Simulation of Gravity Gradient Stabilization of a Satellite

Billy Fung, Ryan Yee, Kyle Boone, Michael Caverley, Alec Douglas[1]
Engineering Physics, University of British Columbia, Vancouver BC, Canada

Gravity gradient stabilization of low-Earth-orbit satellites provides passive orientation without consuming large amounts of power. A satellite using gravity gradient stabilization will be able to orient itself by using the Earth's gravitational force and the moments of inertia experienced by the extended boom as it orbits around the Earth. A simulation was built in order to solve and model rotations of rigid bodies numerically. Our results show gravity gradient stabilization as a feasible option to control the attitude of a satellite such that it is pointed towards the Earth, the stabilization occurs within 10 orbits, which is very fast.

I. INTRODUCTION

Attitude control systems in nanosatellites play an important role in orienting the satellite in the correct direction. After using sensors to determine 3 angles of orientation that are needed to safely determine the amount of torque needed to orient the satellite, active components such as thrusters can be used to control attitude. Using active attitude control systems requires a considerable amount of power, and requires many advanced calculations.[4]

Gravity gradient stabilization systems have been studied and tested. Most satellites make use of active attitude control systems with torquers and thrusters. The advantages of using active systems to provide torque to the satellite is that the characteristics of the control is more accurate at providing quick changes to the orientation of the satellite in any desired orientation. But such active systems require large power draws and on satellite designs, the power distribution is very limited. A gravity gradient stabilization method passively orients the satellite without power draw, and also has a longer lifetime due to the fact that it does not involve any electrical or mechanical components that may fail.[5] Gravity gradient stabilization also does not require computational algorithms, as needed in active systems.

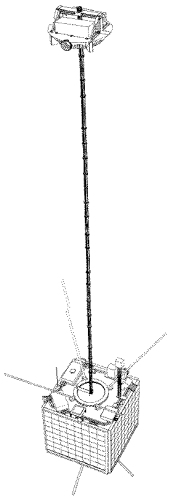

FIG. 1. Gravity Gradient Satellite

Gravity gradient stabilized satellites are theoretically able to passively orient the satellite by having a satellite with the mass distributed such that the centrifugal forces balance with the gravitational forces from the Earth as shown in FIG.1. The force of gravity can be found by:

$$F_{12} = -\frac{GMm}{|r_{12}|^2}\hat{r}_{12} \tag{1}$$

with G being the universal gravitational constant, M being the mass of the Earth, m being the mass of the satellite, and r begin the distance between the satellite and the Earth. An elongated satellite will experience a larger gravitational force at one end than the other. The resulting torque pulls the principle axis of inertia with the smallest principle moment in line with the gravitational force. The focus of our project was to model and simulate a satellite in a low-Earth orbit with gravity gradient stabilization. We numerically simulated the dynamics of a rigid body as it undergoes gravity gradient stabilization.

II. GRAVITY BOOMS

The design of our gravity boom is modelled as an elongated rigid rod. By having an elongated rod, the gravitational forces experienced at each end is different, causing a larger centrifugal force to be applied to the end further away from the Earth, and a larger gravitational force at the end closer to the Earth. As the satellite orbits the Earth, the gravitational and centrifugal forces will cause the satellite to naturally align itself such that the principle axis of inertia with the smallest primary moment is along the vector from the center of mass of the Earth to the center of mass of the satellite.

The gravitational force acting on a mass orbiting the Earth is given by:

$$dF = \frac{Gm}{r^3}(r_c + \rho)dm. \tag{2}$$

And the position vector to the center of mass being expressed as $r_c = r_c(l_{xr}i + l_{yr}j + l_{zr}k)$.

$\rho = xi + yj + zk$ expresses the vector from the center of mass of the earth to a small mass on the satellite, dm.

The torque is found by integrating 2:

$$M_c = -\mu \int_m \frac{(\rho \times r_c)dm}{r^3}. \quad (3)$$

These equations are based off the assumption that the orbiting body is a rigid body, which is not always the case in terms of gravity booms, when flexible components are introduced.

The torque on an object due to a gravity gradient must be derived in terms of the rotation and position of the object. For this derivation, $\vec{r}$ will be defined as the vector from the center of mass of the object to a given point. h will be defined as the distance from the center of the planet around which the object is rotating. The coordinate system used will have $\hat{y}$ aligned with the vector from the planet's position to the object's position, and $\hat{x}$ and $\hat{z}$ as two arbitrary vectors in the plane perpendicular to $\hat{y}$. Using this coordinate system, $\vec{r} = x\hat{x} + y\hat{y} + z\hat{z}$ and the vector from the center of the planet to the center of mass of the object is given by $h\hat{y}$.

The torque experienced by the object due to a force $\vec{F}$ applied at a point at a distance $\vec{r}$ from its center of mass is

$$\vec{T} = \vec{r} \times \vec{F} \quad (4)$$

The force in this case is gravity. Assuming that the changes in gravity due to small changes in the $\hat{x}$ and $\hat{z}$ directions are negligible, the above equation becomes:

$$d\vec{T} = \vec{r} \times \frac{GMdm}{(h+y)^2} h\hat{y} \quad (5)$$

where dm is the amount of mass at radius $\vec{r}$ from the center of mass of the object, and dT is the torque caused by this section of mass.

Letting the density of the object at point $\vec{r}$ be $\rho(\vec{r})$ and integrating over the entire object, the above equation simplifies to:

$$T_x = GM \int \frac{z}{(h+y)^2} dm \quad (6)$$

$$T_y = 0 \quad (7)$$

$$T_z = -GM \int \frac{x}{(h+y)^2} dm \quad (8)$$

For $y << h$ so, the above equations can be simplified using Taylor expansion. For the x-component of torque T_x:

$$T_x = GMh \int \frac{z}{h^2} \left(1 - \frac{3y}{h}\right) dm \quad (9)$$

$$= \frac{GM}{h^2} \int z\rho(\vec{r})dV \quad (10)$$

$$+ \frac{-3GM}{h^3} \int yz\rho(\vec{r})dV \quad (11)$$

These equations can be simplified even further by looking at what they represent. The term labelled (10) is equal to 0 since it represents the integral over the center of mass, which equals zero. The resulting term (11) represents the inertial tensor. The resulting equations of gravity gradient torque are:

$$T_x = \frac{3GMI_{yz}}{h^3} \quad (12)$$

$$T_y = 0 \quad (13)$$

$$T_z = -\frac{3GMI_{xy}}{h^3} \quad (14)$$

By rotating the satellite, the reference frame must also be rotated and shifted accordingly such that the center of mass remains within the satellite. A transformation matrix is applied to change the rotating coordinate frame to the principal coordinate frame.

As the satellite oscillations approach a steady state due to the gravity gradient stabilization, there are variations in torque. These small variations torque add up slowly as time progresses, and by cause the torques to flip the satellite.[5] Such a situation will lead the satellite to destabilize, and then re-stabilize. In order to maintain the steady state, some form of dampening is added to the system. Dampening systems onboard actual gravity gradient satellites include viscous dampeners.

III. SIMULATION

In order to determine if gravity boom stabilization is a viable way of attitude control, a physics simulation software program was created in order to simulate the physics of the satellite while it is in orbit. In order to accurately simulate the dynamics of the satellite in orbit, the simulation uses full-scale, distance and mass parameters. The dynamics of the object in the simulation were calculated using Newton's laws of motion and rotation equations.

The simulation software is programmed using Java and utilizes the Processing integrated development environment. Processing was used because it handles user input and allows us to create 3D graphics. The simulation uses OpenGL in order to create a visually appealing 3D visualization. The source code for the simulation and compiled executable files are submitted along with this report. The source files have a file extension of .pde but can be opened using any text editor.

There were four bodies considered in the simulation: Earth, the Sun, the Moon and the satellite. Each body was given a mass, position, velocity an acceleration and in addition, the satellite was given a rotation parameter. The program performs all of the calculations required to update these parameters at each timestep Δt. Using Δt and the current acceleration, Newton's laws and the rotation equations can be numerically integrated in order to calculate the new parameters for each object. This method can be used to calculate both the orbit of the satellite and it's rotation. Both will be described below.

IV. ORBIT CALCULATION

The only forces considered in this simulation when calculating the orbit are gravitational forces. The reason behind this decision is that gravitational forces are much greater than the other forces present. Other forces like drag on the satellite due to the Earth's outer atmosphere are minimal compared to the magnitude of the gravitational force since the timespan of the simulation is only a few days. Also, only the gravitational forces between the Earth, the Sun, the Moon and the satellite are considered because the gravitational forces due to the other bodies in the solar system are negligible. Another assumption made is that the 4 bodies in the simulation can be considered as point particles when calculating the orbital position of the objects.

The position of objects in the simulation is computed by first calculating the acceleration applied to every object. Gravity is the only force being considered, so the force on each body can be calculated by:

$$\mathbf{a} = \sum_{i=1}^{n} \frac{G m_i \mathbf{r}}{r^3} \tag{15}$$

where G is the gravitational constant, m_i is the mass of the i'th other body and n is the number of bodies. We then used numerical methods in order to compute the velocity and position of the objects.

V. METHODS OF NUMERICAL APPROXIMATION

In order to simulate the motion of multiple objects being acted on by the gravitational forces between them, Newton's second law of motion was solved numerically to obtain both velocity and position using two well known finite difference methods – one of which we ultimately chose for use in the final version of the simulation.

During the early stages of development, Euler's method was used to provide a very rough approximate solution. Euler's Method is an iterative procedure for approximating the solution of an ordinary differential equation, and the main benefit of its use is that it is very simple to implement. Given a calculated acceleration at time t_{n}, the velocity and position after one time step, $t_{\mathrm{n+1}}$, is given by Euler's method to be the following:

$$v_{n+1} = v_n + a_n \cdot \Delta t \tag{16}$$

$$x_{n+1} = x_n + v_n \cdot \Delta t \tag{17}$$

where Δt is the duration of each time step[6]. However, this method has an accumulated error of order Δt, and hence is considered a first-order algorithm. To achieve the desired level of accuracy in the simulation, Euler's Method requires a small time step that demands more computing power than is available to us.

A drastic improvement in the accuracy of the simulation can be made by replacing the Euler Method with the fourth-order Runge-Kutta algorithm (RK4). Instead of calculating the derivative of acceleration using only the beginning of an interval (as the Euler Method does), RK4 approximates the derivative at time $t_{\mathrm{n+1}}$ by using a weighted average of four derivatives, one calculated at the beginning, two in the middle, and once again at the end of an interval. Each successive derivative uses the previous result to obtain a different value. Collecting these four derivatives yields the velocity and position for the next time step:

$$v_{n+1} = v_n + \frac{(k_{1v} + 2k_{2v} + 2k_{3v} + k_{4v})}{6} \tag{18}$$

$$x_{n+1} = x_n + \frac{(k_{1x} + 2k_{2x} + 2k_{3x} + k_{4x})}{6} \tag{19}$$

where

$$k_{1v} = a_n(x_n, v_n, t_n) \cdot \Delta t$$

$$k_{1x} = v_n \cdot \Delta t$$

$$k_{2v} = a(x_n + \frac{k_{1x}}{2}, v_n + \frac{k_{1v}}{2}, t_n + \frac{\Delta t}{2}) \cdot \Delta t$$

$$k_{2x} = (v_n + \frac{k_{1v}}{2}) \cdot \Delta t$$

$$k_{3v} = a(x_n + \frac{k_{2x}}{2}, v_n + \frac{k_{2v}}{2}, t_n + \frac{\Delta t}{2}) \cdot \Delta t$$

$$k_{3x} = (v_n + \frac{k_{2v}}{2}) \cdot \Delta t$$

$$k_{4v} = a(x_n + k_{3x}, v_n + k_{3v}, t_n + \Delta t) \cdot \Delta t$$

$$k_{4x} = (v_n + k_{3v}) \cdot \Delta t$$

The global error of RK4 is of order $(\Delta t)^4$, allowing our simulation to run with a reasonably sized time step while maintaining a high level of accuracy[7].

One possible improvement to the accuracy of the simulation can immediately be seen, though not entirely simple to implement – by using a higher order method of approximation, the error on both velocity and position could be further minimized. For example, Satellite Toolkit uses a seventh-order Runge Kutta algorithm in order to produce extremely accurate results.

VI. ROTATION SIMULATION

In order to determine the rotation of a satellite undergoing gravity gradient forces, a rigid body rotation simulation was created. This involved creating a system to numerically solve generic rigid body rotation problems given an initial angular momentum and rotation along with the torque at each timestep.

A. Encoding Rotation

A method of encoding the rotation of a generic object is required in order to implement a rigid body rotation simulation. Euler angles were initially used. An Euler angle rotation involves three subsequent rotations about an object's local axes. Euler angles are able to encode all possible rotations, but they are a non-linear representation of rotations which makes simulations using them extremely complicated.

To observe this phenomenon, consider Euler angle rotations of a_1, a_2 and a_3 about the local z, x' and z'' axes in that order where $0 \leq a_1 < 2\pi$, $0 \leq a_2 < \pi$ and $0 \leq a_3 < 2\pi$. Then to represent a rotation of b purely about the global z-axis, a_2 must be 0 so the local z-axis is unchanged throughout the three rotations. There are multiple values for a_1 and a_3 such that $a_1 + a_3 = b$ which are all a possible way of encoding this rotation with Euler angles. This phenomenon is called "gimbal lock" and corresponds to a singularity in the mapping of rotations. The above explanation was only for a $ZX'Z''$ Euler angle, but in general gimbal lock will occur for rotations about some axis for any implementation of Euler angles.

Now consider a situation where an object's rotation is specified by $a_1 = a_3 = 0$ and $a_2 > 0$ and the object has angular velocity about the x-axis (illustrated in FIG. 2). Then at some point, the rotation of the object will be aligned with the z-axis. From our choice of Euler angle definitions, $0 \leq a_2 < 2\pi$, so passing this point will require the rotation to be encoded as $a_1 = \pi$, $a_2 > 0$ and $a_3 = 0$. Thus near the singularity, the derivative a_1' is undefined. Similarly, for changes in rotation near this singularity, a_1' will be very large.

If infinitesimally small timesteps are taken, this large change in a_1 can be handled if some additional care is taken when object's rotation is purely about the z-axis. However, for large timesteps, huge amounts of error will be introduced if an object's rotation is near these singularity points. To account for this, a simulation was implemented where rotation was added in very small timesteps near these singularities, up to thousands of timesteps per degree. This method worked (with additional handling for the actual singularity), but was found to still be relatively error prone and cause large slow downs when the object's rotation came near the singularities due to the number of computations being performed. As a result, Euler angles were removed from the simulation altogether and were replaced by unit quaternions.

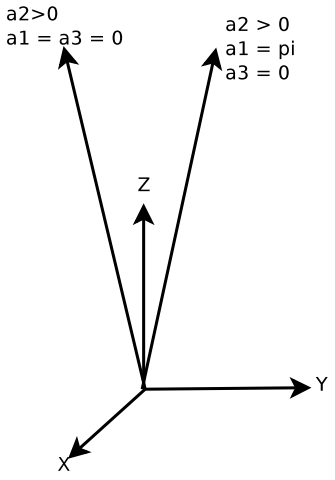

FIG. 2. Illustration of a rotation past a singularity of a set Euler angles

B. Quaternions

In order to explain how quaternions were used to represent rotation, a brief discussion of the mathematics behind them follows. A quaternion is a complex number of the form:

$$q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \tag{20}$$

where:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1 \tag{21}$$

Quaternions have many applications, one of which is representing rotations. To represent a rotation, unit quaternions are required. This implies that the norm of the quaternion is 1:

$$w^2 + x^2 + y^2 + z^2 = 1 \tag{22}$$

A quaternion representing a rotation of angle θ about an axis $\vec{u} = (a, b, c)$ is given by:

$$q = \cos(\theta/2) + \vec{u}\sin(\theta/2) \tag{23}$$

$$= \cos(\theta/2) + (a\mathbf{i} + b\mathbf{j} + c\mathbf{k})\sin(\theta/2) \tag{24}$$

where $\vec{u}$ is a unit vector.

The conjugate of a quaternion is defined as:

$$q^* = -w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \tag{25}$$

For quaternions representing rotation, this is equivalent to a rotation of the same magnitude about the same axis in the opposite direction.

Multiplying quaternions is defined as:

$$\begin{aligned} q_1 q_2 &= (x_1 + y_1\mathbf{i} + w_1\mathbf{j} + z_1\mathbf{k})(w_2 + x_2\mathbf{i} + y_2\mathbf{j} + z_2\mathbf{k}) \\ &= (w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2) \\ &+ (w_1x_2 + x_1w_2 + y_1z_2 - z_1y_2)\mathbf{i} \\ &+ (w_1y_2 - x_1z_2 + y_1w_2 + z_1x_2)\mathbf{j} \\ &+ (w_1z_2 + x_1y_2 - y_1x_2 + z_1w_2)\mathbf{k} \end{aligned} \tag{26}$$

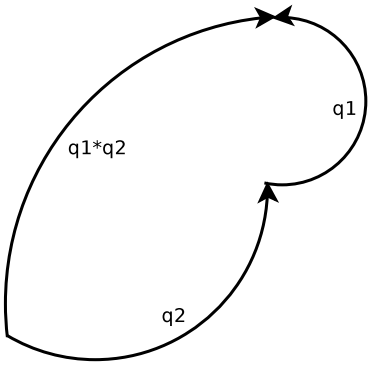

FIG. 3. Multiplication of rotations represented as quaternions

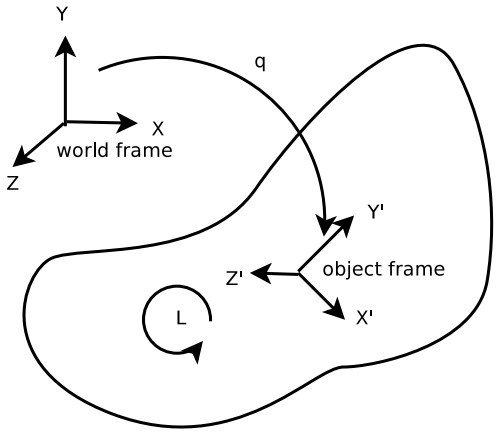

FIG. 4. World and object frame for an object

For quaternions representing rotations, quaternion multiplication is equivalent to applying the rotation represented by q_2 to an object then applying the rotation represented by q_1 to the same object in the rotated state. As can be verified using the above definition for rotation, quaternion multiplication is not associative so $q_1 q_2 \neq q_2 q_1$ in general. It is worth noting at this point that we are discussing "left quaternions" where quaternions chain right to left. An equivalent system can be derived using "right quaternions" which chain left to right. FIG. 3 illustrates this multiplication.

To apply the rotation represented by a quaternion to a vector, the following equation holds:

$$\vec{v'} = q\vec{v}q^* \tag{27}$$

where multiplying a quaternion by a vector is done by representing $\vec{v} = (a, b, c)$ as a quaternion of the form $0 + a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$ and applying quaternion multiplication.

A couple other properties of quaternions will also be used. The inverse and the conjugate of a quaternion are equal ($q^{-1} = q^*$). A multiplication of a quaternion by a scalar involves multiplying the quaternion term by term by the scalar. Finally, rotations represented by quaternions can be converted back and forth to rotation matrix, axis and angle or Euler angle forms.[8]

C. Rigid body rotation using quaternions

Using the properties of the quaternion derived in section VI B, rotational equations of motion for a rigid body can be derived. The physics principles required for these equations will now be detailed.

A rigid body has a moment of inertia tensor I associated with it which represents the ability of a body to resist rotation in a given direction. This inertia tensor is given by:

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{28}$$

where $I_{xy} = I_{yx}$, $I_{xz} = I_{zx}$ and $I_{yz} = I_{zy}$. The inertia tensor depends on the orientation of the body; the principal axes of a body are defined as the axes where $I_{xy} = I_{xz} = I_{yz} = 0$. Assuming that we have the inertia tensor for a given object along its principal axes and q is a quaternion representing a rotation of the principal axes of the body into a new position, the inertia tensor in that new rotation is found by:

$$I' = AIA^T \tag{29}$$

where A is the rotation matrix associated with the quaternion q.

In an inertial reference frame, the angular momentum L and the angular velocity ω are related by:

$$\omega = I^{-1}L \tag{30}$$

Angular momentum and torque τ are related by:

$$\tau = \frac{dL}{dt} \tag{31}$$

From angular velocity, the derivative of the quaternion q representing the current rotation can be calculated as follows:[9]

$$\dot{q} = \frac{1}{2}\omega q \tag{32}$$

Equations (28) through (32) can then be used to derive a system to determine the rotation of a body over time given some generic time dependent torque. The Euler method was used for this purpose. An inertial reference frame is required to use these equations. Thus some arbitrary axes are defined to be the "world frame". Additionally, the "object frame" is defined as some arbitrary frame of reference which rotates with the object. FIG. 4 illustrates these two frames. As a result, the inertia tensor I of an object will be constant in the object frame and can be defined as a constant related to the object. The inertia tensor I_w of the object in the world frame

will depend on the rotation of the object, and can be calculated using equation (29).

Let the state of the rotation of a system at a time t_i be represented by $\{L_i, q_i\}$ where these are the angular momentum and rotation of the system. The rotation q_i is defined to be the rotation from this world frame to the object's frame, and the angular momentum L_i is defined with components about the world frame's axes. If the state of the system $\{L_i, q_i\}$ at time t_i is given, along with a torque τ_i, the new state of the system at time t_{i+1} where $t_{i+1} = t_i + \Delta t$ can be approximated as follows:

First, the new angular momentum L_{i+1} is calculated. From equation (31), this can be approximated as:

$$L_{i+1} = L_i + \tau_i \Delta t \tag{33}$$

Using equation (30), the angular velocity can be calculated from L_{i+1}. In order to use this equation, the moment of inertia tensor in the world frame is required. This can be found using equation (29). q_i is the quaternion corresponding to a rotation from the world frame to the object's frame, so q_0^* represents the rotation that must be applied to the inertia tensor to rotate it into the world frame. If A_0 is the rotation matrix corresponding to q_0, the following equation can therefore be used to approximate angular velocity:

$$\omega_{i+1} = I_w^{-1} L_{i+1} \tag{34}$$

The above equation gives ω_{i+1} whereas ω_i is required in order to determine the added rotation from t_i to t_{i+1}. To compensate for this, ω_0 can be set to zero for the first calculation and ω_i can be calculated based on the state $\{L_{i-1}, q_{i-1}\}$ at time t_{i-1}.

The derivative of the quaternion representing the rotation can now be calculated using equation (32):

$$\dot{q_i} = \frac{1}{2} \omega_i q_i \tag{35}$$

And finally, the new rotation quaternion q_{i+1} can be approximated as follows:

$$q_{i+1} = q_i + \dot{q_i} \Delta t \tag{36}$$

At this point, the state $\{L_{i+1}, q_{i+1}\}$ at time t_{i+1} has been fully determined, and the calculations can be repeated to find further states.

The Euler method of calculating the next state is a very simple approximation of the rotational dynamics of the system, and does encounter multiple issues. First, it assumes that the angular velocity is equal to ω_i for the entire interval t_i to t_{i+1}. This is only reasonable for very small values of Δt, and the timestep in the simulation will have to be sufficiently small in order to compensate. Secondly, equation (36) will cause the norm of the quaternion q_{i+1} to drift slightly from 1. Smaller timesteps will help reduce this effect, but will not eliminate it entirely. To compensate for this, the quaternion must be occasionally normalized which involves simply dividing each term by the norm.

The above algorithm can be used to simulate the rotation of any generic rigid body. The only parameters that were not specified at this point are L_0, I and τ_i. L_0 represents the initial angular momentum of the object, and can be set in order to simulate an object with some initial angular velocity. The moment of inertia tensor I should remain constant throughout the simulation (in the object's frame), and can be varied in order to simulate the rotation of two different objects under the same conditions. τ_i represents the torque on the object at time t_i.

At this point in the derivation, the source of this torque has not been specified. To simulate gravity gradient stabilization, the torque equations derived in section II were used.

Additionally, a small amount of damping was added to simulate the effects described in section II. This was implemented by simply multiplying the angular momentum by a damping factor after every update.

VII. OPERATIONS MANUAL

In order to start the simulation, run the provided executable file. Once the program has started, you will be viewing the Earth and the simulation will be progressing forwards in time. You can control the speed of the simulation by dragging the slider at the bottom left part of the window. Drag the slider all the way to the left to pause the simulation and drag it all the way to the right to run the simulation at maximum speed. You can also change the object you are viewing by pressing the following keys:

Key	Look at
1	Earth
2	Moon
3	Satellite
4	Sun

You can click and drag the mouse to rotate your view about the object you are focusing on. You can also scroll in order to zoom in and out. If you click on the button which is located to the right of the slider, a window will open which displays a graph showing the rotation angle of the satellite. The rotation angle is the angle between the vector pointing from the center of mass of the satellite to the center of the Earth and the principle axis of the satellite with the lowest principle moment.

VIII. RESULTS AND OBSERVATIONS

The simulation demonstrates gravity gradient stabilization of a rigid body satellite. The gravity stabilization can be quantified by observing the rotation angles that the satellite makes as it orbits around the Earth

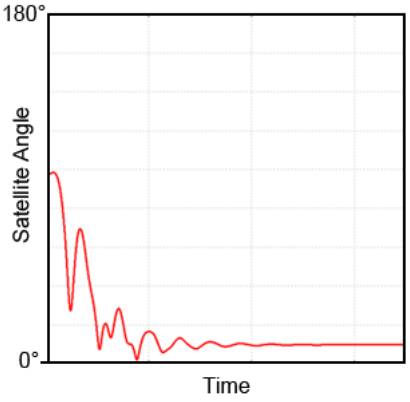

FIG. 5. Satellite angle (degrees) versus time for zero initial angular momentum, clearly showing a stabilization around 10. The time span shown in this graph is fewer than 8 orbits around the Earth.

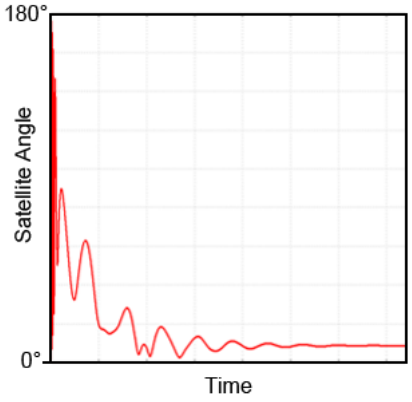

FIG. 6. Satellite angle (degrees) versus time for some non-zero initial angular momentum, eventually showing stabilization identical to that of FIG. 5. The time span shown in this graph is fewer than 10 orbits around the Earth.

over time. From the simulation, data for the satellite angle was collected over time spans showing the effects of gravity gradient stabilization, using various initial conditions. In every case, the satellite eventually came to be in a stable state, often remaining in a constant angle oriented towards the surface of the Earth. FIG. 5 shows the progression of the satellite angle over time from an unstable to stable state given zero initial angular momentum. FIG. 6 displays a similar result, but with some non-zero initial angular momentum. These results are consistent with what is expected of a satellite using a gravity boom.

Data from both FIG. 5 and FIG. 6 were collected from a perfectly circular orbit - to see the effects of an orbit with eccentricity greater than 0, FIG. 7 contains data collected from an orbit with an eccentricity of 0.2. Unlike the two previous cases, the eccentricity of the orbit results in a repeating pattern of angles. The satellite does not stabilize to a constant angle due to the nearly constant rotational velocity of the satellite coupled with the highly-variable orbital angular velocity of the satellite, which varies greatly as the satellite's distance from the Earth changes throughout the orbit.

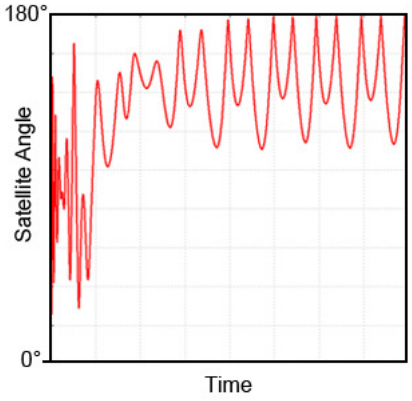

FIG. 7. Satellite angle (degrees) versus time for an orbit with an eccentricity of 0.2. The angle stabilizes in some repeating pattern.

IX. CONCLUSION

This project was successful in accomplishing the primary goal outlined in the Notice of Intent. A simulation was successfully created which modelled a satellite in orbit and its rotational dynamics. The simulation showed that a gravity gradient stabilized satellite does indeed tend towards a configuration with its principal axis with the smallest principal moment always aligned with the direction to the planet around which it is orbiting.

Implementing the simulation was quite difficult due to the complicated mathematics and equations required. The main issue was loss of accuracy which manifested itself in both the position and rotation calculations. Implementing a 4th order Runge-Kutta algorithm for position and using quaternions to encode rotation along with relatively small timesteps reduced this loss of accuracy to an acceptable level. However, there is no real data readily available to compare the results of the simulation to, so the actual accuracy of the simulation is not very well known.

The simulation was successful in simulating orbits with high eccentricities and resulted in interesting configurations as shown in FIG. 7. The equations for tidal locking of large bodies were derived, but were not implemented in this simulation due to time constraints and due to the complexity of the equations. The simulation was run with varying initial angular velocities and found to stabilize within a few orbits regardless of the initial velocity. Some comparisons with Satellite Tool Kit (a program commonly used in industry able to simulate many properties of a satellite) were done but not in enough detail for this report.

In order to improve upon the results of this project,

comparisons with other simulations would be required. For further analysis, comparison of this simulation's gravity gradient stabilization with actual data or programs such as Satellite Tool Kit would be required.

[1]Y.Chen, Z. Hong, C. Lin "Aerodynamic and Gravity Gradient Stabilization for Microsatellites,"Acta Astronautica Vol. 46, pp. 491-499, (2000)

[2]K. Alsaif, K. Al-Dakkan, Eng.Sci (1), Vol. 18, pp. 139-160 (2005)

[3]J. Ashenberg, E. Lorenzini "Active Gravity-Gradient Stabilization of a Satellite in Elliptic Orbits" Acta Astronautica, Vol. 45, No. 10, pp. 619-627, (1999)

[4]V. Piscane "Fundamentals of Space Systems," Oxford University Press, New York (1994)

[5]P.Hughes "Spacecraft Attitude Dynamics", John Wiley and Sons (1986).

[6]Austin, D., Keshet, L., Sjerve, D. (2010). Euler's Method. UBC Calculus Online Course Notes. Retrieved March 20, 2011, from UBC Mathematics Website: http://cello.math.ubc.ca/coursedoc/math100/

[7]Gould, H., Tobochnik, J., Christian, W. (2007). "An Introduction to Computer Simulation Methods: Applications to Physical Systems."(3rd ed.). San Francisco (CA): Pearson.

[8]J. Kuipers "Quaternions and Rotation Sequences" Princeton University Press, New Jersey (1999)

[9]I. Mandre "Rigid body dynamics using Euler's equations, Runge-Kutta and quaternions" retrieved April 9th 2011 from http://www.mare.ee/indrek/varphi/vardyn.pdf