

6. Calculate Forces - Normal Force, Frictional force. (Refer to next section on Frictional Forces).

7. Update $\mathbf{U}_{cm}$ using the fact that $\mathbf{U}_{cm}^{n+1} = \mathbf{U}_{cm}^{n} + dt * F/M_{total}$.

8. Update $\mathbf{X}$, using the fact that $\mathbf{X} = \tilde{X} + \mathbf{X}_{cm}$

9. Update $\mathbf{U}$, using the fact that $\mathbf{U} = \tilde{U} + \mathbf{U}_{cm}$

10. Calculate Net Torque. This is done by taking the cross product, $\tilde{X} \times \mathbf{F}_{point}$, and then summing up the values in MatLab. We obtain a 3×1 vector.

11. Update the angular momentum, L, using the fact that $L = L + dt * sum(\tilde{X} \times \mathbf{F}_{point})$

Frictional Force

As stated earlier, we know from physics that $F_{friction} = -\mu F_{normal}$. This equation comes from a 2 dimensional interpretation. However, in our model, we are working in 3D. For this to work in 3D, we know that the force of Friction is parallel to the direction of the tangent velocity vector,$\mathbf{U}_{tan}$, thus, we will need to calculate that at each time step as well.

Our object is a rigid body built with multiple point masses. To calculate the normal force, F_{normal}, we first look at the object at each time step and see if there are any points below the ground. If it is, then the ground is exerting a normal force on said point.

The way in which this is set up in MatLab is with the following snippet of code:

```
F_normal_point = ((X(:,3)<0).*(S*(-X(:,3))-D*U(:,3)));
F_normal_point = (F_normal_point > 0).*F_normal_point;
```

In the first line, the code $((X(:,3) < 0)$ produces $n \times 1$ matrix, which takes a value of 1 if the condition is true, i.e if the z component of the ith point is below the ground, and 0 if it is not. Now, if it is below the ground, there are two forces acting on said point- we have the force of the ground, which is modeled as a spring and therefore the force produced is S multiplied by the distance of penetration, i.e the zth component of that point and there is a force that is damping the motion, which is $D * U(:,3)$, where U is the velocity vector. Notice that the spring force is positive because we are taking $-X(:,3)$, and $X(:,3)$ is already negative, so these two forces oppose each other.

In this model, the way we treat friction is that we take the sum of positive values of the normal forces on each point. The normal forces on each can be negative when $D * U(:,3) > S * -(X(:,3))$. This is seen in the second line of code.

Now, we need only to calculate the tangent velocity vector. This is done through the following snippet of code:

```
Utan = [U(:,1),U(:,2),zeros(num_points,1)];
Utan_norm = (vecnorm(Utan')' + 10^-10)*[1,1,1];
Utan_unit = Utan./Utan_norm;
```

We take the projection of $\mathbf{U}$ onto the floor, which in this case is the XY plane. This is done easily by just taking X and Y components of U and setting

the Z component to zero. We then find the norm of **U**, and then divide **U** by its norm to get a unit vector in the direction of **U**. Notice that in the normalizing step, we also added a small constant, 10^{-10}. This is done to work around the case when **U** is a zero vector, which occurs at least once at the very beginning of the program.

We are then left with our frictional force as follows:

```
F_friction = - mu*F_normal_point.*Utan_unit;
```

1 Independent Variables

For this project, it is designed so that there are a few parameters to play around with. The following parameters can be played around with:

- Damping Constant
- Coefficient of Friction
- Angle of Tilt
- Magnitude of Initial Angular Momentum
- Direction of Angular Momentum

Results

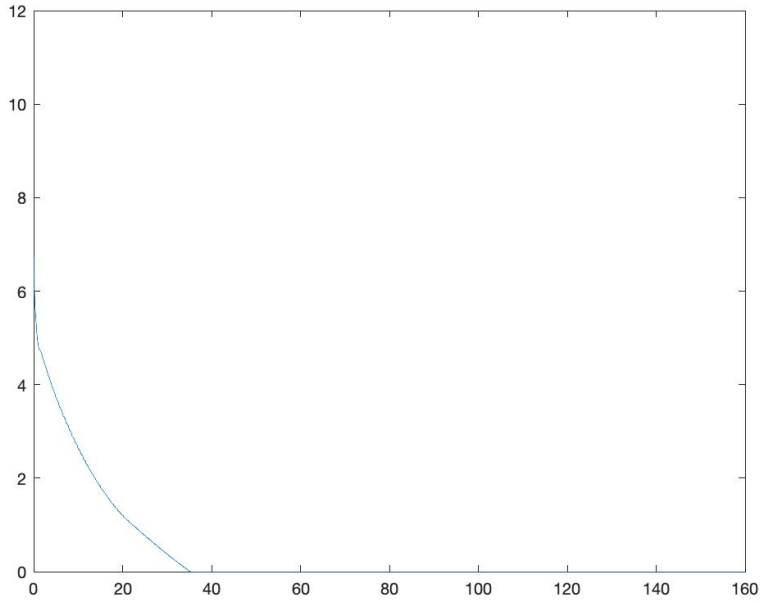
The way in which the results are presented is that we will look at different trials where one thing is one parameter is changed at a time. The title of the subsection should detail what the independent variables are set as.

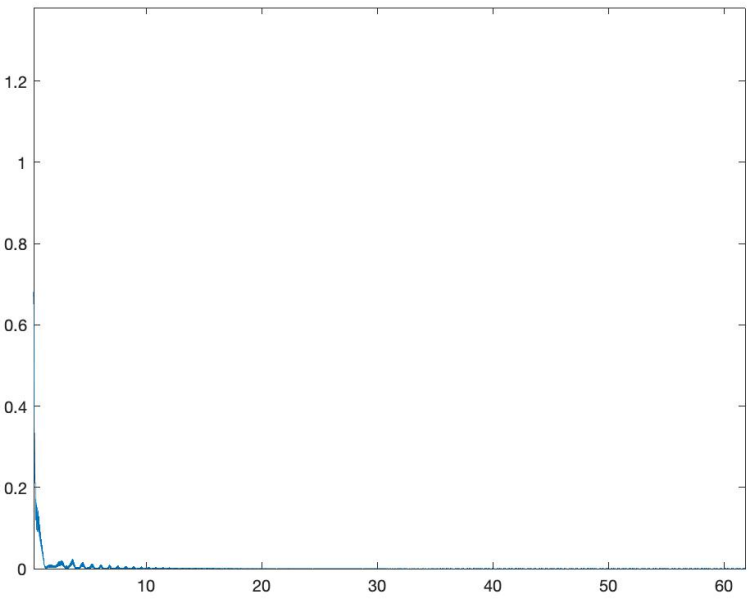
In each trial, the outputs from the program include the lowest contact point of the coin with the ground, and the energy plot. Provided as well are url links to the videos of each coin spinning.

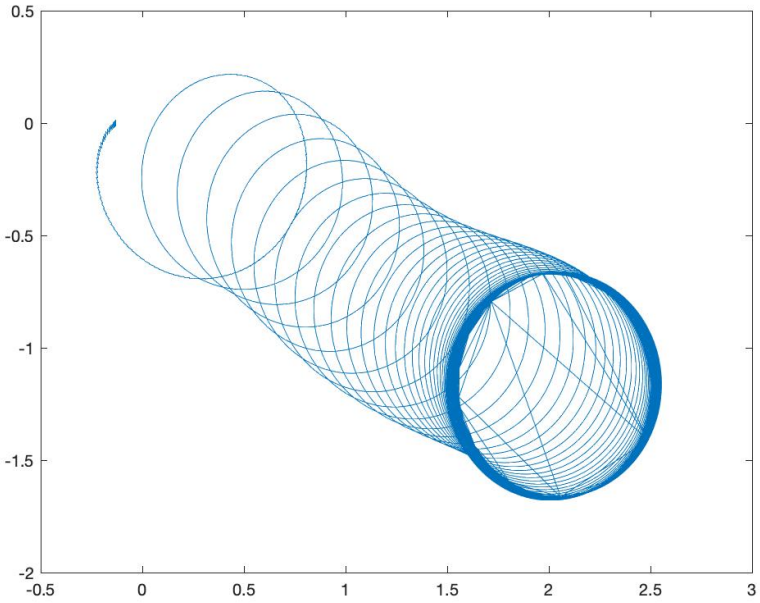
Recall that there are two orientations for Angular Momentum: Perpendicular and Parallel to the face of the coin. This will be noted as rolling ans spinning, respectively. The rolling coins will also have a graph showing the magnitude of the tangent velocity vector with respect to the ground.

Remark: The graphs will appear in the following order- Energy Plot, Utan Plot (if it exists), and XY plot.

Trial 1 - Rolling, angle = 15, D=0.5,mu = 0.3

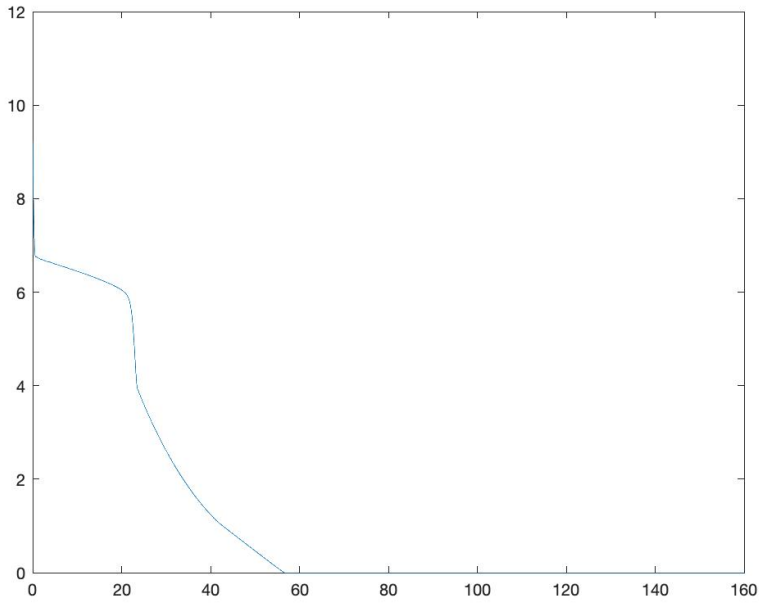


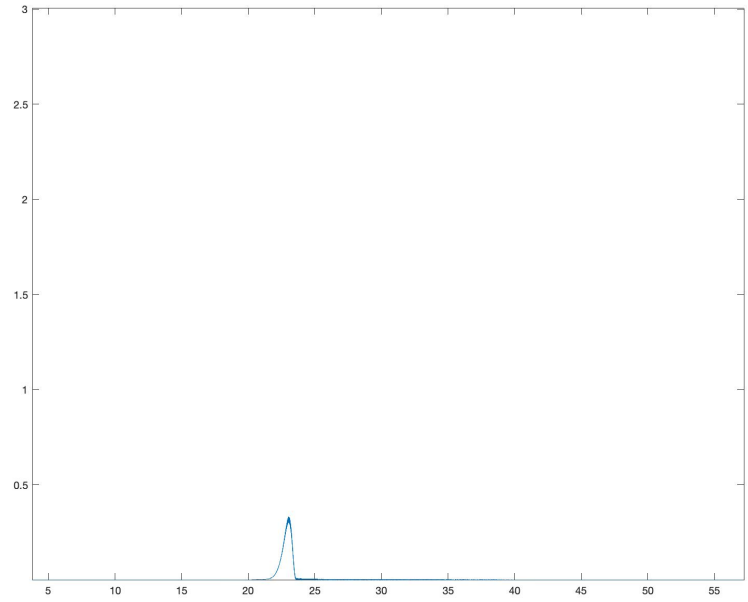


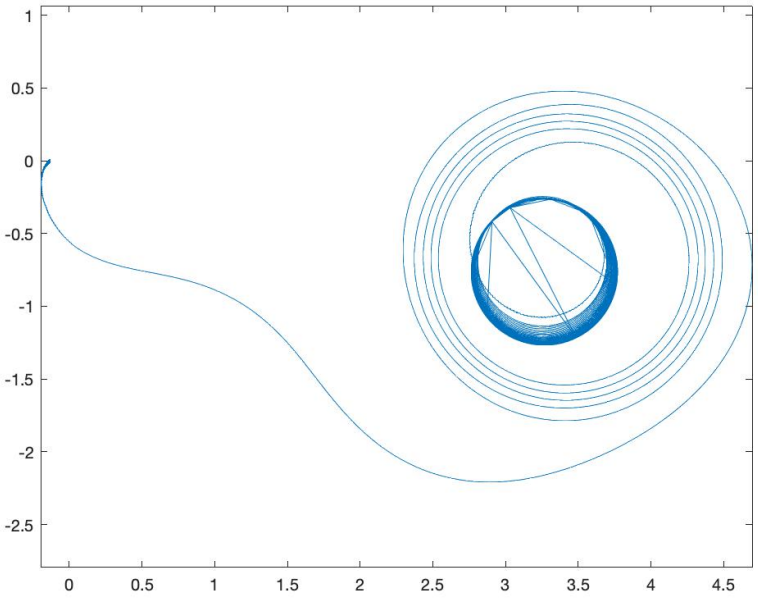


https://youtu.be/B05eZIHi6LA

Trial 2 - Rolling, angle = 15, D=0.5,mu = 0.3, 1.5x Angular Momentum

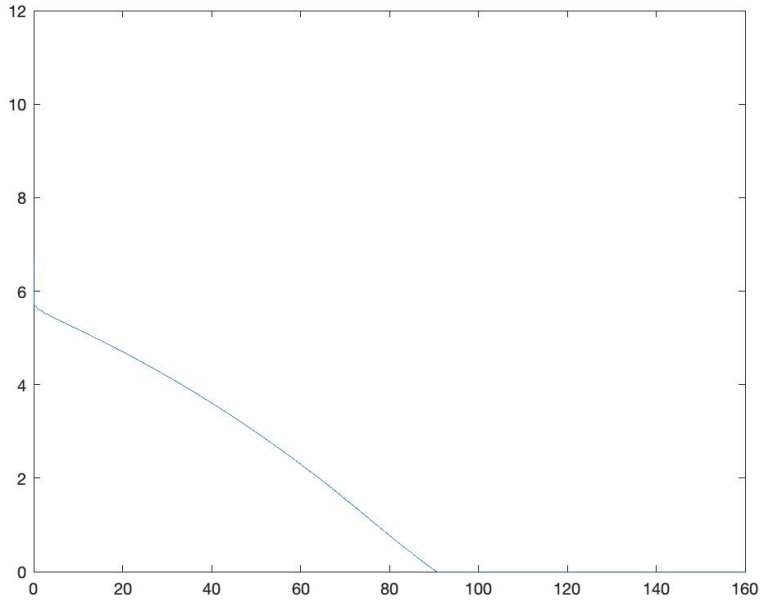


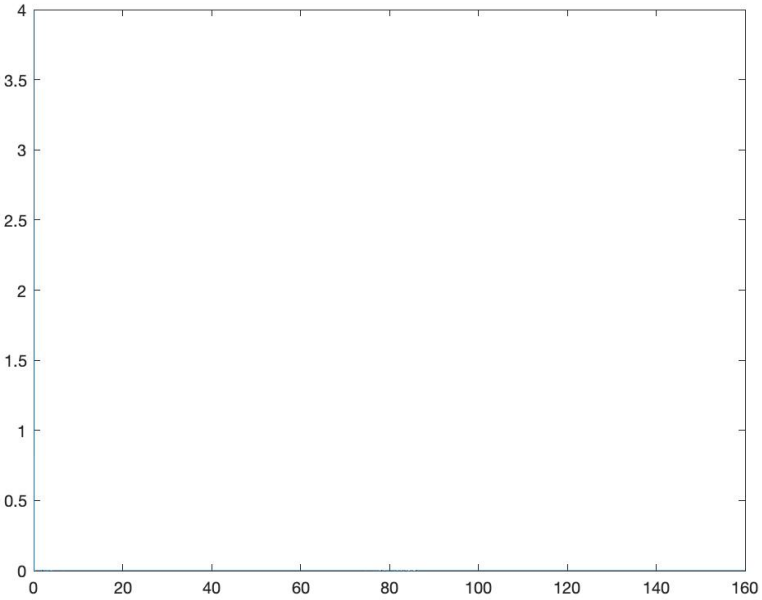


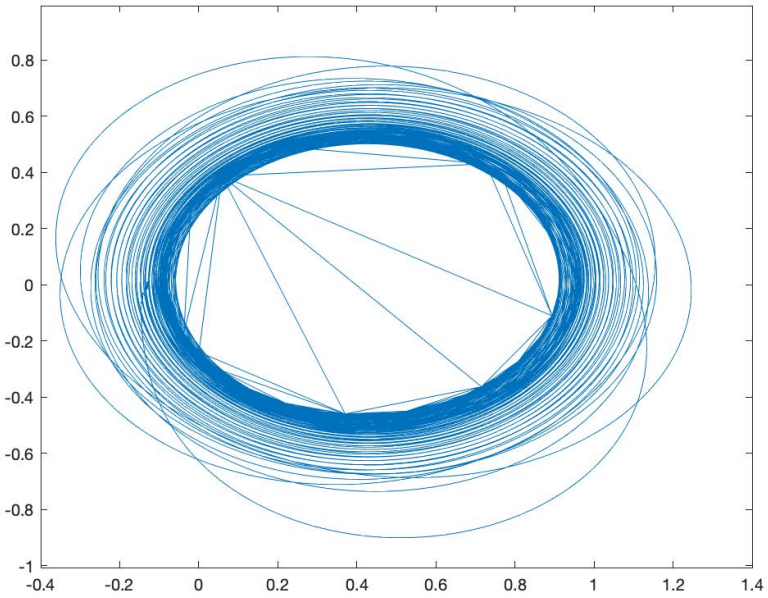


https://youtu.be/m8pXtHMcwwA

Trial 3 - Rolling, angle = 15, D=0.5,mu = 0.8

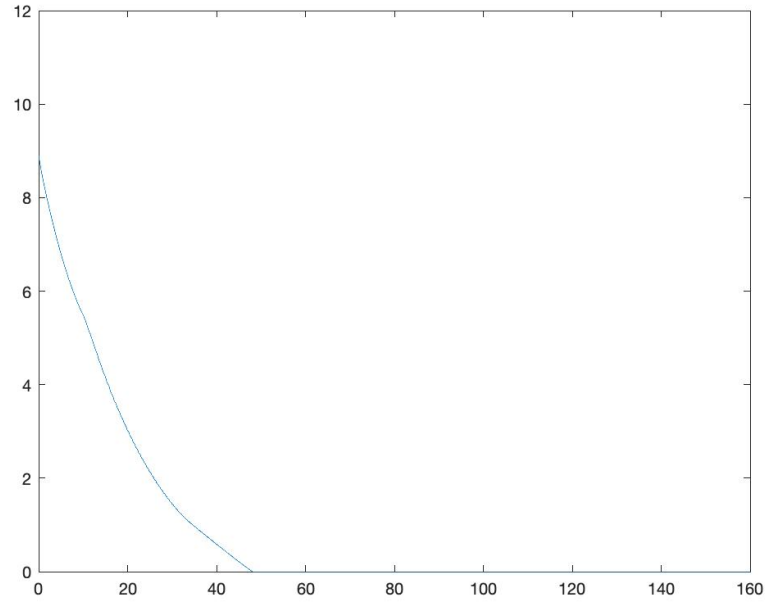


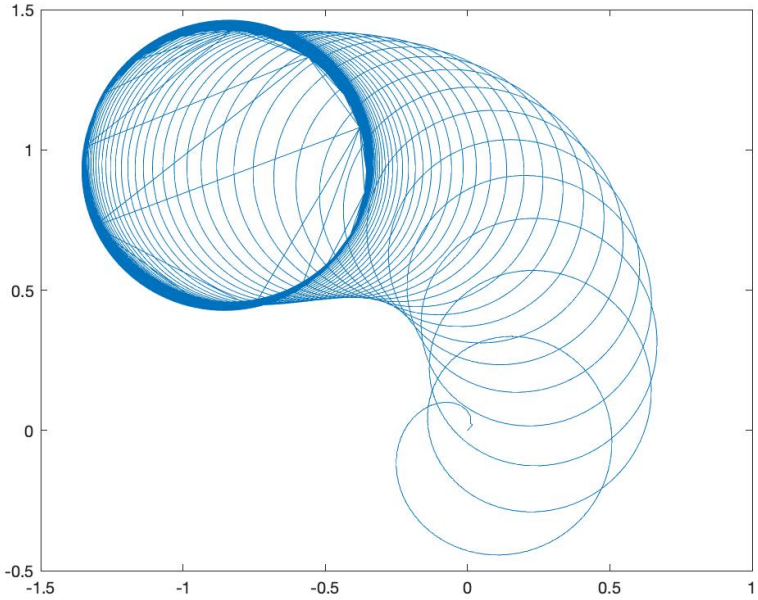




https://youtu.be/Cf96qEt7kLA

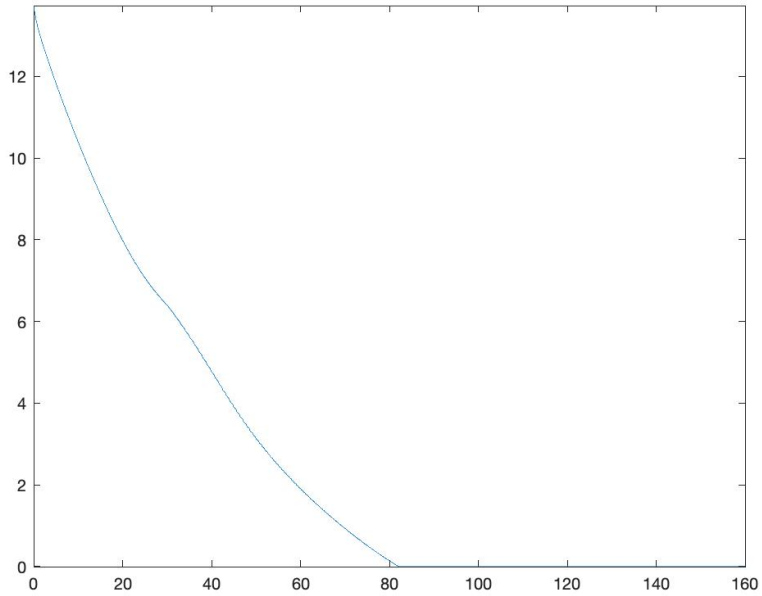
Trial 4 - Spinning, angle = 0, D=0.5,mu = 0.3

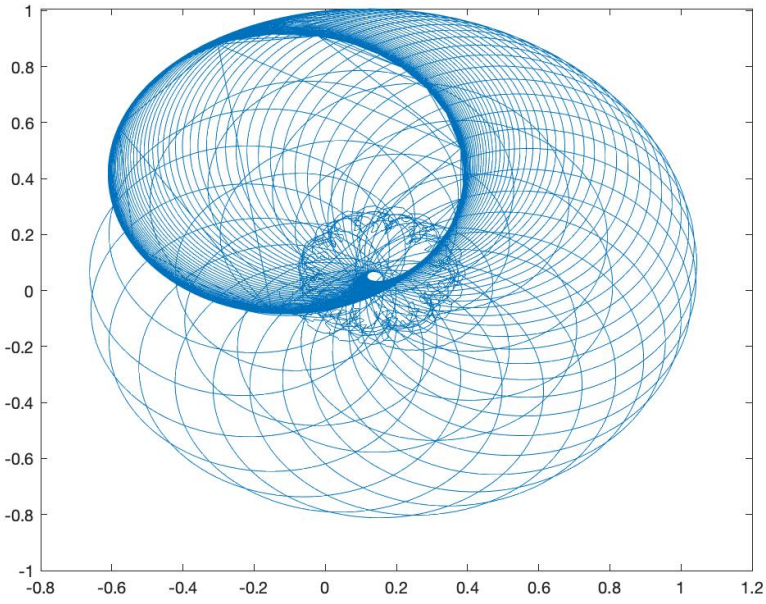




https://youtu.be/nHiPKwAfB9w

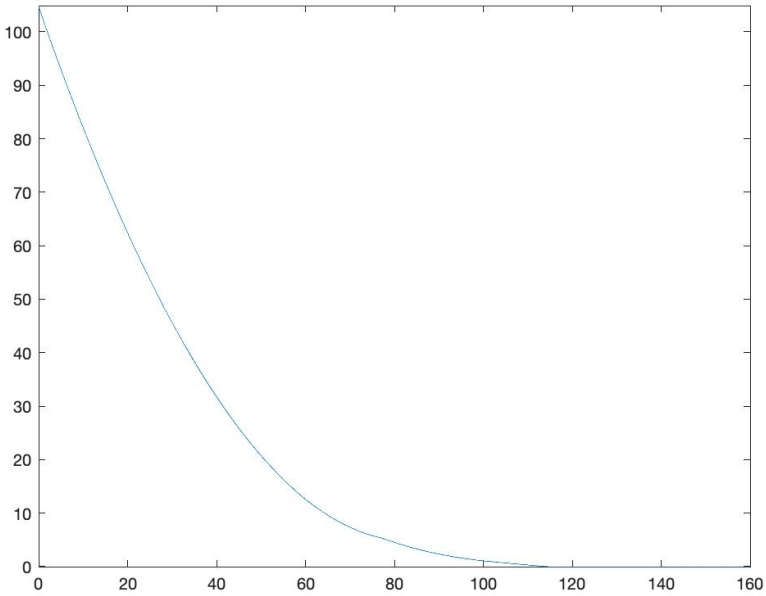
Trial 5 - Spinning, angle = 15, D=0.5,mu = 0.1, 1.5x Angular Momentum

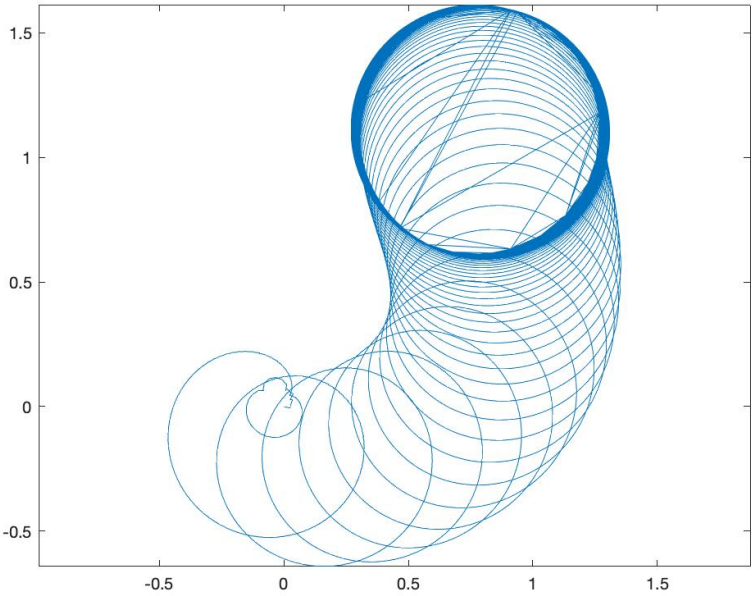




https://youtu.be/2Do6mHtjUtk

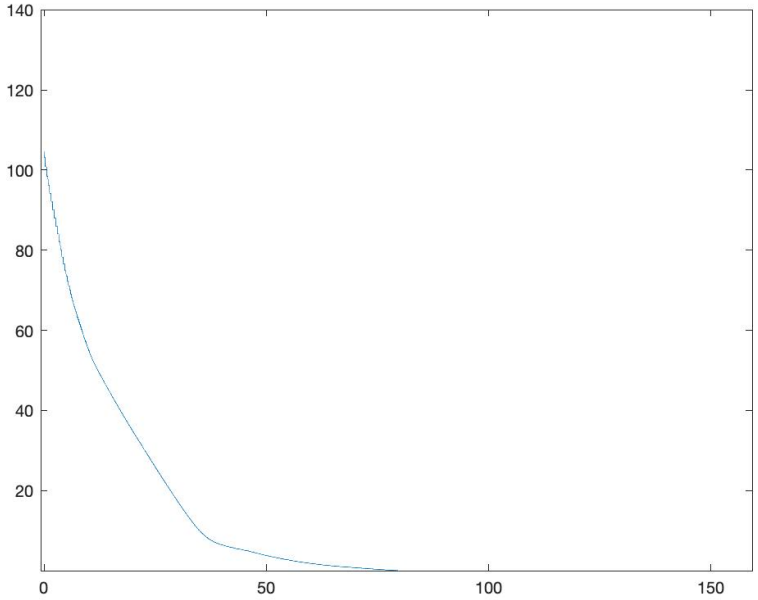
Trial 6 - Spinning, angle = 0, D = 0.5, mu = 0.3, , 5x angular momentum

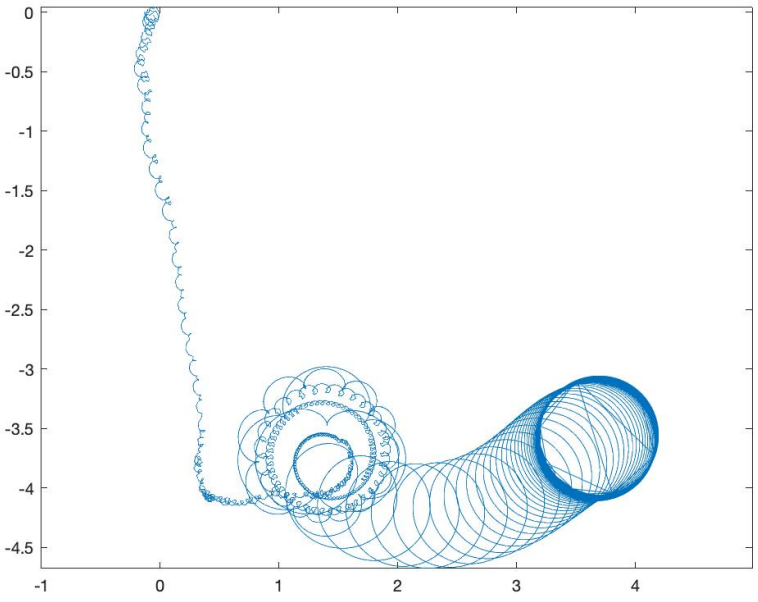




https://youtu.be/5UPVx__00u4

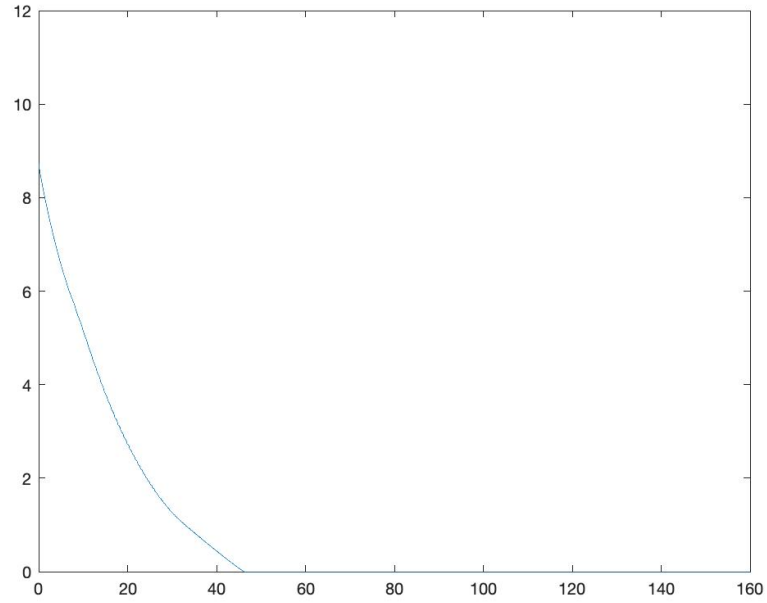
Trial 7 - Spinning, angle = 15, D = 0.5, mu = 0.3, 5x angular momentum

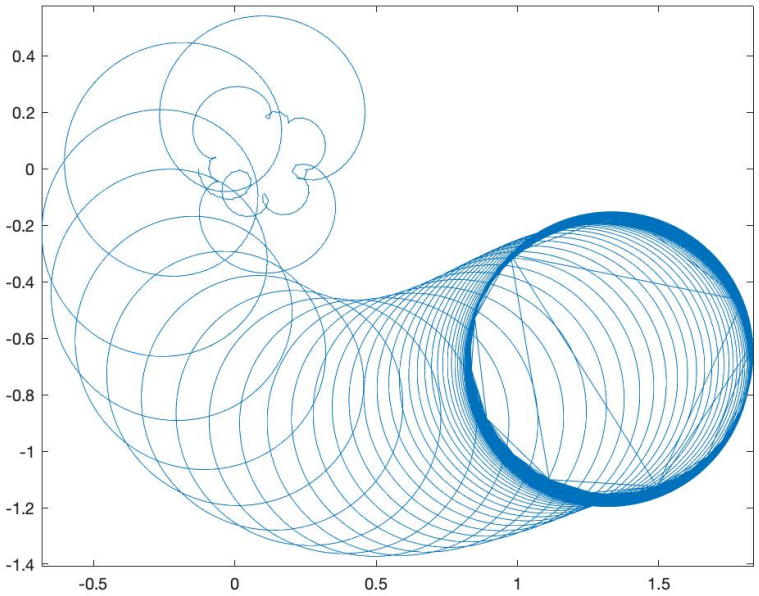




https://youtu.be/8uW3nynClpY

Trial 8 - Spinning, angle = 15, D = 0.5, mu = 0.3





https://youtu.be/7D-YEjNtFYk

Discussion

In all of the energy plots, the function we obtain is smooth. Furthermore, we see that the functions are monotonic decreasing, which is desired as energy should never be increasing in this system. In some of the trials, the function is step-like at the very beginning in energy reduction due to the fact that the coin is jumping, as seen in the videos provided in the URL, but then becomes smooth once the coin has no vertical displacement.

In general, there are three phases in the coin's motion: Pre-Tumble, Tumble, and Post-Tumble (rocking). Here I am calling "tumbling" as the motion where the coin rolls around. If you track a point on the coin, it seems to oscillate up and down, and move slightly with respect to the XY plane. The Pre-Tumble is the phase before the tumble. Depending on the orientation of the coin, this may vary drastically, as we will see when comparing Trials 1 and 2. The Post-Tumble is the last phase before the coin comes to rest. We see here that the coin seems to be rocking back and forth. This is supported by the fact that on the XY plot, we see that the minimum point jumps across, so there is some rocking occurring.

Rolling Coins- Trials 1,2,3

Looking at the coins that are rolling, we will first look at the Utan plots. We notice that the Utan plots start at some value greater than zero and then quickly diminish to zero. The way in which this is interpreted is that the coin is sliding when Utan is nonzero, thus we can see that towards the beginning for all of the rolling coins, there is some skidding on the ground. We see that after some time, namely after the first set of skidding, we can see that the tangent velocity to the ground oscillates but at a much lower range of values, and it tends to zero as time progresses.

Comparing results from Trial 1 and 2, we can see that when we increase the angular momentum, we get that initially, the coin travels in a path before it tips over and then rolls around on the ground by its side. We can see that because of the path taken by the coin initially, the coin settles into a more circular tumble phase, whereas the coin in Trial 1 seemed to be shifting towards the lower right corner during the tumble phase. In both cases, eventually, once energy has decreased enough, the coin starts to rock. Thus, we can see that increasing the angular momentum can have a major effect on the motion of the rolling coin!

Comparing results from Trial 1 and Trial 3, the difference here is that the coefficient of friction,μ, was changed from 0.3 to 0.8. From this, we can see that on the energy graph, the system with higher μ value actually continued to roll for longer. The video files of the coin support this claim. A possible reason is the fact that we have slipping in Trial 1 and no slipping in Trial 3. If you look at the Utan plots for both trials, we can see that Trial 1 has some spikes where

Utan is greater than zero, whereas in Trial 3, we see that the Utan starts off at a significant value greater than 0 and then almost immediately goes to zero.

Spinning Coins - Trials 4,5,6,7,8

Here, the angular momentum vector is parallel to the face of the coin. Specifically, it was set up to be parallel to the z component before the rotation.

In these trials we have some differences in at least of the parameters. We can therefore compare the plots and video for trials with just one change to see how the parameter has an effect, if at all.

Comparing Trial 4 and Trial 8, where the difference is in the angle, the former being vertical and latter being tilted at a 15 degree angle with respect to normal, we can see that in the XY plot, the minimum point of the coin in Trial 8 has a much more interesting path, where it is not circular, whereas the one in Trial 4 seems to be stable and rotating in one spot until it falls over into the tumbling stage, as seen in the video plot. Thus, generally, we can surmise than a tilt will have an effect on the spread of the coin in terms of its movement.

For Trial 4 and Trial 6, the difference is that the magnitude of the angular momentum is increased from 1 to 5. Here, we can see that the results are somewhat uneventful, in that the main difference is the increase in energy in the system. The plot is very similar in that the tumble and post tumble stages look very similar. There is more deviation from the origin in Trial 6, which implies that the coin's axis is changing slightly.

Now, looking at coins where the initial angle is set to be 15 degrees, we can look at how tilted coins differ with different parameters. Comparing Trial 8 with Trial 7, notice that Trial 5 is the same with Trial 8 with the exception of the angular momentum being 5 times greater.

Increasing the angular momentum even further from 1.5 times to 5 times as seen in Trial 7, we can see that the coin actually starts off with a vertical component. In other words, the coin jumps. This explains the plot, where we see that the pre tumble trajectory of the coin to be much longer, as it travels generally along a line before starting to go into a tumble. We can see that the pattern produced is much more intricate in Trial 7 and very much different to that in Trial 5. Thus, we can see that this system is very sensitive to initial conditions, and not surprisingly so.

Lastly, in Trial 5, the coefficient of friction was reduced to 0.1 and angular momentum is increased to 1.5 that of Trial 4. This was done to simulate conditions similar to that in the video. With this initial condition, we can see that the XY plot is very beautiful in that it seemed to trace out a circle.

In conclusion,we can therefore see that the system is very sensitive to the initial conditions.

Further Studies

Potential projects that arise from this project is the study of the interactions of other objects, such as spheres, with the ground. The code allows us to add spin to a sphere to see how it reacts with the ground. This has real world applications in sports like tennis, where one would want to know the trajectory of a ball, with and without spin.

Furthermore, with the code developed in the project, a potential project that stems from this one is to do a statistical analysis on whether starting the coin at an angle will have an effect on the distribution of heads and tails.

Acknowledgements

Special Thanks to Professor Charles Peskin for continuous support and consultation in working on this project with me, T.A Guanhua Sun for helping resolve coding issues, and to Professor Charles Puelz for providing me with a baseline code.

```
clear all
M_total= 1%0.00567 %in KG
g=9.8
S=1000 %stiffness
D=0.5 %damping
mu= 0.8 %sliding friction
tmax=16*10
clockmax= 1*16*10^5
dt=tmax/clockmax
tsave=zeros(1,clockmax);
%defined a rotation matrix
roty = @(t) [cos(t) 0 sin(t) ; 0 1 0 ; -sin(t) 0  cos(t)]
angle = pi/12 %radians, works = pi/8,pi/4, doesnot = pi/12
R = roty(angle);
%%%Coin
    num_points = 120;  %n points on circle
    radius = 0.5;
    initial_pos = [0,0,radius*cos(angle)];
    X = zeros(num_points,3); %position of each node
    U = zeros(num_points,3); %velocity of each node
    %U = [0.5,0.5,0].*ones(num_points,3);
    %U = ones(num_points,3);
    M=(M_total/num_points)*ones(num_points,1); %mass of each node

    % Generating points for the ball
        for num = 1:num_points
            theta = num * 2 * pi / num_points;
            X(num,:) = [0,radius*cos(theta),radius*sin(theta)];
            M(num) =  M_total / (num_points);
        end
        %Introduce a rotation
        X = (R*X')';
        X = X+initial_pos; %Translate ball to starting position

        %create L
        %L = *(cross(X(1,:)-X(16,:),X(2,:)-X(17,:))/norm(cross(X(1,:)-X(16,:),X(2,:)-X(17,:)
        L = 1*(R*[1 0 0]');
        Xcm = (1/M_total)*sum((M.*X)); %Center of Mass
        Ucm = (1/M_total)*sum((M.*U)); %Velocity rel to COM.
        Xrel = X-Xcm;
        Urel = U-Ucm;

%     %Creating Angular Velocity
%     for k = 1:num_points
%         L = L + M(k)*cross(Xrel(k,:),U(k,:))';
```

```
%     end
%


%Output
XcmSaveY=zeros(1,clockmax); %Save the position of center of mass
XcmSaveZ=zeros(1,clockmax);
Esave=zeros(1,clockmax); %Save the total energy in the system
Ymin=zeros(1,clockmax); %Save the position of the top point
Xmin = zeros(1,clockmax); %Save the position of the bottom point
UtanSave = zeros(2,clockmax);

v = VideoWriter('subplus, angle = 15, D=0.5,mu = 0.1, spinning 1.5x Angular Momentum  ');
open(v);

for clock=1:clockmax
  % Compute I
  I = zeros(3, 3);
  for k = 1:num_points
    I = I + M(k).*( (norm(Xrel(k,:))^2).*eye(3) - Xrel(k,:)'*Xrel(k,:) );
  end

  % Solve System for Omega, Update Xrel,Urel
  Omega = I\L;
  L;
  norm(Omega);
  %if(norm(Omega) > 100*eps)
      unit_Omega = Omega/norm(Omega);
      Omega_cross = [0 -Omega(3) Omega(2); Omega(3) 0 -Omega(1); -Omega(2) Omega(1) 0];
      P_Omega = unit_Omega*unit_Omega';
      Xrel = ( P_Omega*(Xrel') + cos(norm(Omega)*dt).*(eye(3) - P_Omega)*(Xrel') + sin(norm(
      Urel = cross((Omega'.*ones(num_points,3)),(Xrel));
  %end

  %Update Xcm
  Xcm=Xcm +dt*Ucm;

  %Update Ucm
  F_normal_point = ((X(:,3)<0).*(S*(-X(:,3))-D*U(:,3)));
  F_normal_point = (F_normal_point > 0).*F_normal_point;

  Utan = [U(:,1),U(:,2),zeros(num_points,1)];
  Utan_norm = (vecnorm(Utan')' + 10^-10)*[1,1,1];
  Utan_unit = Utan./Utan_norm;
  F_friction = - mu*F_normal_point.*Utan_unit;
  %F_friction = (F_normal_point(:)>0).*(- mu*F_normal_point.*Utan_unit);
```

```
  %F_friction = ((X(:,3)<0).*(- mu*abs(M*g).*Utan_unit));
  %F_friction = - mu*abs(F_normal_point/S).*Utan_unit;
  %F_friction = f_friction_kit(X,mu,M_total,g,Utan_unit,num_points);
%   if ~(sum(X(:,3)<0)==0)
%       F_friction
%   end
  force_z = (-M)*g + F_normal_point;
  force_point = zeros(num_points,3);
  force_point(:,1) = F_friction(:,1);
  force_point(:,2) = F_friction(:,2);
  force_point(:,3) = force_z;

  force = zeros(1,3);
   force(:,1) = sum(F_friction(:,1));
   force(:,2) = sum(F_friction(:,2));
   force(:,3) = sum(force_z);
  Ucm=Ucm+dt*force./M_total;


  %Update Angular Momentum

  torque = cross(Xrel,force_point);
  net_torque = sum(torque);
  L = L + dt*net_torque';


  tsave(clock)=clock*dt;
  %Zsave(clock)=Z(:,3);

  % update X
  X = Xcm + Xrel;
  U = Ucm + Urel;



  XcmSaveY(clock) = Xcm(2);
  XcmSaveZ(clock) = Xcm(3);
  [values,index] = min(X);
  Ymin(clock) = X(index(3),2);
  Xmin(clock) = X(index(3),1);
  UtanSave(:,clock) = Utan(index(3),1:2)';

  energy_ground = 0;
  for i = 1:num_points
      if F_normal_point(i) > 0
          energy_ground = energy_ground + 0.5*S*X(i,3)^2;
```

```
      end
  end

  Esave(clock)= M_total*g*Xcm(3) + 0.5*M_total*norm(Ucm)^2 + 0.5*Omega'*I*Omega + energy_gro



  %plot
%   if mod(clock,200) == 0
%   figure(1)
%   title('Plot')
%     x = [X([1:end,1],1)];
%     y = [X([1:end,1],2)];
%     z = [X([1:end,1],3)];
%     plot3(x',y',z','linewidth',4,'Color','b')
%     hold on
%     plot3(X(1,1),X(1,2),X(1,3),'ro','MarkerSize',4);
%     plot3(X((num_points/2+1),1),X((num_points/2+1),2),X((num_points/2+1),3),'go','MarkerSi
%     [value,index] = min(X);
%
%
%
%     line([-10 10],[-10 -10],[-0 -0])
%     line([-10 10],[10 10],[-0 -0])
%     line([10 10],[10 -10],[-0 -0])
%     line([-10 -10],[-10 10],[-0 -0])
%
%     for i = -9:9
%         line([-10 10],[-i -i],[-0 -0])
%         line([-i -i],[10 -10],[-0 -0])
%     end
%     view(45,15)
%     hold off
%
%     xlim([-10 10])
%     ylim([-10 10])
%     zlim([-10 10])
%
%    %test for rigid body
%    %norm(X(16,:)-X(1,:))
%    frame = getframe(gcf);
%    writeVideo(v,frame);
%   end

end
```

```
close(v)

figure(2)
%title('Top Point Z component axis vs Time')
plot(Xmin,Ymin)
figure(3)
%title('Center of Mass Z component axis vs Time')
plot(tsave,XcmSaveZ)
figure(4)
%title(Center of Mass Y component axis vs Time')
plot(tsave,XcmSaveY);

figure(5)
%title('Energy vs Time')
plot(tsave,Esave)
axis([0,tmax,0,12])
UtanVec = UtanSave(1,:).^2 + UtanSave(2,:).^2

figure(6)
plot(tsave,UtanVec)
```