Simulation of Rigid Body Dynamics in Matlab

Varun Ganapathi
Department of Physics
Stanford University

May 14, 2005

Abstract

This report presents a simulator of rigid dynamics of a single body in Matlab. The focus was on the conservation of Angular-Momentum and we assume that we're in the center of mass frame with no external forces. We show that the simulation agrees with the analytical solution for the case of a body with two equal moments. Qualitatively, we also show that the model exhibits the expected behavior when the moments of inertia are all different. We extend the model to include applied torques, but the torque must be calculated analytically through some other means. We show the numerical solution of the example of a rigid body with two rockets on each side of an ellipsoid, aimed to provide equal and opposite forces, and thus some total torque about the z-axis in the inertial frame.

1 Introduction

Many think of classical mechanics as highly intuitive and easy to visualize. However, even in what is the most well understood area of physics, there are still behaviors that arise that are surprising and unexpected. An example is the behavior of a free body rotating in space. Most people would expect that a body simply rotating should generally continue to rotate at the same rate about the same axis in the absence of torque, while this in fact generally untrue. ℓ and ω are not always in the same direction.

Yet, it is not the case that rigid body dynamics rarely occur. In fact, many objects in the real world exhibit these effects. If the angular momentum and angular velocity vector of a car's tires don't align, the passengers will feel an uncomfortable vibration due to the precession of the tire about the angular momentum vector. In a helicopter, the rotor disc has a large moment of inertia and spins quickly, therefore having a large angular momentum, which has significant effect on the helicopter's handling in the air.

This paper describes a way to numerically solve the equations of motion for a rotating rigid body. The Euler equations, found in any graduate level mechanics text, form the foundation of our method. [1] From there, we write the first order differential equation relating orientation represented as quaternion to the angular velocity. The Runge-Kutta method is used to integrate the resulting coupled pair of first order differential equations.

2 Equations of Motion

Here we summarize the physics relating to rigid body rotation. Angular Velocity $\omega\mathbf{u}$ is a vector quantity. The magnitude of ω represents the rate of rotation about the axis defined by the unit vector $\mathbf{u}$. The velocity of a point on a body rotating with angular velocity ω is given by

$$v = \omega \times \mathbf{r}$$

which leads to the equation relating the derivative of a vector in an inertial reference frame to the derivative in reference frame rotating with angular velocity Ω:

$$\frac{d}{dt}\left(Q_{inertial}\right) = \frac{d}{dt}\left(Q_{body}\right) + \Omega \times Q \tag{1}$$

Every rigid body has an associated inertia tensor (2) that is symmetric and real-valued, shown here with summations. For continuous bodies, the sums are trivially replaced with integrals.

$$\mathbf{I} = \begin{bmatrix} \sum m(y^2+z^2) & -\sum mxy & -\sum mxz \\ -\sum myx & \sum m(x^2+z^2) & -\sum myz \\ -\sum mzx & -\sum mzy & \sum m(x^2+y^2) \end{bmatrix} \tag{2}$$

The Principal Axis of a body (about an origin O) is any axis through O with the property that if ω points along the axis, then $\ell = \lambda\,\omega$. In other words, the principal axes are the eigenvectors of the inertia tensor. (2) It is always possible to choose a coordinate system such that the inertia tensor is a diagonal matrix. This is a direct result of a mathematical theorem stating that any real symmetric matrix is diagonalizable.

Angular momentum is now defined as the inertia tensor I multiplied in the normal matrix fashion with the angular velocity vector ω.

$$\ell = \mathbf{I}\omega \tag{3}$$

The law of conservation of angular momentum states that

$$\frac{d}{dt}\ell = \tau \tag{4}$$

In the case of zero torque, applying (1) to ℓ, substituting (2) and expanding the cross product results in the Euler equations

$$\begin{aligned} \dot{\omega_1} &= \left(\frac{\lambda_2 - \lambda_3}{\lambda_1}\right)\omega_2\omega_3 \\ \dot{\omega_2} &= \left(\frac{\lambda_3 - \lambda_1}{\lambda_2}\right)\omega_3\omega_1 \\ \dot{\omega_3} &= \left(\frac{\lambda_1 - \lambda_2}{\lambda_3}\right)\omega_1\omega_2 \end{aligned} \tag{5}$$

where λ_i is the i^{th} diagonal element of the inertia tensor. It is important to remember that in the Euler equations, ω_i is the i^{th} element of ω expressed in the principal axis frame. This is why the the off-axis terms of **I**(2) do not appear in the Euler equations(5).

3 Numerical Solution of Equations

The Euler equations can be written in vector form as

$$\mathbf{I}\dot{\omega} = \ell \times \omega \tag{6}$$

where all the quantities are expressed in the principal axis frame. Since the angular velocity equation is independent of all other state variables in the simulation, we can consider solving the problem in two steps. The first step is to simulate the evolution of the angular velocity vector in the body frame, and the second step is to solve for the orientation.

3.1 Solving for Angular Velocity in the Body Frame

The solution $\omega(t+dt)$ given $\omega(t)$ is straight-forward. In Matlab, based on the vector form of the Euler equations, we can write a function giving $\dot{\omega}$ as:

```
function [Wdot] = getWdot( W, T, I )
Imat = diag( I);
Imat_inv = diag( 1 ./ I );
Wdot = Imat_inv* cross( Imat * W, W );
```

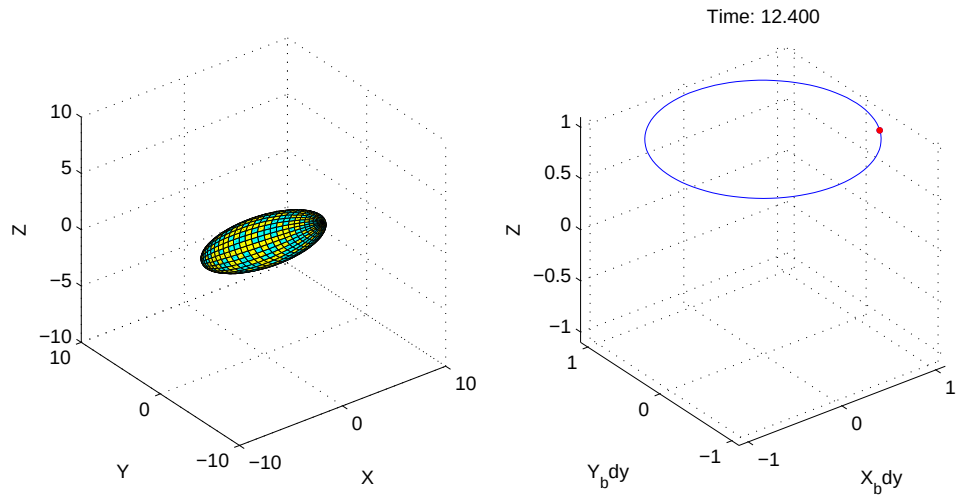

Figure 1: Superimposed analytical and numerical Solution for a symmetrical top with initial I = [2 2 8], W = [1 0 1] (The solutions are indistinguishable in this view)

Then we can numerically integrate ω using fourth order Runge-Kutta:

```
k_w(:,1) = dt * getWdot( W, t, I );
k_w(:,2) = dt * getWdot( W+.5*k_w(:,1), t+.5*dt, I);
k_w(:,3) = dt * getWdot( W+.5*k_w(:,2), t+.5*dt, I );
k_w(:,4) = dt * getWdot( W+k_w(:,3), t + dt, I );
W = W + (1/6) * ( k_w * [1;2;2;1]);
```

We can test the part of the code simulating the evolution of ω in the body frame for the simple case in which two of the three terms of the diagonalized inertia tensor are equal:

$$\lambda_1 = \lambda_2 < \lambda_3$$

In this case, with the intial $\omega = [\omega_0\ 0\ \omega_3]$, we can show that the analytical solution[2] of ω is:

$$\omega = [(\omega_0 \cos(\Omega t)),\ (\omega_0 \cos(\Omega t)),\ \omega_3)]$$

where

$$\Omega = \frac{\lambda_3 - \lambda_1}{\lambda_1}\omega_3$$

The figure(1) shows the numerical and analytical solutions for $w_b(t)$ superimposed on the right side. In the diagram the oval is actually a circle plotted analytically using plot3 viewed from an oblong angle. The two are completely coincident, at least at the resolution of the figure. Figure(2) shows the plot of error as a function of time step.

3.2 Solving for the Orientation

In the case of linear motion, we found that we could write two coupled first order differential equations describing the motion.

$$\dot{x}(t) = v(t)$$
$$\dot{v}(t) = f(x, v, t)$$

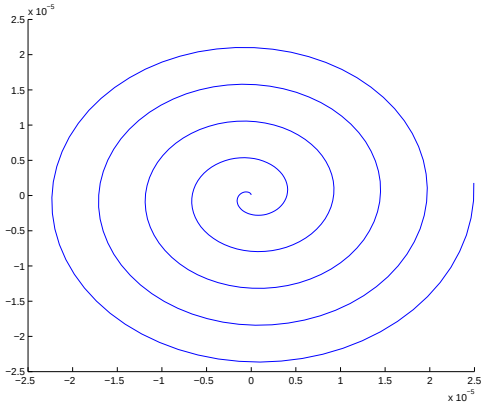

Figure 2: Error in prediction of ω shown for 300 time steps. The y axis the error in ω_y, x is error in ω_x, both in units of 10^-5 rad/s. The error spirals outward gradually with time.

where f is the force as a function of position, velocity and time. To solve that problem we interleaved the Runge-Kutta updates to integrate the coupled equations. Similarly, in this case, we can imagine a variable Q that represents the orientation. The derivative of the orientation is some function of ω.

It is possible to use Euler angles to represent orientations, but there are two problems that would occur. The first is that there is a problem known as gimbal lock where two of the rotational axes of an object merge together. After this happens, it is essentially impossible to rotate about one of the three dimensions. For more details, see [3]. Second, the integration of Euler angles requires many evaluations of trigonometric functions which are generally slower than multiplications.

3.3 Quaternions

Quaternions are four-dimensional unit vectors of the form $Q = [s\ \mathbf{v}]$, developed by the physicist Hamilton. To represent a rotation of angle θ about a given axis $\mathbf{u}$, one can write the quaternion

$$Q = [\cos(\frac{\theta}{2})\ \sin(\frac{\theta}{2}) * \mathbf{u}]$$

Many operations are defined for quaternions. The most important for the purpose of this paper is the formula converting a quaternion into a rotation matrix.

$$R_q = \begin{bmatrix} 1 - 2v_y^2 - 2v_z^2 & 2v_xv_y - 2sv_z & 2v_xv_z + 2sv_y \\ 2v_xv_y + 2sv_z & 1 - 2v_xv_x - 2v_zv_z & 2v_yv_z - 2sv_x \\ 2v_xv_z - 2sv_y & 2v_yv_z + 2sv_x & 1 - 2v_xv_x - 2v_yv_y \end{bmatrix} \tag{7}$$

If a quaternion represents the orientation of a rigid body, then $r_w = R_q r_b$, where r_b is a column vector $[xyz]$ represents a point in the frame of the rigid body, and r_w represents the same point in the world frame.

A quaternion can also be thought of as representing a rotation. Multiplying two quaternions p, q, gives a quaternion that represents the application of the rotations represented by p and q in sequence.

$$q_3 = q_1 \times q_2 \tag{8}$$

$$= [s_1s_2 - v_1 \cdot v_2\ ,\ s_1 * v_2 + s_2v_1 + v_1 \times v_2] \tag{9}$$

In order to integrate the orientation, the following formula giving the derivative of the quaternion as a function of orientation and angular velocity in the world frame is also required:

$$\dot{\mathbf{Q}} = [0\ w] \times Q$$

where [0 w] is interpreted as a quaternion and the rules for quaternion multiplication applied (8). Due to numerical error, a quaternion may gradually drift from having unit length. It turns out that the correct way to deal with this problem is to simply re-normalize the quaternion so that it points in the same direction but has unit magnitude.

3.4 Integrating Orientation

To find the derivative of the quaternion representation of the orientation at each time step given the angular velocity in the body frame and the current orientation of the rigid body, we apply the following steps.

1. Find the rotation matrix representing the current orientation of the rigid body
2. Rotate ω_b into the world frame
3. Find $\dot{\mathbf{Q}}$ given $\mathbf{Q}, \omega_w$

In Matlab, the code is:

```
function [qdot] = getQdot(w q )
R = quatToMat(q); w_inl = R*w;
```

We can then apply fourth-order Runge-Kutta in Matlab as follows. In the code shown here, `Q,W` are respectively quaternions representing the orientation and the angular velocity.

```
%R-K4 Integrate W to find Wnew in body frame
k_q(:,1) = dt * getQdot( W, Q );
k_w(:,1) = dt * getWdot( W, t, I );
k_q(:,2) = dt * getQdot( W+.5*k_w(:,1), Q+.5*k_q(:,1) );
k_w(:,2) = dt * getWdot( W+.5*k_w(:,1), t+.5*dt, I);
k_q(:,3) = dt * getQdot( W+.5*k_w(:,2), Q+.5*k_q(:,2) );
k_w(:,3) = dt * getWdot( W+.5*k_w(:,2), t+.5*dt, I );
k_q(:,4) = dt * getQdot( W+k_w(:,3), Q+k_q(:,3) );
k_w(:,4) = dt * getWdot( W+k_w(:,3), t + dt, I );
Q = Q + (1/6) * ( k_q * [1;2;2;1]);
W = W + (1/6) * ( k_w * [1;2;2;1]);
% renormalize q
Q = Q ./ sqrt(dot(Q,Q));
```

4 Analytical Solution of a simple case and comparison

To evaluate the correctness of our model, we continue considering the case of a symmetric top with two moments of inertia equal and one moment of inertia greater. We take the derivation of the analytical solution given here from Landau. Let us denote the principal axes with the same eigenvalue as x_1, x_2 and x_3 as the principal axis with the largest moment of inertia. Since $\lambda_1 == \lambda_2$, we can rotate x_1 and x_2 axes about x_3 arbitrarily. In other words, because the top is symmetric about x_3, any choice of x_1, x_2 such that they are orthogonal to each other and x_3 is valid. This means that we can choose x_2 to be perpendicular to both the angular momentum vector and x_3. In that case, ω lies in the same plane as ℓ and x_3. This means that the instantaneous velocity of any point along the ω vector through the origin of the rigid body is perpendicular to the radius vector. This implies that the symmetric top will precess at a constant velocity in a cone about the angular momentum vector and spin about x_3 at a constant rate. The true solution is given by:

$$\Omega_{pr} = |\ell|/\lambda_1 \tag{10}$$

$$\Omega_3 = (|\ell|/\lambda_3)\cos(\theta) \tag{11}$$

where θ is the angle between x_3 and ℓ. The program comes with a test script which can run any of five different tests and plots the analytical solution for both the path of the angular velocity vector and the x_3 vector. The animation helps illustrate the behavior the best. However, figure(3) tries to illustrate the behavior as well(See caption). The numeric error in estimating the orientation of the top is similar to the plot error of the angular velocity vector in figure(2).

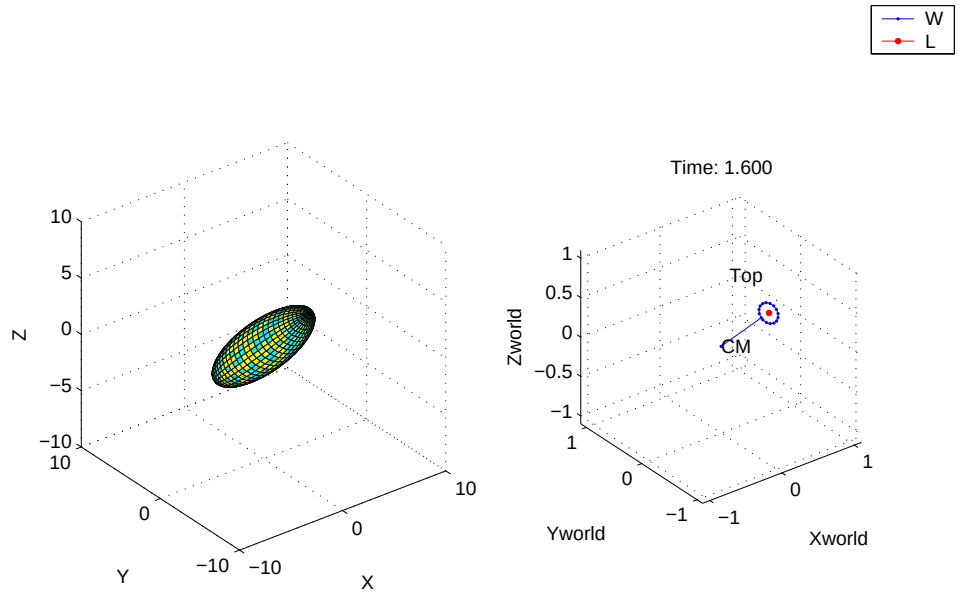

Figure 3: Simple case of symmetrical Top: The graph on the right side shows the angular momentum vector and angular velocity vector in the world frame. As shown, the ω lies on a cone centered about ℓ. The analytical solution is shown as the thin blue line, and the solid blue dots show the evolution of the numeric solution, which fall with high accuracy on the analytic solution.

5 Graphics Display

An important part of any computational physics model is an graphical representation that helps us to understand what is happening in the model. After all, part of the purpose of the model is to help us to undertand cases that are not easily soluble analytically and to gain intuition. In the code we try to show a three dimensional display that illustrates how the rigid body behaves. Since any inertia tensor is defined by the three diagonal elements in the principal axis frame, the ellipsoid representing the body can cover any possible case.

In Matlab it is possible to create a set of points defining the surface of an ellipsoid in one line. The various dimensions of the ellipsoid are scaled by the moment of inertia about that axis, so that if we considered the ellipsoid as a volume of constant density, the moments of inertia would be correct to a scale factor. This helps us to visualize how the rigid body actually behaves.

```
[xd,yd,zd]=ELLIPSOID(X(1),X(2),X(3),I1,I2,I3,numVertices);
```

To plot the resulting surface with a color map defined by `c` we use the `surf` command. In the file ”eulrig.m”, c is given by a matlab function that generates a matrix with a striped pattern, and in ”torque.m”, c is the texture map of the earth.

```
surf(x,y,z,c);
```

In order to rotate the ellipsoid to point in the correct direction to represent the orientation of the rigid body, we multiply each point in the original surface by the rotation matrix taking the body coordinate system to the world system. As described earlier, this is given by the formula converting a quaternion to a rotation matrix. Each time step, the program finds and plots the new rotated surface points.

```
points = [ xd(:), yd(:), zd(:) ];
R = quatToMat( Q );
```

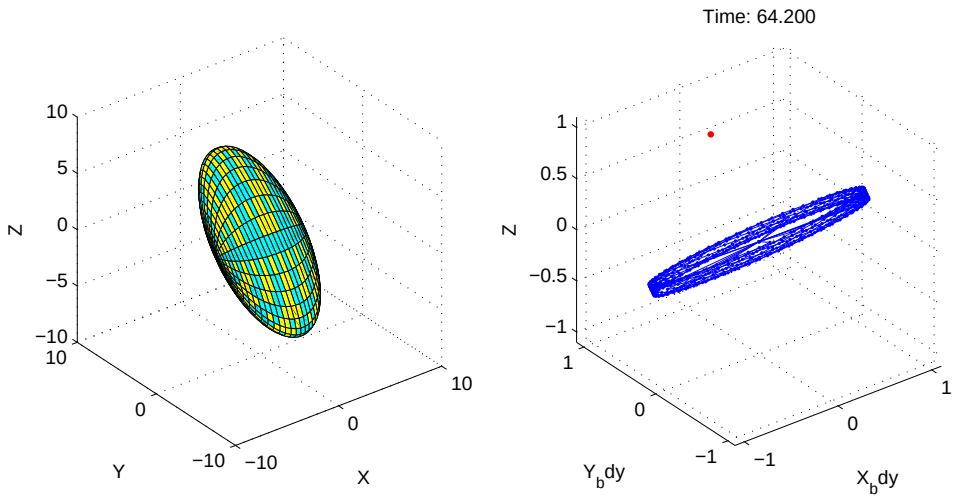

Figure 4: The asymmetrical top, with $I = [2\ 8\ 4]$ and initial $\omega = [0\ 1\ 0.1]$ The solid blue dots show the evolution of the point on the top of the top. The resulting path is perturbed from the true path.

```
points = (R * points')';
x = reshape( points(:,1), md, nd );
y = reshape( points(:,2), md, nd );
z = reshape( points(:,3), md, nd );
surf(x,y,z,c);
```

6 Discussion

The simulator correctly predicts the solution for the cases that have analytical solutions. Moreover, the simulator has qualitatively the correct behavior for the one other possible case, that is of a completely asymmetrical top. As we know from analysis of the Euler equations, a top with all different inertia values rotates stably about either its minor or major principal axis (Figure 4), but rotates unstably about the principal axis with the median moment.(Figure 5) This means that if the body is initially set to rotate about its major or minor principle axis, small perturbations will not be amplified. Rather, the body will just oscillate about the original axis about which it was set to rotate. However, if the body is rotating about its middle principle axis, small perturbations will be amplified. You can test this your self by throwing an eraser in the air with an initial spin about each axis.

7 Torque

To investigate the behavior of a normal top, torque was added into the model. The Euler equation with torque is:

$$\hat{\tau_w} = \dot{\ell} + \Omega \times \hat{\ell}$$

The vectors ℓ, τ and Ω are all expressed in the body frame basis. While the program cannot automatically calculate the torque on a rigid body, it can take a function that gives the torque as a function of orientation and time in the body frame. (If the torque is only known in the world frame, it is trivial to rotate it into the body frame by using the rotation matrix given by the quaternion.) A simple case is that of a symmetrical top (two moments equal, one greater) with rockets on the top and bottom edge, pointing in opposite directions and applying equal and opposite forces proportional on the top. This means that the center of mass of the top remains fixed because the net force on the top is zero.

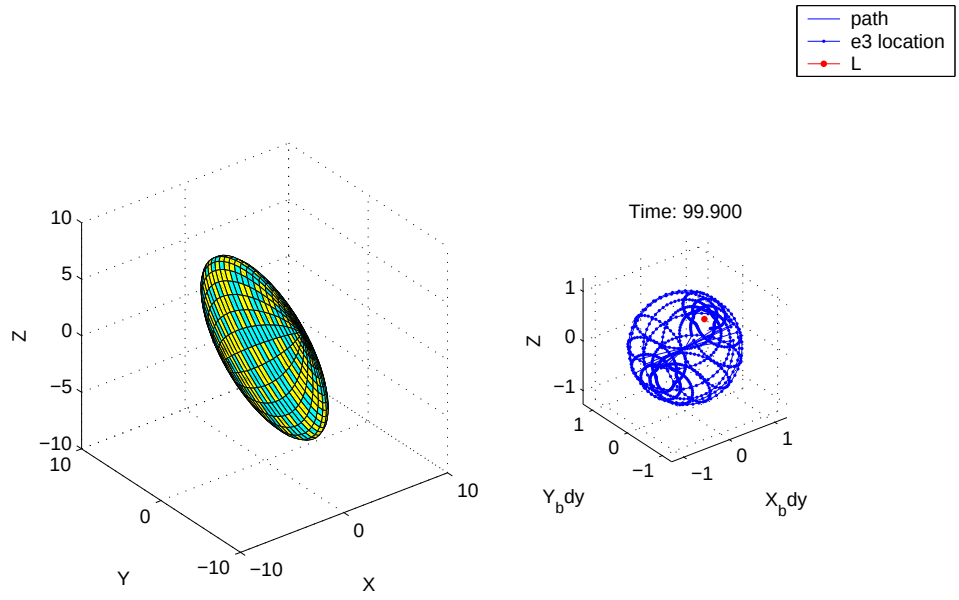

Figure 5: The asymmetrical top, with $I = [2\ 8\ 4]$ and initial $\omega = [0\ .1\ 1]$. The solid blue dots show the evolution of the top of the assymetrical top, defined by the $\hat{z}$ in the body frame rotate into the world frame. The red dot is the angular momentum, which stays fixed.

However, the top feels a net torque in the direction of the $\hat{z} \times x_3$ This is very similar to the case of a top under the influence of gravity (in fact, the forces are chosen to be proportional to the $\sin(\theta)$, where θ is angle of the inclination). The code is very similar to the code with no torque, except that in the Runge-Kutta code, we must recalculate the torque each time we find a new estimate of the orientation, because the torque is a function of orientation. The only other change is to the code calculating the derivative of the angular velocity which now includes the applied torque correction.

```
function [Wdot] = getWdot( W, T, I )
Imat = diag( I);
Imat_inv = diag( 1 ./ I );
Wdot = Imat_inv*(T+cross(Imat*W,W ) );
```

Figure (6) attempts to illustrate how the top behaves. As the diagram shows, the angular momentum vector moves in a cone about the $\hat{z}$ axis. The main axis of the top runs circles around angular momentum vector as it precesses about the $\hat{z}$ axis. This is what one would expect and matches the qualitative descriptions given in several mechanics textbooks.

8 Conclusion

The behavior of a body undergoing rigid-body rotation is surprisingly chaotic and complicated. We have explained a simulator for rigid-body dynamics that exhibits the behaviors we expect. The model seems to be very accurate since it uses quaternions and fourth order Runge-Kutta integration, yet quickly enough to see the animation in real time. The graphics of the simulator make it possible for the user to gain intution about how rigid bodies behave. We added simple implementation of torque to also be able to visualize the gyroscopic effects exhibited by simple tops.

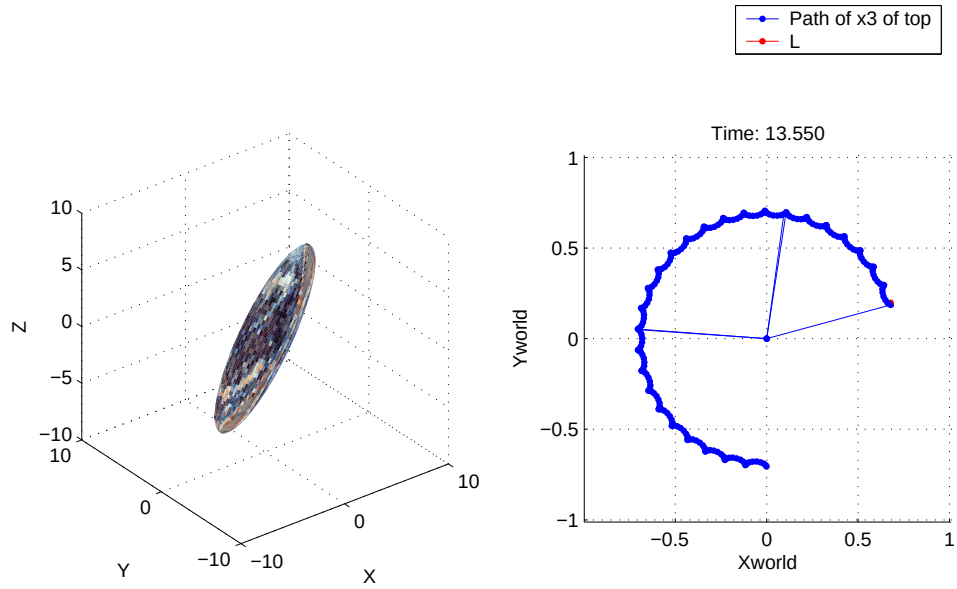

Figure 6: Symmetrical top with an applied $\tau = \hat{z}_{world} \times \hat{x3}_{top}$ There are visible oscillations as expected.

References

[1] L.D. Landau and E.M. Lifshitz. *Course of Theoretical Physics Volume 1*. Butterworth-Heinenann, 1960. Chapter 9.

[2] John R. Taylor. *Classical Mechanics*. University of Colorado, 2004. Chapter 9.

[3] www.anticz.com/eularqua.htm.