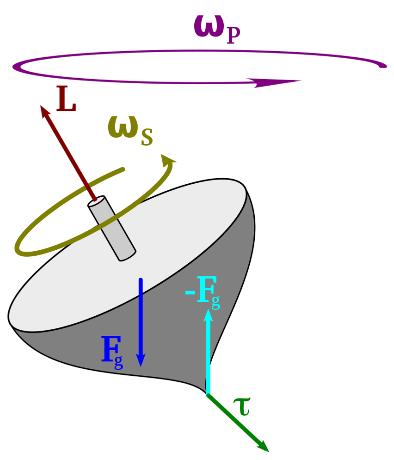
Simulation on Precession Motion of a Gyroscope and Analysis of Precession MotionWith Respect To the Tilted Axis

Meiqi Liu
Due Nov 1st

1.Introduction

Imagine a circle of spheres with mass circulating around a center of mass. In the plane of the circle/disk, there is a rotation around the center of mass of the system through the spin axis. The axis is not perpendicular to the ground but it is tilted to one side (and not parallel to the ground neither). There is a constant gravitation force F acts on the system to sustain the rotation(assume there is no friction in the air). We want to simulate motion of the disk (the system of spheres) more specifically study what factors would have impacts on precession. I'm going to use plots on trace of a mass point on the axis top and to trace the 3 dimensional position of a mass point in order to study the precession motion in x-y plane and change of the position in the z direction with respect to time step.

2.Equation and Motion



(Figure 1)

Based on the rigid body motion dynamic lecture, we can extract the key formulas to build our system of rigid body in the system. In a pure physics model(figure1), an initial velocity should be provided to push little sphere on the circle and causes an initial torque that leads spheres spinning around the spin axis. The precession is induced by gravitational force on the center of mass. Precession is defined as change of angular velocity and angular momentum produced by a torque. The gravitational force would induce such a torque and torque is rate of change of angular momentum,resulting in the following equation:

$$\tau = \frac{dL}{dt} = \sum_{k=1}^{n} X_k \times F_k \qquad (1)$$

Since $\tau = \overline{r} \times \overline{F} = \|r\| \cdot \|F\| \sin\theta$ (θ *is the angle between r and F*), the vector r is level arm and F is the force to cause spinning which is G in our case. Thus if we set our center of precession at origin(0,0,0) the level arm should become just X_k-coordinate of the sphere. F_k is known a gravitational force on kth mass point. And for a cross product we can use right hand rule to find the direction of torque.

And as we have analyzed earlier, there is an initial angular momentum to sustain the self-spinning motion(spinning around axis). Now a torque induced by G would have an impact on the initial angular momentum, resulting in a new angular momentum in the direction of the sum of the torque direction. This updating of angular momentum at each time step would cause precession to happen when the spin axis is not perpendicular to the ground.

So basically, we can simplify our program by fixing the bottom of axis (center of spinning) at (0,0,0) and setting the initial angular moment in prevent of setting up more variables such as center of mass and initial velocity. At each time step, we can compute and update torque. Angular momentum can be seen as overlaps of the spin axis so as we update angular momentum at each time step using torque we can observe the axis spinning around the center.

Ω is angular velocity and I is the moment of inertia tensor.
To simplify we need to assume Xcm is also at origin

To deduce the formula for L:

$$\tau = \frac{dL}{dt} = \sum_{k=1}^{n} (X_k - X_{cm}) \times F_k$$

$$L = mrv = \sum_{k=1}^{n} m_k (X_k - Xcm) \times (U_k - U_{cm}),\ U_k = U_{cm}(t) + \Omega(t) \times (X_k(t) - X_{cm}(t))$$

$$\Rightarrow L(t) = \sum_{k=1}^{n} m_k (X_k - X_{cm}) \times (\Omega(t) \times (X_k(t) - X_{cm}(t)))$$

$$\Rightarrow L(t) = \sum_{k=1}^{n} m_k \left\| X_k(t) - Xcm(t) \right\|^2 \cdot \Omega(t) - \sum_{k=1}^{n} m_k (X_k(t) - Xcm(t))(X_k(t) - Xcm(t)) \cdot \Omega(t)$$

To deduce the formula for I :

$L = I \cdot \Omega$ (3)

$$\Rightarrow I(t) = \sum_{k=1}^{n} m_k (\|X_k(t)\|^2 E - X_k(t)(X_k)^T) \text{ if } Xcm = 0 \quad (2)$$

So we deduced the formula for I which can be used later to compute angular velocity Ω. As we know the angular velocity we can update the position of mass sphere by using an operator P.

3.Numerical Method

In light of lecture note and with a little modification we can get our matlab code.
We assume the initial L and to update L we need to set up parameter logically:

force = mass $\times$ g (about 9.8 kg/m^2);
Radius=0.8;
Sum of mass=5kg; since each mass point is 0.5 and there are 10 points in total;
I=5 kg $\cdot (0.8m)^2 \approx 3kg \cdot m^2$
$|L| = 3kg \cdot m^2 \cdot 2\pi/s = 20kg \cdot m^2/s$
L=(4, 0,20) $\sim$ (400, 0, 2000) a small value on x dimension to make axis tilted, then we increase the value of the vector 100 times for simulated object to move faster in matlab.

Initial torque $L(t)$ can be computed by equation (1)

At each time step, since the conservation of angular moment in the rigid body we can compute the I at each time step using (2) and from where we can get angular velocity from (3) and by Omega we can construct an operator P and finally build a transformation matrix R to update the mass point position when angular velocity is imposed. At last, we can use the initial torque and plug the new position into equation (1) to update the net torque. This will complete the last step of the time step and the program will start to repeat the loop until the end time is reached.

$$I(t) = \sum_{k=1} m_k(\|X_k(t)\|^2 - X_k(t)X_k(t)^T)$$

$$\Omega(t) = I(t)\backslash L(t)$$

$$R(\Omega, \Delta t) = P(\Omega) + \cos(\|\Omega(t)\|\Delta t)(E - P(\Omega)) + \sin(\|\Omega(t)\|\Delta t)(\frac{\Omega}{\|\Omega\|}\times)$$

$$X_k(t + \Delta t) = R(\Omega, \Delta t)X_k(t)$$

$$L(t + \Delta t) = L(t) + \Delta t\sum_k X_k(t + \Delta t) \times F_k(t)$$

4.Program with Comments

```
%--------data--------------------------------------------------------------%
close all; clear all; clc;


% number of dimension
xyz = 3;

% number of nodes around the circumference of the disk
n = 10;

% initialize links and nodes
num_nodes = 14;
num_links = 23;%
jj = zeros(num_links,1);
kk = zeros(num_links,1);
X = zeros(num_nodes, xyz);% position

% set physical parameters
radius = 0.8; % radius of ball

M = ones(num_nodes,1);
M(1:n) = (5)*ones(n,1); % mass of each node
M(n+1:13)=1;% start from n+1
M(end) = 1; % mass of the last node which is the bottom of the axis in our case
```

```
G =10; % magnitude of the gravitational force


% set numerical parameters
dt = 1e-3;%this control the speed of animation
end_time = 400 *dt; %how long the time span is;
timevec = 0:dt:end_time;

% make a gravitational force
F_gravity = zeros(num_nodes, xyz);
F_gravity(:,1) = 0;
F_gravity(:,2) = 0;
F_gravity(:,3) = -G.*M;


% drawing the circle,we wanr the disk a little bit above the ground- especially the point(0,0,0) so
that
% our axis can stand on the center(0,0,0)
center_circle = [0, 0, 1];

% drawing the circle
for k = 1: k
   theta = 2*pi*k/n;
   X(k,:) = center_circle + radius*[cos(theta), sin(theta),0];

end


% set position of center
X(n+1,:) = center_circle;


% set position of the three other endpoints
X(12,:) = center_circle+[0,0,0.3];
X(13,:) = center_circle-[0,0,0.7];
X(14,:)=[0,0,0];


% naming index sets for the links
```

```
spokes = 1:n;
rimlinks = (n+1):2*n;


% build structure by creating links
jj(spokes) = 1:n;
kk(spokes) = n+1;
jj(rimlinks) = 1:n;
kk(rimlinks) = [2:n, 1];

%connect 12 and 13, all connect to 11
jj(21) = 11;
jj(22) = 11;
kk(21) = 12;
kk(22) = 13;

% connect 13 and 14
jj(23)=13;
kk(23)=14;

% so there are 14 pts in total

% start to plot the space
figure(1)
colordef black
grid on
axis equal
view(3)
hold on

xlabel('X')
ylabel('Y')
zlabel('Z')


% plot initial structure
x = [X(jj,1) X(kk,1)];
y = [X(jj,2) X(kk,2)];
z = [X(jj,3) X(kk,3)];
```

```
plot3(x',y',z','linewidth',4)

%------------------------------------plot done ----------------------------------------------------

% initialize the angular momentum

% L = [500; 0; 2000];
L0=[0;0;1000]
 L=rotate*L0;

%titled the axis
theta=pi/4;
rotate=[1 0 0;0 cos(theta) -sin(theta);0 sin(theta) cos(theta)];
for k=1:14
    X(k,:)=(rotate*X(k,:)')';
end



% we want to store the position of the head of the axis at each time step
 point_position = zeros(length(timevec), xyz);

 %make a video of result
 v = VideoWriter('newfile.avi');
 open(v)


for t = 1:length(timevec)
      % this just displays the current simulation time in the matlab window
      disp(timevec(t));



    I = zeros(xyz, xyz);
    % compute moment of inertia tensor
    for k = 1:13% 13 mass pts
    I = I + M(k).*( (norm(X(k,:))^2).*eye(xyz) - X(k,:)'*X(k,:) );
    end
```

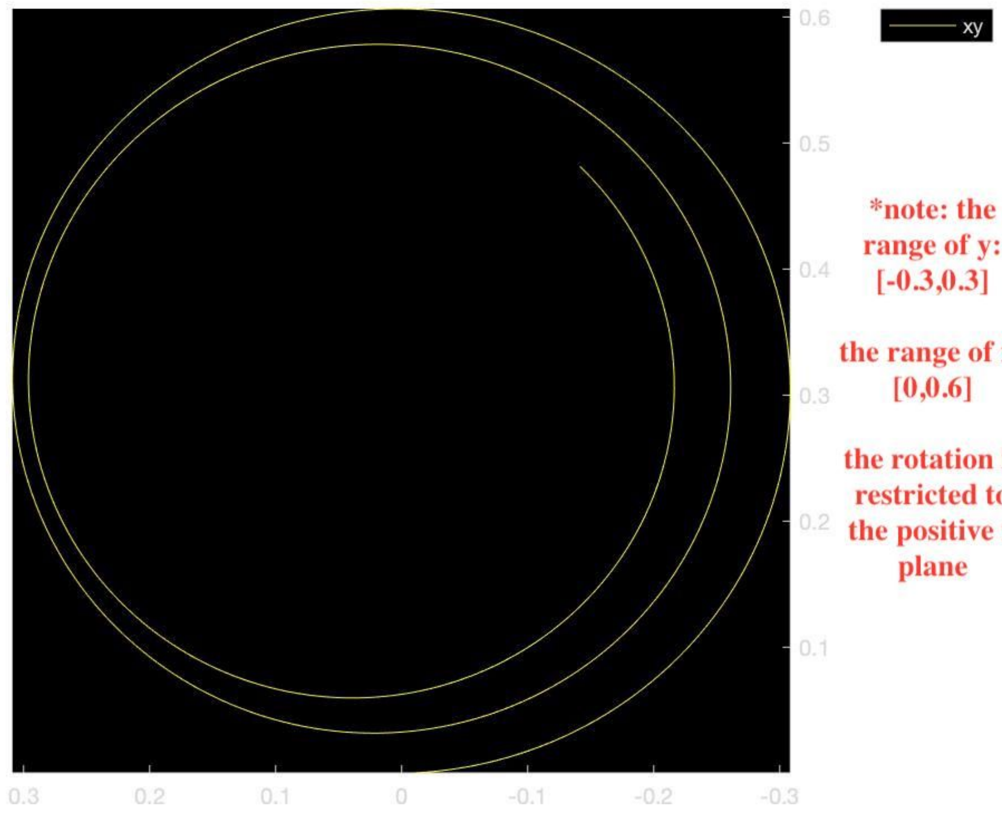
```
   % compute the angular velocity
     Omega = I\L;

   % normalize angular velocity if it is nonzero
  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';
       X= ( P_Omega*(X') + cos(norm(Omega)*dt).*(eye(xyz) - P_Omega)*(X') +
sin(norm(Omega)*dt).*(Omega_cross*(X'))./norm(Omega) )';
  end



  net_torque = zeros(xyz,1);
     % update torque
    for l = 1:13% 13 mass pts
       net_torque = net_torque + cross(X(l,:)', F_gravity(l,:)');% + ground_force(l,:)');
    end

   L = L + dt.*net_torque;

   % picked a point to draw
     point_position(t,:) = X(n+2,:);

   % plot the current position of the ball
   figure(2);
   x = [X(jj,1) X(kk,1)];
   y = [X(jj,2) X(kk,2)];
   z = [X(jj,3) X(kk,3)];
   plot3(x',y',z','linewidth',4)
   xlim([-3 3])
   ylim([-3 3])
   zlim([-3 3])
   pause(0.0001)

   frame = getframe(gcf);
   writeVideo(v,frame);
```

```
end

figure(3); hold on
plot(timevec,point_position,'linewidth',2)
legend('x','y','z')

figure(4); hold on
plot3(point_position(:,1),point_position(:,2),point_position(:,3))

legend('x','y','z')

close(v)
```

5.Results + Discussion

Method 1

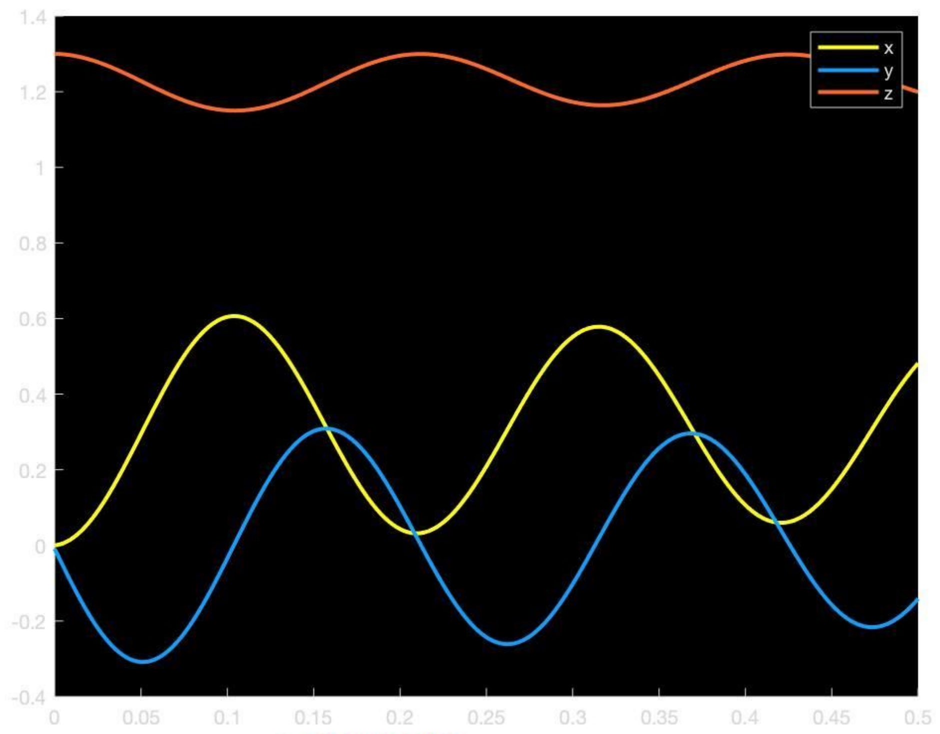
If we apply an angular momentum that we have calculated from above with a random direction such that direction vector is not perpendicular to the ground, the spinning trace is restricted to the positive x domain. This means the axis would become perpendicular to the ground when it goes close and above to the origin position. This situation the model is correct in terms of physics, however, it is hard to observe in real life since it’s hard to set up the initial angular momentum titled at exact such angle that we formulated in the beginning. We are setting up angle by manipulation of vector for angular momentum. In the matlab code, the rigid body structure firstly stands perpendicular on ground and then a titled angular moment is applied to allow precession to happen. The initial structure is not tilted properly as it is supposed to be parallel to the angular momentum and that is what we will implement in the second method. For method 1, there is an angle between the angular momentum and axis, this will cause difference between the matlab simulation and real physics phenomenon. We plot the position of axis top in terms of time step with curve represents x,y and z position with respect to time and the other graph shows the trace that the axis top draws in 3d space.



***note: the range of y: [-0.3,0.3]**

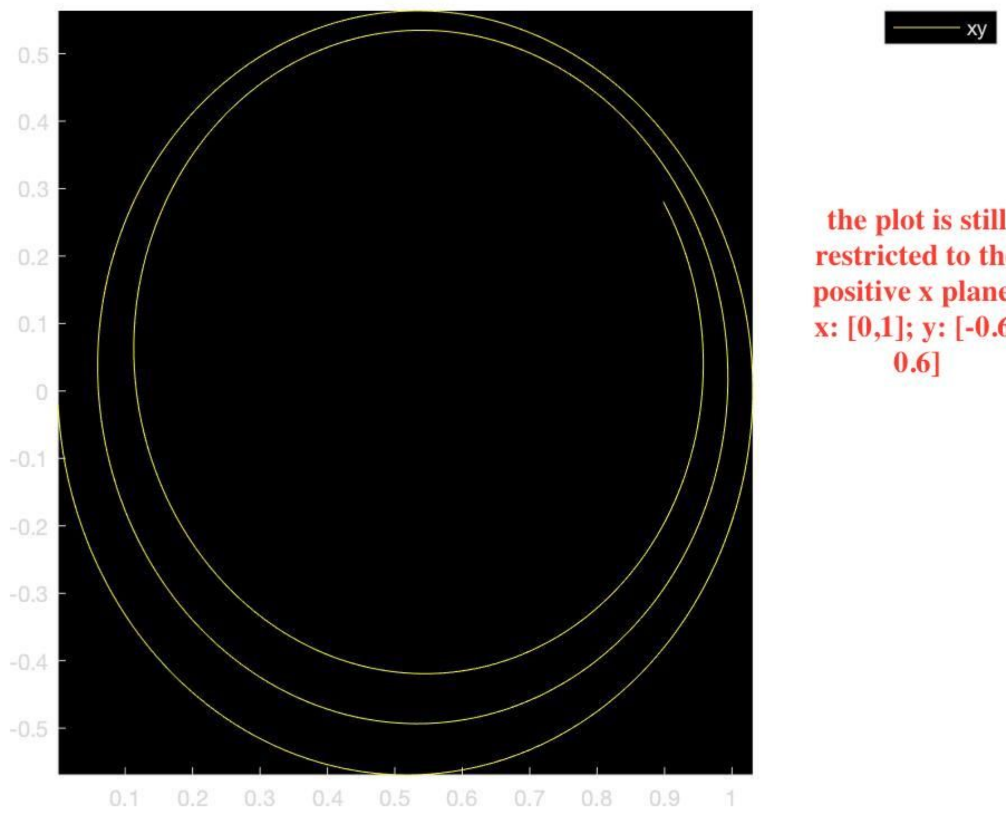
the range of x: [0,0.6]

the rotation is restricted to the positive x plane

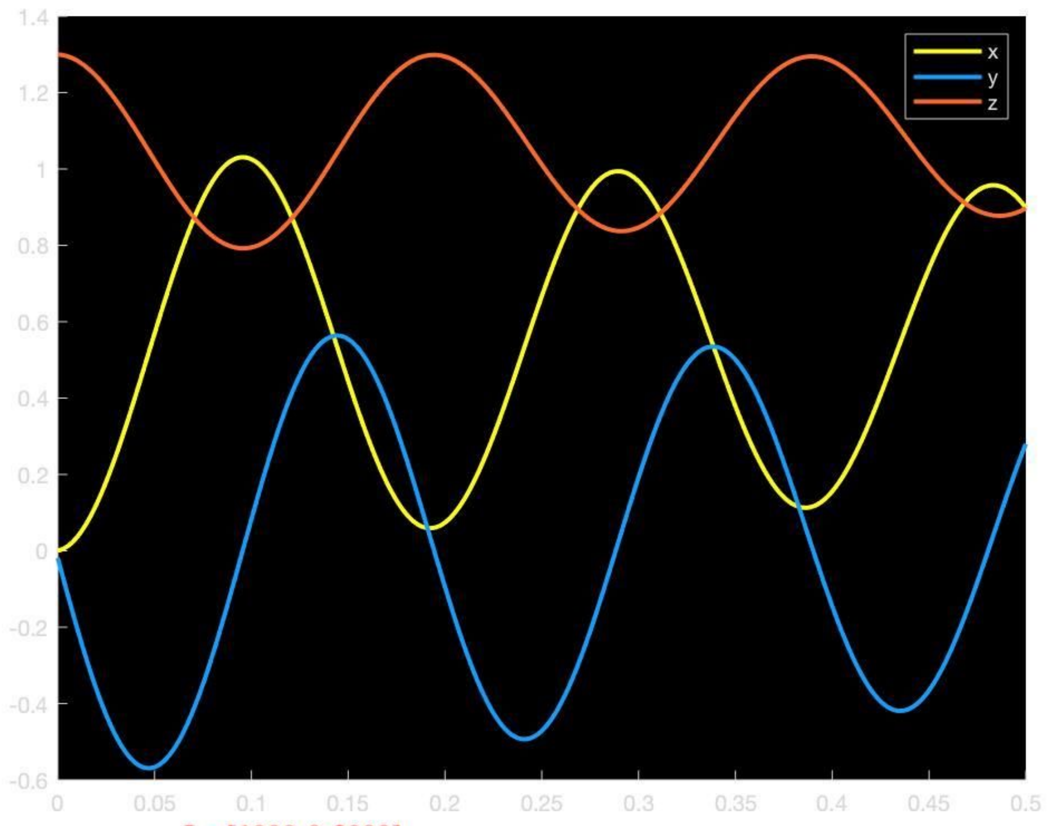


L =[500;0;1000]

This red curve shows there is a change in the z direction; the curve is fluctuating means when the axis is rotating around the spinning center the axis top is not stable as it has a little oscillation in the z direction as well. And if we increase the value of angular momentum in the x direction, by which we really are decreasing the angle between angular momentum and the ground. So with a more tilted angular moment we can have another set of result plot.



the plot is still restricted to the positive x plane; x: [0,1]; y: [-0.6, 0.6]



L =[1000;0;2000]

The results show more change in both x,y and z position for same time, means the gyro spinning faster and therefore result more error in z position. The circular trace becomes larger since we decrease the angle.

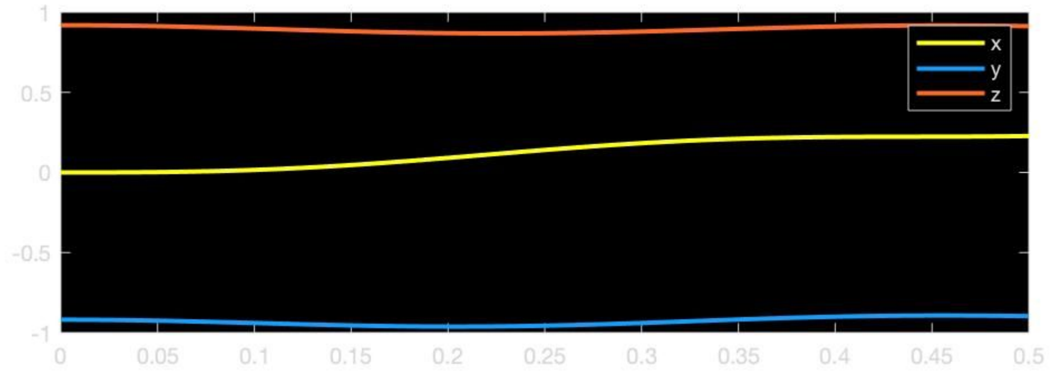
<u>Method2</u>

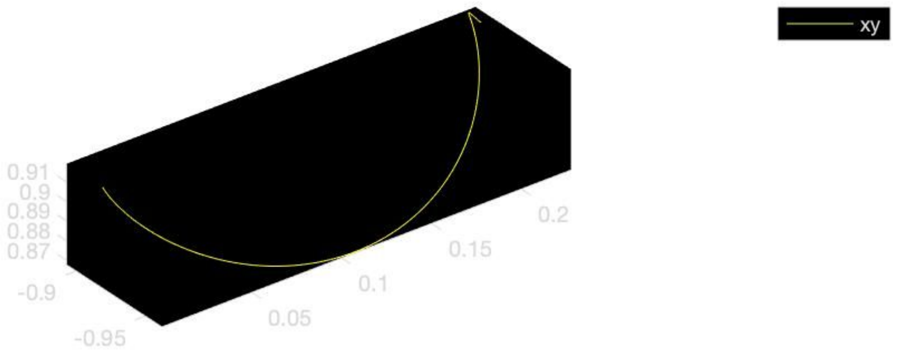
In this method we try the more natural model with angular momentum and the axis tilted overlapping each other at the same time. In order to do that, in matlab code in the place before numerical method we need to add:

```
 %titled the axis
theta=pi/4;
rotate=[1 0 0;0 cos(theta) -sin(theta);0 sin(theta) cos(theta)];
for k=1:14
   X(k,:)=(rotate*X(k,:)')';
end
```

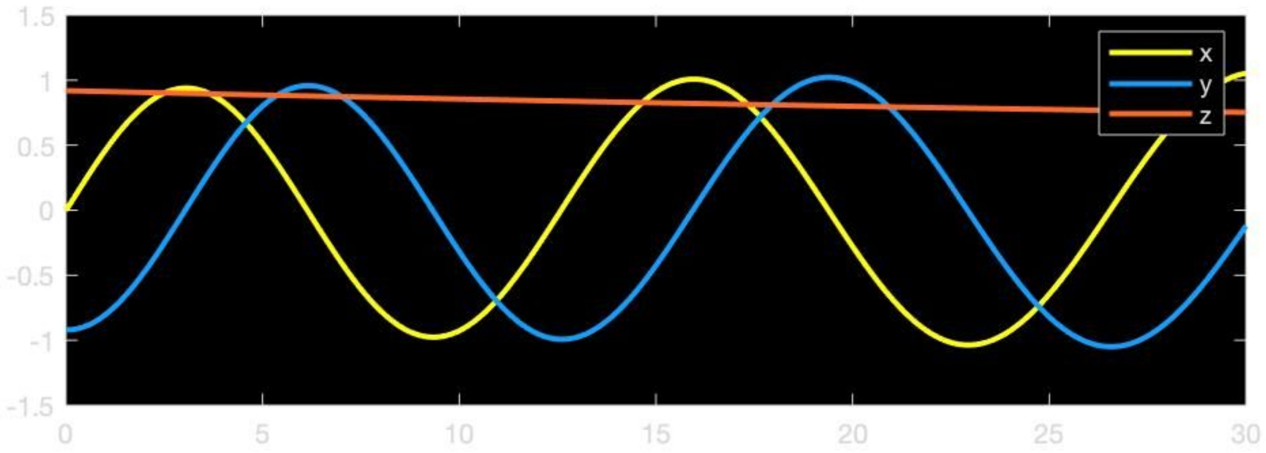
L0=[0;0;1000];
L=rotate*L0;

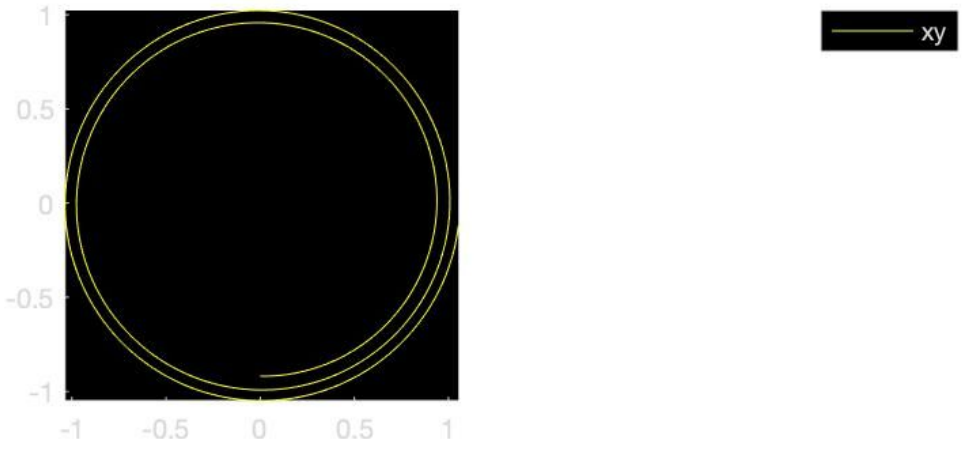
We can observe no change in the z direction. To get the below graph we plug in L0=[0;0;1000] then apply a rotation matrix to tilt L0 and rigid body structure. The below shows the result without changing time dt=1e-3. The animation appears only rotation around the axis but no rotation around the center. The plot below implies xy position are still changing but in an extreme slow rate. So we need to decrease the time step to speed up animation.





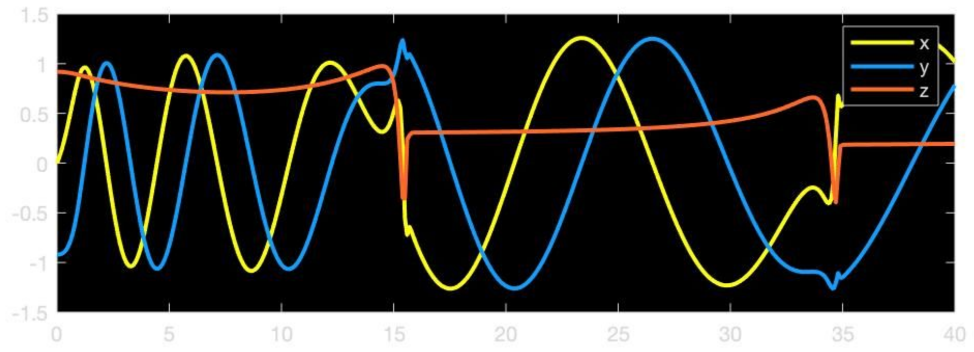
After shrinking the time step (dt=1e-1) we have the following result. The animation looks more smooth and natural as the trace is almost a nice circle and the change of height is almost a constant shown by a straight line. And the spinning rate around the axis is faster than the spinning rate around the center, which can be explained by physics model.

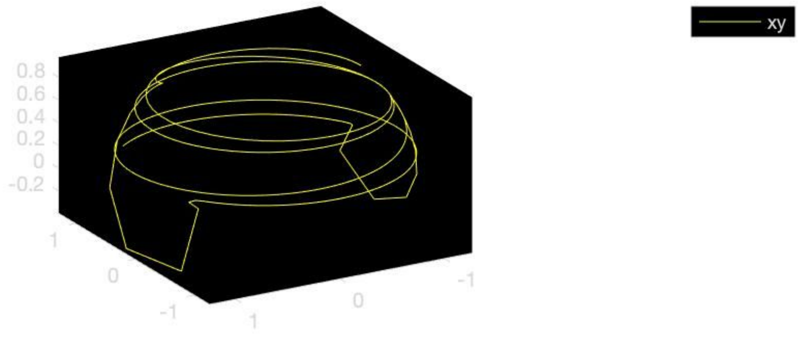




If we want to observe the rigid body wobbles we can construct time interval and apply an extra force on the axis top to result a different plot.

```
% to apply an extra force on the axis top
  if(t>0 && t<=dt*200)
          F_gravity(12,3)=F_gravity(12,3)-100;
end
```





In this plot, we can see the wobble in z direction is periodic; the radius of the circle in the trace is increasing as the height of the axis is dropping showing there is an error.

Overall, our experiment shows matlab can accurately simulate the precession in physics world. And in all animations in method2, we can observe the rotation around spin axis is the way more faster than the rotation around the center. This is interesting to see how matlab matches the true physical case. The larger the angle from the axis to the ground the faster the precession motion is. And the spinning around axis is relatively faster than spinning speed around the center.