Dylan Johnston | 1003852690
Nikhil Narayanan | 1000448465
Masters of Engineering | Aerospace
University of Toronto | UTIAS

Spacecraft Attitude Estimation

AER 1513
State Estimation for Aerospace Vehicles
December 21st, 2018

Contents

1 Introduction and Motivation **1**

2 Dynamics **1**
- 2.1 Reference Frames . . . 1
- 2.2 Dynamics of a Rigid Body . . . 2
- 2.3 Disturbance Torques . . . 2
  - 2.3.1 Gravity Gradient Torque . . . 2
  - 2.3.2 Residual Magnetic Dipole Torque . . . 2
  - 2.3.3 Solar Radiation Torque . . . 2
  - 2.3.4 Aerodynamic Drag Torque . . . 2
- 2.4 Attitude Dynamics . . . 3
- 2.5 Orbital Dynamics . . . 3

3 Simulation **3**

4 EKF Algorithm **4**
- 4.1 Motion and Observation Models . . . 4

5 Results **4**
- 5.1 Simulation and Data Set Creation . . . 5
- 5.2 Measuring Angular Velocities . . . 7
- 5.3 Including Magnetic Field Measurements . . . 8

6 Discussion **8**

7 Conclusion **8**

8 Matlab Code **10**

List of Figures

1 Keplerian orbital parameters. . . . 3
2 Error in ω_1 . . . 5
3 Errors in Euler Angle Evolution . . . 5
4 Evolution of angular velocity with time . . . 6
5 Evolution of Disturbance Torques with time . . . 6
6 Errors in angular velocity with time . . . 7
7 Euler Angle Evolution with time . . . 7

List of Tables

1 Keplerian Orbital Parameters . . . 3
2 Noise characteristics of the sensors used in the simulation, where r is a random number between 0 and 1. . . . 5

1 Introduction and Motivation

Spacecraft attitude and orbital state estimation is a necessary component in predicting and controlling a spacecraft's behaviour. In communications and global positioning functions, state estimation helps ensure that a satellite is pointed in the correct direction to relay signals, and in the event that it is not, on board actuators can create a torque to correct the satellite's attitude. State estimation is an important part of interplanetary orbital trajectories as well, with several important roles specifically during mid course corrections and gravity assist maneuvers. Many future space missions are designed around spacecraft formation flying technology, with applications such as stereographic imaging, long baseline interferometery, and synthetic aperture radar. In all of these applications, state estimation is imperative to ensuring the array of spacecrafts maintains its desired formation.

Estimating the state of an spacecraft can be quite challenging for a number of reasons. First, Euler's rigid body equation is a non-linear differential equation, and there are in general no closed form solutions, except for a few unique cases. However, most spacecraft are not rigid, due to the inclusion of antennae, solar panels, or moving components such as a the Canadarm, and these flexible modes greatly add to the complexity of the dynamic differential equation. Second, while the the computational power typically exists to numerically solve such differential equations, spacecraft are highly limited in their payload by constraints such as cost, weight, and energy use. This makes low weight, cheap, and efficient state estimators a crucial element of mission design.

Spacecraft state estimation is typically performed using on board sensors. Standard sensors include photoelectric sun sensors, angular rate gyroscopes, and, when the spacecraft is in orbit about a body with a magnetic field, magnetometers. A sun sensor is a type of passive remote sensing. Sunlight travels through a slit on board the spacecraft and falls on an array of photodetector cells, generating a voltage on the cells which are hit. This voltage is registered electronically, and if an array of two perpendicular sensors is used, the direction of the sun can be fully determined. One key drawback of using this type of sensor is that sensor data is only available while the spacecraft is not eclipsed by the body it is orbiting.

Angular rate gyroscopes measure angular rate directly based on the rotation of a spacecraft. Gyroscopes use Earth's gravity in order to determine orientation. The sensor consists of a freely rotating disk called a rotor mounted on a spinning axis in the center of a larger and more stable wheel. As the axis turns, the rotor remains stationary, indicating a rate of rotation around a particular axis. This generates a signal while the spacecraft is rotating, and gives zero signal if the spacecraft has no angular rate.

Magnetometers are a third type of sensor which use electromagnetic coils to determine attitude. Based on the Maxwell-Faraday equation, a changing magnetic field induces an electromotive force, which induces a current within the electromagnetic coils. The current through the coils is non-zero while the magnetic field is changing, which is the case during attitude rotation or over a longer period of time as the magnetic field changes based on orbital position.

The aim of this project is to use a suite of sensors, beginning with angular rate gyroscopes, and later including magnetometer measurements and finally sun sensors, to determine the attitude and angular rates of a spacecraft in orbit. An extended Kalman filter state estimator will be developed in order to determine it's effectiveness in determining the state of the spacecraft. The data that will be used for the various sensors in the problem will be generated synthetically using a numerical simulation.

In order to limit the scale of the problem, the orbital position of the spacecraft will not be estimated, but will be simulated in order to provide the information necessary to calculate the disturbance torques acting on the body. Along with the orbital position, the angular rate and attitude of the spacecraft will be simulated under the influence of disturbance torques using a fourth order Runge-Kutta method, and will be used as the ground truth. This data will then be corrupted by sensor noise, and will be fed into the extended Kalman filter which will try and determine the angular rate and attitude of the satellite without knowledge of the external disturbance torques.

2 Dynamics

2.1 Reference Frames

Several reference frames will be used in the mathematical description of this problem.

The Earth-centered inertial frame, $\boldsymbol{f}_I$, has its origin fixed to the center of mass of the earth, however this could be modified to any body that the spacecraft is orbiting. The 1-axis of this reference frame points towards the vernal equinox of the Earth's orbit.

The perifocal reference frame, $\boldsymbol{f}_O$, also has it's origin fixed to the center of mass of the Earth. The 1-axis of this frame points to the periapsis of the orbit, while it's 3-axis points parallel to the orbital angular momentum vector.

The orbiting body-fixed spacecraft frame, $\boldsymbol{f}_B$, has it's origin fixed to the center of mass of the spacecraft, while it's axes are typically oriented to align with the principle moments of inertia of the spacecraft.

2.2 Dynamics of a Rigid Body

All rotating rigid bodies follow Euler's equations for rotational dynamics:

$$\boldsymbol{u} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega}^{\times}\boldsymbol{I}\boldsymbol{\omega} \tag{1}$$

Which can be rearranged to form the system of non-linear differential equations needed to solve for the rotational kinematics:

$$\begin{aligned} M_1 &= I_1\dot{\omega_1} + (I_3 - I_2)\omega_2\omega_3 \\ M_2 &= I_2\dot{\omega_2} + (I_1 - I_3)\omega_3\omega_1 \\ M_3 &= I_3\dot{\omega_3} + (I_2 - I_1)\omega_1\omega_2 \end{aligned} \tag{2}$$

Finding solutions to this set of equations is difficult for two particular reasons; the first is that the equations are coupled and are nonlinear (products of $\omega_i(t)$ in the differential equations) and the second is that there is no knowledge of the reference frame in which the respective $\omega_i(t)$ are measured from. In the case where there is no control inputs or disturbances , $\boldsymbol{u} = \boldsymbol{0}$.

2.3 Disturbance Torques

In reality, spacecraft motion is not torque free and there are a number of external torques from the environment that contribute to disturbances to the spacecraft during its motion.

2.3.1 Gravity Gradient Torque

Gravity gradient torque acts typically on large orbiting bodies where the gravitational field experienced on each end of the spacecraft may differ thus causing a net torque. This is captured by:

$$\boldsymbol{G}_{gb} = \frac{3\mu_e}{r^5}\boldsymbol{R}_b^{\times}\boldsymbol{I}\boldsymbol{R}_b \tag{3}$$

Where $\boldsymbol{R}_b$ is the position of the spacecraft in earth's orbit, r is the norm of the position vector, and μ_e is the gravitational parameter for the earth.

2.3.2 Residual Magnetic Dipole Torque

The torque due to Earth's geomagnetic field acting on the residual magnetic dipole moment of the spacecraft is given by:

$$\boldsymbol{G}_{mb} = \boldsymbol{m}_b^{\times}\boldsymbol{B}_b \tag{4}$$

Where:
$\boldsymbol{R}_b$: position of spacecraft in earth's orbit.
$\boldsymbol{m}_b$: spacecraft residual magnetic dipole
$\boldsymbol{B}_b$: magnetic field at given position We will model the earth as a dipole, so that it can be described in the inertial frame as:

$$\boldsymbol{B}_I = \begin{bmatrix} (B_r\cos\delta + B_\theta sin\delta)\cos\alpha - B_\phi\sin\alpha \\ (B_r\cos\delta + B_\theta sin\delta)\sin\alpha + B_\phi\cos\alpha \\ B_r\sin\delta - B_\theta\cos\delta \end{bmatrix}$$

$$\begin{bmatrix} B_r \\ B_\theta \\ B_\phi \end{bmatrix} = \left(\frac{a_e}{r}\right)^3 \begin{bmatrix} 2g_1^0\cos\theta_m + (g_1^1\cos\phi_m + h_1^1\sin\phi_m)\sin\theta_m \\ g_1^0\sin\theta_m + (g_1^1\cos\phi_m + h_1^1\sin\phi_m)\cos\theta_m \\ g_1^1\sin\phi_m - h_1^1\cos\phi_m \end{bmatrix}$$

The constants used in this model are determined from literature.

2.3.3 Solar Radiation Torque

Solar radiation torques can have a non-negligible effect on the attitude of a spacecraft as well. Given the normal vectors describing the orientation of each surface, the center of pressure, which includes the wetted surface area of the spacecraft and the angle of incidence, can be calculated via

$$\hat{\mathbf{c}}_{psb} = \int\int_{A_w} \boldsymbol{\rho}_b\hat{\boldsymbol{n}}_{nb}^T\boldsymbol{C}_{BI}\boldsymbol{f}_{sI}dA \tag{5}$$

Once the center of pressure has been calculated, the torque due to solar radiation pressure can be calculated via:

$$\boldsymbol{G}_{sb} = \hat{\mathbf{c}}_{psb} \times \hat{\boldsymbol{F}}_{sb} \tag{6}$$

Where $\hat{\boldsymbol{F}}_{sb}$ is the force exerted on the spacecraft due to solar radiation pressure in the body-fixed frame.

2.3.4 Aerodynamic Drag Torque

Similar to the solar radiation torque, the aerodynamic drag torque can be calculated via a cross product of the spacecraft's center of pressure with the drag force. The spacecraft's center of pressure is, once again:

$$\hat{\mathbf{c}}_{pab} = \int\int_{A_w} \boldsymbol{\rho}_b\hat{\boldsymbol{n}}_{nb}^T\boldsymbol{C}_{BI}\boldsymbol{f}_{aI}dA \tag{7}$$

While the torque due to aerodynamic drag can be calculated via:

$$\boldsymbol{G}_{ab} = \hat{\mathbf{c}}_{pab} \times \hat{\boldsymbol{F}}_{ab} \tag{8}$$

Where $\hat{\boldsymbol{F}}_{ab}$ is the force exerted on the spacecraft due to aerodynamic drag in the body fixed frame.

2.4 Attitude Dynamics

The attitude of the spacecraft is given by its Euler angles: roll, pitch, and yaw:

$$\boldsymbol{\theta} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^T \tag{9}$$

Given a set of angular rates measured on board the spacecraft on say a rate gyroscope, the rate of change of these Euler angles can be expressed as:

$$\dot{\boldsymbol{\theta}} = \boldsymbol{S}^{-1}(\boldsymbol{\theta})\boldsymbol{\omega}_{iv} \tag{10}$$

Which can be integrated to solve for the individual Euler angles. The matrix $\boldsymbol{S}$ can be calculated via the following method:

The $\boldsymbol{S}$ matrix originates from Poisson's kinematical equation which relates angular velocities to the rate of change of the rotation matrix:

$$\boldsymbol{\omega}^\times = \dot{\boldsymbol{C}}\boldsymbol{C}^T \tag{11}$$

Using the chain rule we have:

$$\begin{aligned}\boldsymbol{\omega}^\times = -(&\dot{\boldsymbol{C}}_z\boldsymbol{C}_y\boldsymbol{C}_x + \boldsymbol{C}_z\dot{\boldsymbol{C}}_y\boldsymbol{C}_x + \\ &\boldsymbol{C}_z\boldsymbol{C}_y\dot{\boldsymbol{C}}_x)\boldsymbol{C}_x^T\boldsymbol{C}_y^T\boldsymbol{C}_z^T\end{aligned}$$

Using the orthogonality property of rotation matrices $\boldsymbol{C}\boldsymbol{C}^T = \mathbf{1}$ we can simplify the above:

$$\begin{aligned}\boldsymbol{\omega}^\times &= -(\dot{\boldsymbol{C}}_z\boldsymbol{C}_z^T) - \boldsymbol{C}_z(\dot{\boldsymbol{C}}_y\boldsymbol{C}_y^T)\boldsymbol{C}_z^T \\ &-\boldsymbol{C}_z\boldsymbol{C}_y(\dot{\boldsymbol{C}}_x\boldsymbol{C}_x^T)\boldsymbol{C}_y^T\boldsymbol{C}_z^T\end{aligned}$$

With a few more simplifications we get that:

$$\boldsymbol{\omega} = \dot{\theta}_3\mathbf{1}_z + \boldsymbol{C}_z\dot{\theta}\mathbf{1}_y + \boldsymbol{C}_z\boldsymbol{C}_y\dot{\theta}_1 = \boldsymbol{S}\dot{\boldsymbol{\theta}} \tag{12}$$

Which results in the desired equation:

$$\dot{\boldsymbol{\theta}} = \boldsymbol{S}^{-1}\boldsymbol{\omega}$$

2.5 Orbital Dynamics

A complete description of a spacecraft's orbit can be provided using the following quantities:

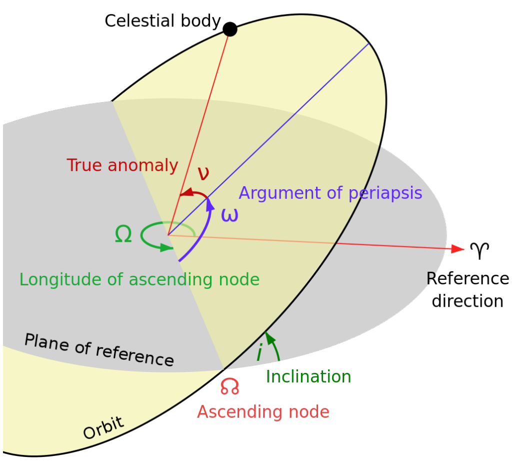

Figure 1: Keplerian orbital parameters.

a	Semi-major Axis	Longest diameter of an ellipse
e	Eccentricity	Deviation of a curve from circularity
i	Inclination	Vertical tilt of ellipse wrt reference plane
Ω	RAAN	Orients ascending node with vernal point
ω	Arg of Periapsis	Orientation of ellipse in orbital plane
ν	True Anomaly	Position of body on ellipse at given time

Table 1: Keplerian Orbital Parameters

All of the above orbital elements remain approximately constant for an orbiting body in the absence of external torques and forces, except for the True Anomaly, which can be solved for using the mean anomaly as follows:

$$M = \sqrt{\frac{\mu}{a^3}}(t - t_0) = E - e\sin(E) \tag{13}$$

Where E is the eccentric anomaly, and is solved for through an iterative process (Newton-Raphson method for example). The true anomaly is then solved for via:

$$\nu(t) = 2\arctan\left(\sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2}\right) \tag{14}$$

Once the orbiting body's true anomaly has been determined, its position and velocity vectors can be determined via:

$$\vec{\boldsymbol{r}} = \vec{\boldsymbol{F}}_I^T\boldsymbol{C}_{IO}\frac{a(1-e^2)}{1+e\cos(\nu)}\begin{bmatrix}\cos\nu\\ \sin\nu\\ 0\end{bmatrix} = \vec{\boldsymbol{F}}_I^T\boldsymbol{C}_{IO}\begin{bmatrix}r\cos\nu\\ r\sin\nu\\ 0\end{bmatrix} \tag{15}$$

$$\vec{\boldsymbol{v}} = \vec{\boldsymbol{F}}_I^T\boldsymbol{C}_{IO}\sqrt{\frac{\mu}{a(1-e^2)}}\begin{bmatrix}-\sin\nu\\ \cos\nu + e\\ 0\end{bmatrix} \tag{16}$$

Where:

$$\boldsymbol{C}_{IO} = \left[\mathbf{C}_3(\omega)\mathbf{C}_1(i)\mathbf{C}_3(\Omega)\right]^T \tag{17}$$

3 Simulation

The data set that was used for this project was simulated using a fourth order Runge-Kutta method. The simulation allows for the choice of 5 of the 6 orbital parameters, while the initial value for the true anomaly can be chosen by adjusting the starting time of the simulation. The simulation then calculates the position of the spacecraft at each time step, which is then used to determine the disturbance torques acting on the spacecraft.

The moment of inertia matrix, $\boldsymbol{I}$, can also be chosen prior to generating the dataset, however the moment of inertia matrix must be in the principle axis frame of the vehicle. The simulation script also allows for the choice of initial attitude and angular rates of the spacecraft, however, either the rates, or the time step, must be small in order for the small angle approximation used in the extended Kalman filter to apply.

Finally, the magnetic dipole moment of the spacecraft may also be chosen. The aerodynamic and solar radiation torques were not included in order to limit the complexity of the simulation, however they also amount to the smallest disturbance torques out of the four considered.

Once the orbital position is calculated for each time step, the disturbance torques are calculated, and then added to the motion model. The motion model is then propagated to determine the new angular rates, and the attitude of the spacecraft is calculated from the attitude dynamics described in section 2.4. Finally, the angular rates and attitudes output by the simulation are corrupted by zero mean Gaussian noise, so that the extended Kalman filter has an uncertain state to estimate.

4 EKF Algorithm

The assumption is made that the estimation process used in this project follows the Markov property, in that the future states depend only on the current state and not on the preceding states. The goal of the estimation algorithm is to compute the belief function for the state $\boldsymbol{x}_k$:

$$p(\boldsymbol{x}_k|\hat{\boldsymbol{x}}_0, \boldsymbol{v}_{1:k}, \boldsymbol{y}_{1:k}) \tag{18}$$

In general, a set of motion and observation models is used for the system with state $\boldsymbol{x}_k$, inputs $\boldsymbol{v}_k$ and measurements $\boldsymbol{y}_k$ and associated process and measurement noise of $\boldsymbol{n}_k$ and $\boldsymbol{w}_k$:

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{v}_{k-1}, \boldsymbol{w}_k) \\ \boldsymbol{y}_k &= \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{n}_k) \end{aligned} \tag{19}$$

The belief function is constrained to be Gaussian so that:

$$p(\boldsymbol{x}_k|\hat{\boldsymbol{x}}_0, \boldsymbol{v}_{1:k}, \boldsymbol{y}_{1:k}) = \mathcal{N}(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{P}}_k) \tag{20}$$

Where $\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{P}}_k$ are the mean and covariances respectively of the state. The centerpiece of the Extended Kalman Filter algorithm is the ability to linearize the motion and observation model about some operating point $\boldsymbol{x}_{op}$, which in this case will be the current state estimate mean $\hat{\boldsymbol{x}}_{k-1}$:

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{v}_{k-1}, \boldsymbol{n}_k) &\approx \check{\boldsymbol{x}}_k + \boldsymbol{F}(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1}) + \boldsymbol{w}'_k \\ \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{n}_k) &\approx \hat{\boldsymbol{y}}_k + \boldsymbol{G}_k(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k) + \boldsymbol{n}'_k \end{aligned} \tag{21}$$

Where $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{n}'$ and $\boldsymbol{w}'$ are the respective Jacobians of the motion, observation model with respect to the state and noise. From this point the statistical properties of the approximated motion and observation models are calculated.

$$\begin{aligned} E[\boldsymbol{x}_k] &= \check{\boldsymbol{x}}_k + \boldsymbol{F}(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1}) \\ E[\boldsymbol{y}_k] &= \hat{\boldsymbol{y}}_k + \boldsymbol{G}_k(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k) \end{aligned} \tag{22}$$

Where the last term is dropped since the noise is zero mean Gaussian. The remainder of the Kalman filter is propagated as described in the course notes:

$$\begin{aligned} \hat{\boldsymbol{P}}_k &= \boldsymbol{F}_{k-1}\hat{\boldsymbol{P}}_{k-1}\boldsymbol{F}_{k-1}^T + \boldsymbol{Q}'_k \\ \check{\boldsymbol{x}} &= \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{v}_k, \boldsymbol{0}) \\ \boldsymbol{K}_k &= \check{\boldsymbol{P}}_k\boldsymbol{G}_k^T(\boldsymbol{G}_k\check{\boldsymbol{P}}_k\boldsymbol{G}_k^T + \boldsymbol{R}'_k \\ \hat{\boldsymbol{P}}_k &= (\boldsymbol{1} - \boldsymbol{K}_k\boldsymbol{G}_k)\check{\boldsymbol{P}}_k \\ \hat{\boldsymbol{x}} &= \check{\boldsymbol{x}} + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{g}(\check{\boldsymbol{x}}_k, \boldsymbol{0}) \end{aligned} \tag{23}$$

4.1 Motion and Observation Models

The motion model of the system is given by:

$$\boldsymbol{f}(x_k, t_k) = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{\omega}) \\ \boldsymbol{f}(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}^{-1}(-\boldsymbol{\omega}^{\times}\boldsymbol{I}\boldsymbol{\omega}) \\ \boldsymbol{S}^{-1}(\boldsymbol{\theta})\boldsymbol{\omega} \end{bmatrix} \tag{24}$$

Where external torques have been included in the generation of the data set to test the robustness of the Extended Kalman Filter.

The observation model of the system is given by:

$$\boldsymbol{g}(y_k, t_k) = \begin{bmatrix} \boldsymbol{\omega}(t_k) \\ \boldsymbol{C}_{BI}(\boldsymbol{\theta}(t_k))\mathbf{B}_I(t_k) \\ \boldsymbol{C}_{BI}(\boldsymbol{\theta}(t_k))\mathbf{s}_I \end{bmatrix} \tag{25}$$

The Jacobian matrix of the motion model is given by:

$$\boldsymbol{F}(x_k, t_k) = \begin{bmatrix} \boldsymbol{I}^{-1}[(\boldsymbol{I}\boldsymbol{\omega}^{\times} - \boldsymbol{\omega}^{\times}\boldsymbol{I}] & 0 \\ \boldsymbol{S}^{-1}(\boldsymbol{\theta}) & \left[\frac{\partial \boldsymbol{S}^{-1}(\boldsymbol{\theta})}{\partial \theta_1}\boldsymbol{\omega} \quad \frac{\partial \boldsymbol{S}^{-1}(\boldsymbol{\theta})}{\partial \theta_2}\boldsymbol{\omega} \quad \frac{\partial \boldsymbol{S}^{-1}(\boldsymbol{\theta})}{\partial \theta_3}\boldsymbol{\omega}\right] \end{bmatrix} \tag{26}$$

The Jacobian matrix of the observation model is given by:

$$\boldsymbol{G}(x_k, t_k) = \begin{bmatrix} \boldsymbol{1} & 0 \\ 0 & \left[\frac{\partial \boldsymbol{C}_{BI}}{\partial \theta_1}\boldsymbol{B}_I \quad \frac{\partial \boldsymbol{C}_{BI}}{\partial \theta_2}\boldsymbol{B}_I \quad \frac{\partial \boldsymbol{C}_{BI}}{\partial \theta_3}\boldsymbol{B}_I\right] \\ 0 & \left[\frac{\partial \boldsymbol{C}_{BI}}{\partial \theta_1}\boldsymbol{s}_I \quad \frac{\partial \boldsymbol{C}_{BI}}{\partial \theta_2}\boldsymbol{s}_I \quad \frac{\partial \boldsymbol{C}_{BI}}{\partial \theta_3}\boldsymbol{s}_I\right] \end{bmatrix} \tag{27}$$

5 Results

In the results, two separate simulations were done. The first had no disturbance torques and measured angular rates only to obtain an estimate for angular rates and then Euler angles. The second simulation included disturbance torques and attempted to estimate Euler angles within the EKF algorithm as opposed to numerically integrating the estimated angular rates. The disturbance torques are given in Figure 5. First, the initial conditions and simulation methods are detailed below:

5.1 Simulation and Data Set Creation

The simulation was run with the following initial conditions, using a fourth order Runge Kutta method to integrate the Euler rigid body equation.

e	a (m)	i	ω	Ω	ν_0
0.5	1.47×10^7	$\frac{\pi}{4}$	$\frac{\pi}{8}$	$\frac{\pi}{6}$	0

Keplerian orbital parameters used to create data set. Angles are in radians.

The following were the initial conditions used in the simulation:

$$\boldsymbol{I} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 50 \end{bmatrix}$$

$$\boldsymbol{\omega_0} = \begin{bmatrix} 0 & 0.1 & 1 \end{bmatrix}^T$$

$$\boldsymbol{\theta_0} = \begin{bmatrix} 0.1 & 0.1 & 0 \end{bmatrix}^T$$

$$\boldsymbol{m}_b = \begin{bmatrix} 0.1 & 0.1 & 0.1 \end{bmatrix}^T$$

The generated dataset was then corrupted with zero mean Gaussian noise. The variance of the noise added to the magnetometer and angular rate sensors is summarized below.

	Angular Rate Sensor Noise	Magnetometer Sensor Noise
Formula	$\sqrt{12\sigma_\omega^2}[r - \frac{1}{2}]$	$\sqrt{12\sigma_m^2}[r - \frac{1}{2}]$
Variance	$2.79 \times 10^{-6} \frac{rad^2}{s^2}$	$2 \times 10^{-9} Teslas^2$

Table 2: Noise characteristics of the sensors used in the simulation, where r is a random number between 0 and 1.

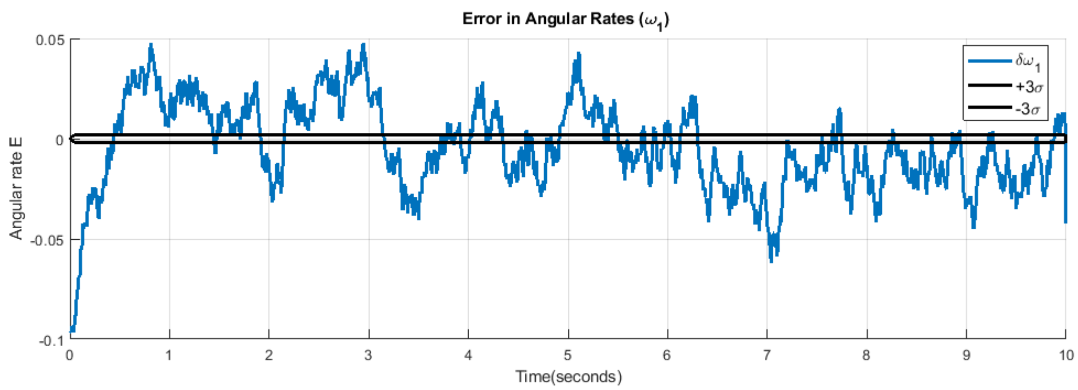

Figure 2: Error in ω_1

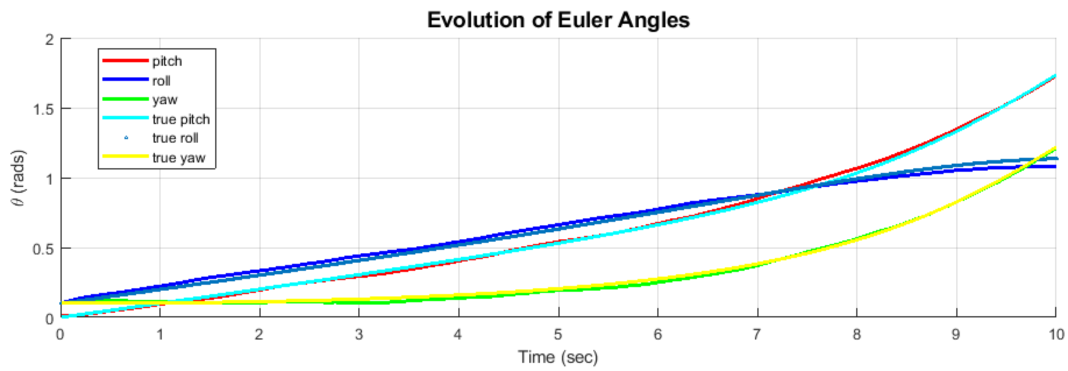

Figure 3: Errors in Euler Angle Evolution

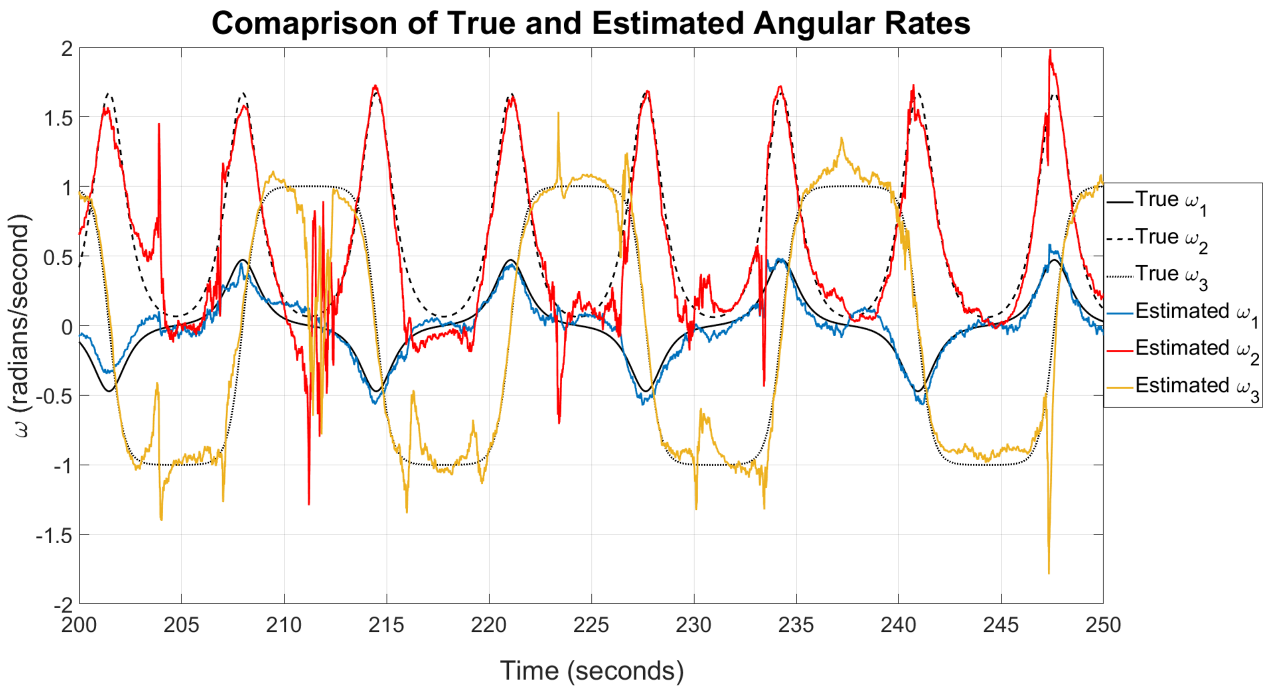

Figure 4: Evolution of angular velocity with time

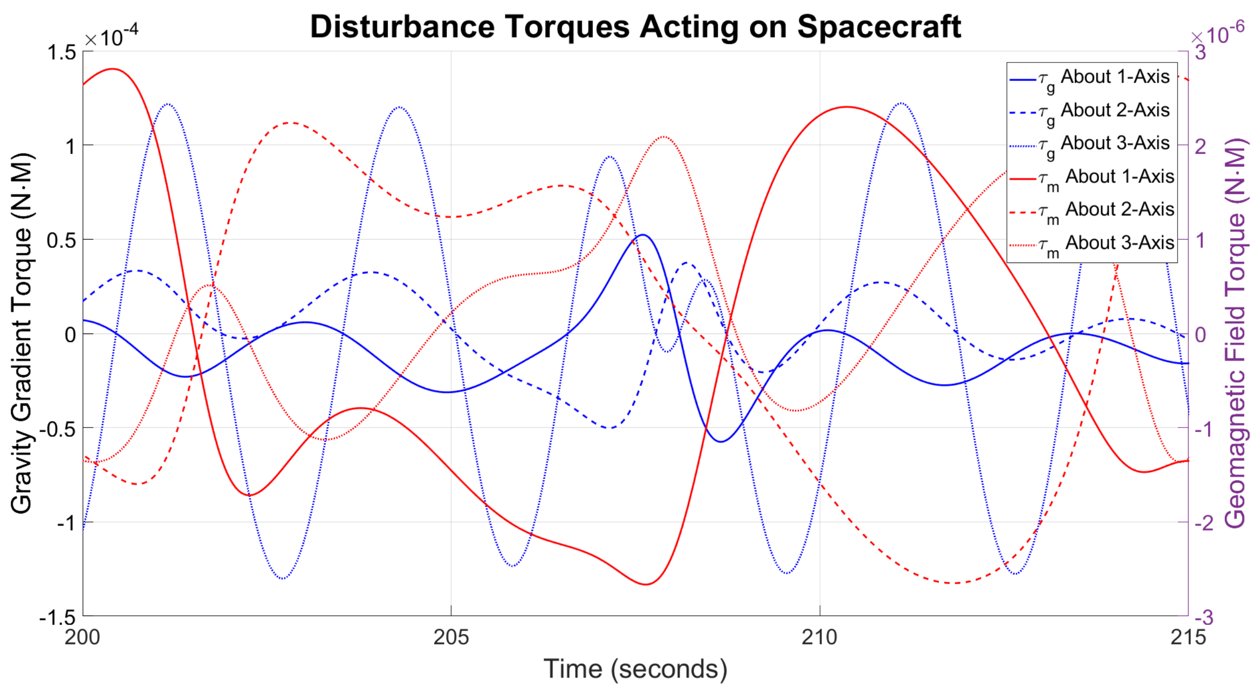

Figure 5: Evolution of Disturbance Torques with time

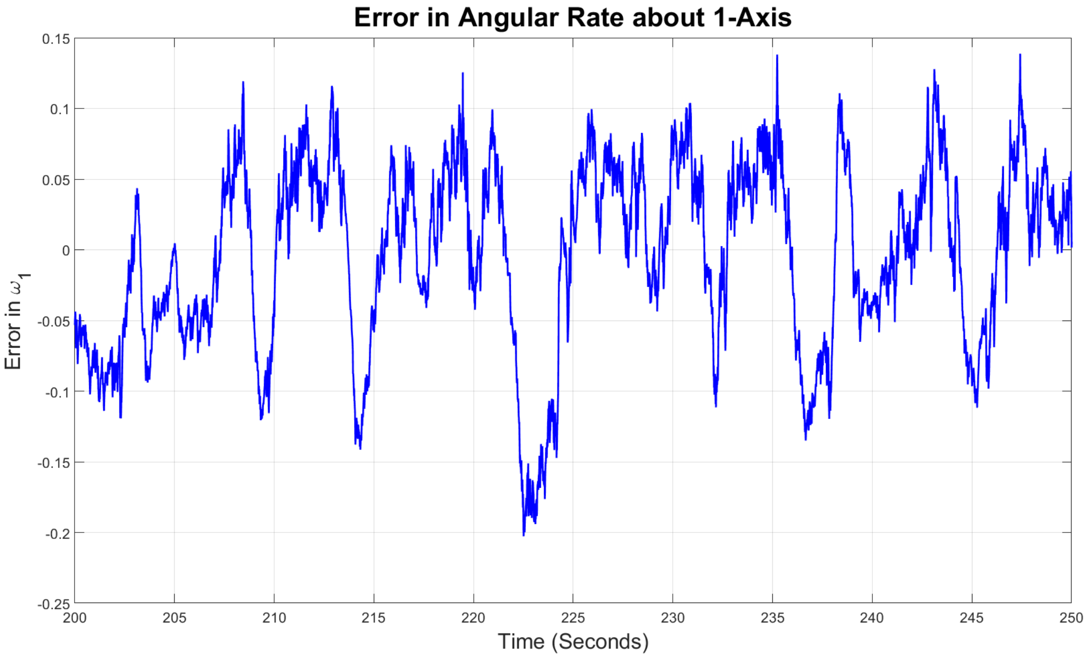

Figure 6: Errors in angular velocity with time

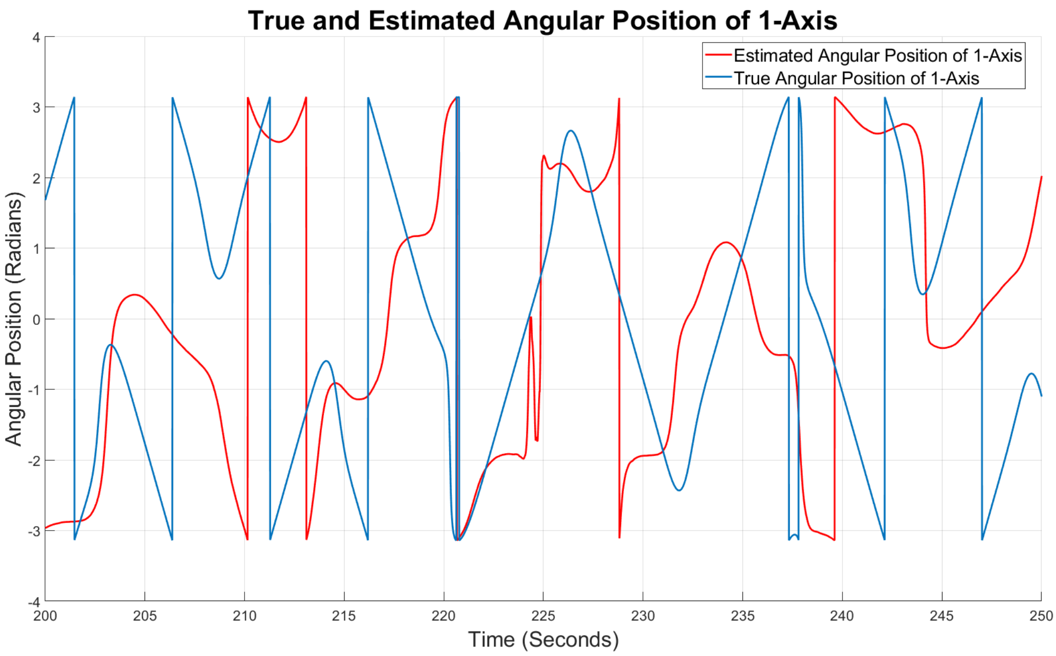

Figure 7: Euler Angle Evolution with time

5.2 Measuring Angular Velocities

In the first case, only angular velocities were measured i.e. the state estimation problem was a 3 state problem.

The angular velocities were obtained from measurements on board and then these were integrated to obtain a set of Euler angles from the initial conditions. The errors dynamics for the angular velocities are shown in Figure 2 and the Euler Angles are shown in Figure 3. As expected, the error dynamics in the angular rates recovers the noise associated with the angular rate measurements. Given the relatively low errors in the angular rates, the integration of these rates produces an accordingly low error in the Euler angle evolution. However, without a way to correct for the estimated attiude, the error in attitude grows unbounded.

5.3 Including Magnetic Field Measurements

Although the Euler angles were well tracked with reasonably small error, over time there would be an irrecoverable drift in the estimation of the Euler angles since, over the ten seconds of simulation, the algorithm is essentially dead reckoning. As such, the inclusion of the measurement of the magnetic field is necessary in order to fuse the integration of the angular rates with some external angle measurement. In this case, noise in the rate measurement (as well as the magnetic field) is associated with the measurement model. The noise in the process is used to capture the nature of the disturbance torques. The disturbance torques affect only the angular rates. The covariance matrices are:

$$\begin{aligned} \boldsymbol{Q} &= \begin{bmatrix} q\mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\ \boldsymbol{R} &= \begin{bmatrix} \sigma_\omega^2\mathbf{1} & \mathbf{0} \\ \mathbf{0} & \sigma_m^2\mathbf{1} \end{bmatrix} \end{aligned} \tag{28}$$

6 Discussion

When just the angular rates were measured, the values for the integrated Euler angles was accurate initially, due to the high degree of accuracy in the angular rates determined by the filter. However, since nothing was being fed back to correct these angles, over time the error in the accuracy of the Euler angles grows unbounded. This is a significant drawback from just using angular rate measurements unless the only objective of the filter is to determine the angular rates, which would be useful in an application such as spacecraft detumbling.

For the second simulation, the standard deviation in the error in angular rates is extremely small. This is likely due to the fact that the angular rates were propagated using a fourth order Runge-Kutta propagation scheme. Since this is the most accurate integration scheme available, its no surprise that the error in the angular rates is quite small. The $\pm 3\sigma$ bounds would be larger and fully encapsulate the error displayed if an integration scheme such as a first order Newton method was used.

Although the state of the Euler angles was attempted to be extracted in the second simulation, the results appear to be inaccurate and do not capture the true state to any functional degree. This issue likely originates from singularity issues in matrix inversion which occurs in the computation of the Kalman Gain matrix. The singularity arises from the fact that the magnetic field is very small (in the order 10^-9) and the covariance matrix for this magnetic field approaches MATLAB's floating point precision limit which led to badly conditioned matrices.

7 Conclusion

A data set describing the time evolution of the angular rates and attitude of a spacecraft was created. The spacecraft, under the effects of both gravity gradient and geomagnetic field disturbance torques, was propagated using a fourth order Runge-Kutta method. The data was then corrupted with noise and fed through an extended Kalman filter with the intention of estimating both the angular rates and attitudes of the spacecraft over the duration of the simulation period.

The filter was very accurate at estimating the angular rate of the spacecraft throughout the extent of the simulation, however, an accurate estimate of the attitude of the spacecraft could not be determined. Problems arose due to singularities that exist in the Euler angle-axis representation of rotational motion. The implementation of lie groups and lie algebra in order to bypass this issue could not be completed in time.

The extensive potential scope of this problem was the main incentive for its selection. The extended Kalman filter should be compared to other filters, such as the extended H-∞ filter, the nonlinear predictive filter, and some sort of batch method. In addition to different types of possible filters, this project could be extended to a full POSE problem, whereby the position of the spacecraft in its orbit is also uncertain and should be estimated, extending this problem from 6 degrees of freedom to potentially 9 or 12 degrees of freedom. The addition of a sun sensor to the suite of available measurements, the inclusion of both solar radiation and aerodynamic drag torques as disturbance torques on the vehicle, and the inclusion of plant errors which arise from uncertainty in the moment of inertia tensor are all ways of increasing the complexity of the problem. A final future work extension of this project is to develop a control system which takes the information provided by the state estimator and attempts to orient the vehicle in some desired way.

References

[1] Timothy D. Barfoot. *State estimation for robotics*. Cambridge University Press, 2017.

[2] Charles K. Chui and Guanrong Chen. *Kalman Filtering with Real-Time Applications*. Springer, 1991.

[3] Peter C. Hughes. *Spacecraft attitude dynamics*. W. Ross MacDonald School Resource Services Library, 2009.

[4] Andrew H. Jazwinski. *Stochastic processes and filtering theory*. Dover Publications, 2007.

[5] Anton de Ruiter. "Non-Linear State-Estimation for Spacecraft Attitude Determination". PhD thesis. 2001.

[6] James Richard Wertz. *Spacecraft attitude determination and control*. Kluwer Academic Publishers, 2002.

8 Matlab Code

```
close all;
clear all;
clc;

format long
% AER1513 State Estimation

% Final Project

% 3 Dimensional Rotation Simulation

% This script generates an angular rate dataset for a satellite, with moment of inertia
% matrix I, freely rotating in 3 dimensional space. Initially, zero torques
% will be applied to the satellite. Eventually, gravity gradient,
% aerodynamic, and solar radiation torques will be included.

%% Definition of Variables
global tstep
global t
global n
global tfinal
global eccentricity
global mu
global a
global ae

% Simulation variables
tstep = 0.01;
tfinal = 2000;
t = (0:tstep:tfinal);
n = length(t);

% Initialize Vectors
w1 = zeros(n,1); w2 = zeros(n,1); w3 = zeros(n,1);
r1 = zeros(n,1); r2 = zeros(n,1); r3 = zeros(n,1);
v1 = zeros(n,1); v2 = zeros(n,1); v3 = zeros(n,1);
pitch = zeros(n,1); roll = zeros(n,1); yaw = zeros(n,1);
TA = zeros(n,1);

% Initial Conditions
ae = 6371200; % radius of earth in meters
a = ae + 1000000; % semi major axis of orbit
eccentricity = 0;
inclination = pi/2; % Degrees
RAAN = 0; % degrees
omega = 0; % degrees
tp = 0; %seconds
mu = 3.986*10^14;
I = [100,0,0;0,10,0;0,0,50]; % Moment of inertia in body fixed frame
rconst = sqrt(mu/(a*(1 eccentricity^2)));
```

```

magdip = [0.1;0.1;0.1]; % Magnetic dipole moment of spacecraft in Amperes * meters^2

w1(1) = 0; w2(1) = 0.1; w3(1) = 1; % Initial angular rates
pitch(1) = 0.1; roll(1) = 0.1; yaw(1) = 0; % initial attitude
r1_0 = a; r2_0 = 0; r3_0 = 0; % Initial orbital position
TA(1) = 0; % Initial true anomaly in radians
v1(1) =  rconst * sin(TA(1)); v2(1) = rconst * (eccentricity + cos(TA(1))); v3(1) = 0; % Initial orbital velocity in m/s

% Define initial rotation matrices
CBI = Rotation321(roll(1),pitch(1),yaw(1))';
CIO = Rotation313(omega,inclination,RAAN)';
CIB = CBI';

sig = (100*0.00016*pi/180)^2*10^5; % the standard deviation of the corruption noise

% Euler's RBE in 3 separate equations
dw1 = @(t,w1,w2,w3,D)  ((I(3,3)   I(2,2))/I(1,1))*w2*w3 + D;
dw2 = @(t,w1,w2,w3,D)  ((I(1,1)   I(3,3))/I(2,2))*w1*w3 + D;
dw3 = @(t,w1,w2,w3,D)  ((I(2,2)   I(1,1))/I(3,3))*w2*w1 + D;

% Rate of change of orbital position
dr1 = @(t,TA)  rconst * sin(TA);
dr2 = @(t,TA) rconst * (eccentricity + cos(TA));
dr3 = @(t,TA) 0;

% Initialize Runge Kutta Intermediate Values
kw1 = zeros(1,4);
kw2 = zeros(1,4);
kw3 = zeros(1,4);

b = [1 2 2 1];  %RK4 weighting coefficients

%% Simulation

% Runge Kutta fourth order simulation

for i = 1:(n 1)
% Update Satellite Position
    TA(i) = 2*atan(sqrt((1+eccentricity)/(1 eccentricity))*tan(EccentricAnomaly(i)/2));
    radius(i) = a*(1 eccentricity^2)/(1+eccentricity*cos(TA(i)));

    r1(i) = radius(i)*cos(TA(i));
    r2(i) = radius(i)*sin(TA(i));
    r3(i) = 0;
    R = CIO*[r1(i);r2(i);r3(i)]; % Position of spacecraft in inertial frame R_I

    v1(i) = dr1(t(i),TA(i));
```

```
    v2(i) = dr2(t(i),TA(i));
    v3(i) = dr3(t(i),TA(i));

% Create Disturbance Torques
% % Geomagnetic Dipole Torque
    BField(i,1:3) = (CBI*CalculateMagneticField(R,i))';
    BField(i,4) = norm(BField(i,1:3));

    Tmag = SkewSymmetric(magdip)*CBI*CalculateMagneticField(R,i);
% % Gravity Gradient Torque
    Tgrav = ((3 * mu)/norm(R)^5) * SkewSymmetric(CBI*R)*I*(CBI*R);

    Disturbances = Tmag + Tgrav;
    D(:,:,i) = [Tmag,Tgrav];

% Update Satellite Angular Rates
    kw1(1) = dw1(t(i),w1(i),w2(i),w3(i),Disturbances(1));
    kw2(1) = dw2(t(i),w1(i),w2(i),w3(i),Disturbances(2));
    kw3(1) = dw3(t(i),w1(i),w2(i),w3(i),Disturbances(3));

    kw1(2) = dw1((t(i) + (tstep/2)), (w1(i) + (tstep/2)*kw1(1)), (w2(i) +
        (tstep/2)*kw2(1)), (w3(i) + (tstep/2)*kw3(1)),Disturbances(1));
    kw2(2) = dw2((t(i) + (tstep/2)), (w1(i) + (tstep/2)*kw1(1)), (w2(i) +
        (tstep/2)*kw2(1)), (w3(i) + (tstep/2)*kw3(1)),Disturbances(2));
    kw3(2) = dw3((t(i) + (tstep/2)), (w1(i) + (tstep/2)*kw1(1)), (w2(i) +
        (tstep/2)*kw2(1)), (w3(i) + (tstep/2)*kw3(1)),Disturbances(3));

    kw1(3) = dw1((t(i) + (tstep/2)), (w1(i) + (tstep/2)*kw1(2)), (w2(i) +
        (tstep/2)*kw2(2)), (w3(i) + (tstep/2)*kw3(2)),Disturbances(1));
    kw2(3) = dw2((t(i) + (tstep/2)), (w1(i) + (tstep/2)*kw1(2)), (w2(i) +
        (tstep/2)*kw2(2)), (w3(i) + (tstep/2)*kw3(2)),Disturbances(2));
    kw3(3) = dw3((t(i) + (tstep/2)), (w1(i) + (tstep/2)*kw1(2)), (w2(i) +
        (tstep/2)*kw2(2)), (w3(i) + (tstep/2)*kw3(2)),Disturbances(3));

    kw1(4) = dw1((t(i) + tstep), (w1(i) + tstep*kw1(3)), (w2(i) + tstep*
       kw2(3)), (w3(i) + tstep*kw3(3)),Disturbances(1));
    kw2(4) = dw2((t(i) + tstep), (w1(i) + tstep*kw1(3)), (w2(i) + tstep*
       kw2(3)), (w3(i) + tstep*kw3(3)),Disturbances(2));
    kw3(4) = dw3((t(i) + tstep), (w1(i) + tstep*kw1(3)), (w2(i) + tstep*
       kw2(3)), (w3(i) + tstep*kw3(3)),Disturbances(3));

    w1(i+1) = w1(i) + (tstep/6)*sum(b.*kw1);
    w2(i+1) = w2(i) + (tstep/6)*sum(b.*kw2);
    w3(i+1) = w3(i) + (tstep/6)*sum(b.*kw3);

% Update Satellite Orientation
    Cupdate = newC([w1(i+1);w2(i+1);w3(i+1)]);
    Cnew = Cupdate*CIB;

    theta_true(i,:) = ExtractEuler(Cnew);
    CIB = Cnew;
    CBI = CIB';

```

```
    if mod(i,1000) == 0
        CurrentTime = i/100
    end
end

% Stack vectors into a single matrix
W = [w1,w2,w3];
P = (CIO*[r1,r2,r3]')';
V = [v1,v2,v3];
O = theta_true;

% Add Noise

for i = 1:n
    w1n(i) = w1(i) + 10*sqrt(12*sig)*(rand(1)   0.5);
    w2n(i) = w2(i) + 10*sqrt(12*sig)*(rand(1)   0.5);
    w3n(i) = w3(i) + 10*sqrt(12*sig)*(rand(1)   0.5);
end

Wn = [w1n',w2n',w3n'];

%% Plotting

% Plot Noisy Angular Rates
figure(1)
hold on
plot(t,Wn(:,1),'r  ',t,Wn(:,2),'b',t,Wn(:,3),'g  ')
hold off

% Plot Orbit
figure(2)
hold on
plot(P(:,1),P(:,2));
hold off

% Plot Attitude
```

```
close all
clear all
clc

% Full POSE problem
global tstep
global t
global n
global tfinal

run('ThreeAxis.m');

%% Main
fprintf('Running main\n');
x = zeros(n,6);

%q =   0.5^2;
```

```

q = (100*0.00016*pi/180)^2;
r = (100*0.00016*pi/180)^2*10^2;

Q_omega = q*eye(3);  % [diag(q)];


Q = [Q_omega, zeros(3,3);
      zeros(3,3), zeros(3,3)];

R_omega = r*eye(3); % [diag(r)];
R_theta = 10000*2e 9^2 * eye(3);

R = [R_omega, zeros(3,3);
     zeros(3,3), R_theta];

P0 = diag([1,1,1,1,1,1]); % 6 states
xhat0 = [0,0.1,1, 0, 0.1, 0.1]; %W(1,:);

xhat1 = xhat0;
P1 = P0;

ROTS = zeros(3,3*n); % save rotation matrices
B = BFieldN; %magnetic field as measured on board


theta_0 = [0.1; 0.1; 0]; % angle with resepct to the inertial frame
C0 = Cx(theta_0(1))*Cy(theta_0(2))*Cz(theta_0(3));
Cp = C0;

SB = zeros(6,n);

for i = 1:n 1
    w= xhat1(1:3);
    th = xhat1(4:6);

    xhat2 = EulersRBE(w,th,I); %xhat1;
    P2 = MotionJacobian(w,th,I)*P1*MotionJacobian(w,th,I)' + Q;
    H = ObservationJacobian(th,B(i,:)');

    K = P2*H'*inv(H*P2*H' + R);

    meas = [Wn(i,:)';BFieldN(i,:)'];
    xhat3 = xhat2 + K * (meas   Observation(xhat2,BField(i,1:3)'));
    xhat3(4:6) = wrapToPi(xhat3(4:6));
    P3 = (eye(6)   K*H)*P2;

    SB(:,i) = sqrt(diag(P3));

    x(i,:) = xhat3;

    xhat1 = xhat3';
    P1 = P3;
```

```
end

%% Plotting
close all
clc

figure
plot(t(1:end 1),Wn(:,1),'r',t(1:end 1),Wn(:,2),'g',t(1:end 1),Wn(:,3),'k'); %t(1:end 1),Wn(:,2),'g',t(1:end 1),Wn(:,3),'b');
hold on
plot(t,x(:,1),t,x(:,2),t,x(:,3))%'c',t,x(:,2),'k',t,x(:,3),'y');
title('Comaprison of true and measured angular rates');
%legend('True','Measured');

figure
plot(t(1:end 1),Wn(:,1) x(1:end 1,1),'b');


figure
hold on
%plot(t,W(:,1),'k',t,W(:,2),'k',t,W(:,3),'k');
plot(t,x(:,4),'r'); %t,x(:,5),'b',t,x(:,6),'g');
plot(t(1:end 1),theta_true(:,1))% ,'c',t(1:end 1),theta_true(:,2),'k',t(1:end 1),theta_true(:,3),'y');
%legend('pitch','roll rate','yaw rate');
hold off

for i = 1:200000
    GravGrad(:,i) = D(:,2,i);
    Mag(:,i) = D(:,1,i);
end

tprime = t(1:200000);

figure
hold on
plot(tprime,GravGrad(:,:))
yyaxis right
plot(tprime,Mag(:,:))
hold off
```