ME529L: Attitude Estimation and Control: Fall 2022

Modelling Attitude Kinematics and Dynamics of a Satellite: Report

ME20B052 Sneha M S

Table of Contents:

- Problem Statement
- Introduction
- Attitude Estimation Methods
- Assumptions
- Results and Observations
- Conclusion
- Appendix

Problem Statement:

Model a satellite with a gyroscope, home tracker, and star trackers A and B as described below.

A satellite has the following principal-axes moments of inertia

$$J = \begin{bmatrix} 480 & 0 & 0 \\ 0 & 640 & 0 \\ 0 & 0 & 720 \end{bmatrix} \text{kg-m}^2,$$

and is equipped with the following sensors and actuators:

- thrusters that can generate moments $\vec{n} = [n_1 \quad n_2 \quad n_3]^T$ about the principal x, y and z directions,
- a gyroscope that can measure body-frame components of the angular velocity $\vec{\omega}$ along the x, y, and z directions with a Gaussian noise of rms 1 rad/s in each component,
- a star tracker that can measure the direction $\vec{h}$ of star A with a Gaussian noise of rms 0.1 in each component and additionally, a uniform sampling jitter of $\pm$0.01 s in the line of apparent motion,
- a star tracker that can measure the direction $\vec{k}$ of star B with a Gaussian noise of rms 0.1 in each component and additionally, a uniform sampling jitter of $\pm$0.01 s in the line of apparent motion,
- a home-tracker that can point in the direction $\vec{g}$ of the center of the Earth with Gaussian noise of rms 0.1 in each component and a uniform sampling jitter of $\pm$0.01 s in the line of apparent motion.

Assume the ground-frame fixed to the center of the Earth is inertial, and that the components of the measured vectors in the ground-frame are

$$h = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},\ k = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix},\ g = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

respectively.

Model (in Matlab) the attitude kinematics and dynamics of the satellite if the only external moments are on account of the thrusters, which can be controlled independently by user inputs to generate moments

$$\vec{n} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} = J\dot{\vec{\omega}} + \vec{\omega} \times J\vec{\omega}.$$

Also, model the output of the sensors with the indicated noise.

The noise model for the angular velocity would be

$$\vec{\omega}_{meas} = \vec{\omega} + \nu_\omega \vec{r}_\omega,$$

where $\nu_\omega = 1\,\mathrm{rad/s}$ is the rms noise in the measurement, and $\vec{r}_\omega$ is a 3-vector of random variables drawn out of a standard Gaussian distribution.

The noise model for the direction measurement u would be

$$u_{meas} = (1_{3\times 3} - \nu_T(2r_T - 1)[\omega\times])C^T g + \nu_u r_u,$$

where $C^T g$ is the ideal measurement in the absence of any errors, $\nu_T = 0.01\,\mathrm{rad/s}$ is the maximum amplitude of the sampling jitter, $\nu_u = 0.1$ is the rms noise in the measurement, r_T is a scalar random variable drawn out of a standard uniform distribution, and r_u is a 3-vector of random variables drawn out of a standard Gaussian distribution.

Having modeled the system, estimate the satellite attitude from the sensor measurements using

1. Triad algorithm with the home-tracker and star A tracker sensor
2. q-method (you can use the Quest algorithm if you wish), and
3. an extended attitude Kalman filter,

for the following initial condition

$$\check{q}(0) = \check{1},\ \vec{\omega} = \begin{bmatrix} 1 \\ 10 \\ 1 \end{bmatrix}$$

and the following three-moment inputs

1. $n_1 = n_2 = n_3 = 0$
2. $\vec{n} = -K_P\vec{p}$
3. $\vec{n} = -K_P\vec{p} - K_D\left(\hat{\omega} - \vec{\omega}_{cmd}\right)$

where $\hat{\omega}$ is the measured angular velocity, $\vec{\omega}_{cmd} = [0 \quad 5 \quad 0]^T \text{rad/s}$ is the desired angular velocity, $\vec{p}$ is the vector portion of the estimated attitude, $K_P = \text{diag}([1000 \quad 0 \quad 1000]) \text{ N-m/rad}$ is the proportional feedback gain, and $K_D = 2000 \text{ N-ms/rad}$ is the derivative feedback gain.

Introduction

The attitude of a spacecraft is a set of at least three independent quantities or any parametrization of the attitude matrix, which basically indicates the body's rotational position in relation to an inertial reference frame. Knowing this information is crucial for any successful space mission. The spacecraft attitude information will be constantly fed into the control system, which will evaluate and proceed with the actions required to achieve the desired attitude. This is the function of the ADCS – Attitude Determination and Control Subsystem.

Attitude determination methods can roughly be divided into static and dynamic categories. Static methods are time-independent, where all measurements (e.g., sun vector, magnetic field vector, etc.) are taken simultaneously or close enough in time that spacecraft motion between observations can be ignored or compensated for. It is a deterministic approach, and there is no need for information about past states.

Besides taking into account movement and, consequently, being time-dependent, dynamic attitude determination methods also consider measurement, not as a deterministic process but one where random noise is present, which requires a statistical approach. Filtering methods, such as Kalman filters, are used to organize information from past and actual measurements, knowledge of spacecraft motion, and possible errors in the system dynamics model.

In this project, the satellite attitude is estimated from the sensor measurements using

1. Triad algorithm with the home-tracker and star A tracker sensor
2. q-method (you can use the Quest algorithm if you wish), and
3. an extended attitude Kalman filter

The first two are attitude determination algorithms (static), and the last is an attitude estimation algorithm (dynamic).

Attitude Estimation Methods

1. Triaxial Attitude Determination (TRIAD) method:

The Triad method is one of the earliest and most straightforward solutions to the spacecraft attitude determination problem. Given the knowledge of two vectors in the

reference and body coordinates of a satellite, the Triad algorithm obtains the direction cosine matrix relating to both frames.

Triad algorithm problem statement: Estimate the attitude matrix C from the body-frame measurements of the components u, v of two vectors, whose components in the ground-frame, g, h, are already known (for example, acceleration due to gravity and geomagnetic field intensity).

Construct the orthogonal triads

$$X = \left[\frac{g}{\|g\|} \quad \frac{g\times h}{\|g\times h\|} \quad \frac{g\times(g\times h)}{\|g\times(g\times h)\|}\right]$$
$$= \left[\frac{g}{\|g\|} \quad \frac{g\times h}{\|g\times h\|} \quad \frac{gg^Th - hg^Tg}{\|g\|\|g\times h\|}\right],$$

for the ground-frame, and

$$Y = \left[\frac{u}{\|u\|} \quad \frac{u\times v}{\|u\times v\|} \quad \frac{u\times(u\times v)}{\|u\times(u\times v)\|}\right]$$
$$= \left[\frac{u}{\|u\|} \quad \frac{u\times v}{\|u\times v\|} \quad \frac{uu^Tv - vu^Tu}{\|u\|\|u\times v\|}\right],$$

for the body frame.

The estimate for the attitude matrix is

$$\hat{C} = XY^T.$$

The TRIAD method has the advantage of being very simple and intuitive but has the disadvantage of only allowing the use of two vectors. Besides, the information from the second vector is only partially used as part of the cross-product to determine, whereas the information of the first is totally used.

Also, If we want to improve the TRIAD method using several vectors, each DCM relating the body attitude to the inertial frame obtained from each pair of vectors would look slightly different due to sensor imprecisions. This problem is not faced when solved Wahba's problem.

---

2. Davenport's Q-method:

Wahba's problem, first posed by Grace Wahba in 1965, seeks to find a rotation matrix between two coordinate systems from a set of (weighted) vector observations. The cost function that Wahba's problem seeks to minimize is as follows:

$$J(\mathbf{R}) = \frac{1}{2}\sum_{k=1}^{N} a_k \|\mathbf{w}_k - \mathbf{R}\mathbf{v}_k\|^2 \text{ for } N \geq 2$$

where $\mathbf{w}_k$ is the kth 3-vector measurement in the reference frame, $\mathbf{v}_k$ is the corresponding kth 3-vector measurement in the body frame, and **R** is a 3 x 3 rotation matrix between the coordinate frames. a_k is an optional set of weights for each observation.

Several solutions to the problem have appeared in literature, notably Davenport's q-method.

Q-method problem statement: Estimate the attitude quaternion $\check{q}$ from the body-frame measurements of the components u, v, ..., of Euclidean vectors, whose components in the ground-frame, g, h, ..., are already known (for example, acceleration due to gravity, geomagnetic field intensity, sun tracker, etc.), to minimize the weighted quadratic error function

$E(\check{q}) = a\|g - \check{q} \otimes u \otimes \check{q}^{-1}\|^2 + b\|h - \check{q} \otimes v \otimes \check{q}^{-1}\|^2 + \dots.$

with measurement weights a, b, ...

Such that $a + b + \cdots = 1$,

The error energy function may be scaled by the irrelevant factor 2, negated, and shifted, and expressed as a quadratic form in $\check{q}$:

$$\begin{aligned} J(\check{q}) &= a(g \otimes \check{q})^T(\check{q} \otimes u) + b(h \otimes \check{q})^T(\check{q} \otimes v) + \dots \\ &= a\check{q}^T[g\otimes]^T[\otimes u]\check{q} + b\check{q}^T[h\otimes]^T[\otimes v]\check{q} + \dots \\ &= \check{q}^T D\check{q}, \end{aligned}$$

where

$$\begin{aligned} D &= a[g\otimes]^T[\otimes u] + b[h\otimes]^T[\otimes v] + \dots \\ &= a\begin{bmatrix} 0 & g^T \\ -g & -[g\times] \end{bmatrix}\begin{bmatrix} 0 & -u^T \\ u & -[u\times] \end{bmatrix} + b\begin{bmatrix} 0 & h^T \\ -h & -[h\times] \end{bmatrix}\begin{bmatrix} 0 & -v^T \\ v & -[v\times] \end{bmatrix} + \dots \end{aligned}$$

We use the method of Lagrange multipliers since we need to enforce the normalization constraint:

$J_a(\check{q}) = J + \lambda\big(1 - \check{q}^T\check{q}\big) = \check{q}^T D\check{q} + \lambda\big(1 - \check{q}^T\check{q}\big).$

Applying the first-order optimality constraints, we obtain

$0 = \partial_{\check{q}} J_a = 2\check{q}^T D - 2\lambda\check{q}^T$

$\Rightarrow D\check{q} = \lambda\check{q}.$

Therefore, the optimal attitude estimate $\check{q}$ is an eigenvector of Davenport's matrix D. We choose the eigenvector corresponding to the maximum eigenvalue $\lambda_{\max}$ to maximize $J = \check{q}^T D\check{q} = \lambda$.

The disadvantage of this method is the necessity of calculating the eigenvectors and eigenvalues of a four-by-four matrix, which may be demanding in terms of computational resources.

3. Extended Kalman Filter

Kalman filtering (KF) is a very popular tool for treating measurement data and making good estimations out of them. It is an iterative algorithm that estimates what is noise and what is usable information. KF could be used for getting better estimates out of noisy measurements.

Other important characteristics of KF are that it is a discrete process, data is sampled at intervals of time Δt, and it is a recursive process; it only needs information from the actual and previous states. The process is divided into a prediction phase, where a preliminary estimation is made, and an update phase, where observation results are incorporated.

The state equations must first be linearized using Taylor expansion to solve the attitude estimation problem.

Attitude KF (AKF) problem statement: Given the attitude system equations

$$\dot{\check{q}} = \frac{1}{2}\check{q} \otimes \omega, \quad \hat{\omega} = \omega + \xi,$$

$$u = \check{q}^{-1} \otimes g \otimes \check{q} + \theta_u, \; v = \check{q}^{-1} \otimes g \otimes \check{q} + \theta_v, \; \dots,$$

estimate the attitude quaternion $\check{q}$ using the body-frame angular velocity measurement $\hat{\omega}$, and measurements of the components u, v, ..., of Euclidean vectors, whose components in the ground-frame, g, h, ..., are already known (for example, acceleration due to gravity, geomagnetic field intensity, sun tracker, etc.),

given statistics $E[\xi] = 0$, $E[\theta] = 0$, $\mathrm{Cov}(\xi, \xi) = \Xi$, and $\mathrm{Cov}(\theta, \theta) = \Theta$ for the noise in the measurements, and the initial estimate $\check{p}_0$, to minimize a quadratic residual error.

Let the estimated and residual rotation at any time step be given by the quaternions $\check{p}$ and $\check{r}$. So,

$$\check{r} = \check{p}^{-1} \otimes \check{q} = \begin{bmatrix} r_0 \\ \vec{r} \end{bmatrix} \approx \begin{bmatrix} 1 \\ \vec{r} \end{bmatrix}.$$

The Kalman filter is implemented upon the small incremental rotation $\check{r}$ (rather than the estimate $\check{p}$ for the aggregate rotation). The attitude equations' linearization is accomplished by assuming that / is nearly equal to the identity quaternion.

The predict step takes the form

$$\check{p}_{k\,|\,k-1} = \check{p}_{k-1} \otimes \check{\varphi}_{k-1}, \quad \check{\varphi}_{k-1} = \begin{bmatrix} \cos\left(\|\hat{\omega}_{k-1}\| dt/2\right) \\ \sin\left(\|\hat{\omega}_{k-1}\| dt/2\right) \hat{\omega}_{k-1} / \|\hat{\omega}_{k-1}\| \end{bmatrix},$$

$$\check{r}_{k\,|\,k-1} = \check{1},$$

$$R_{k\,|\,k-1} = \mathrm{Cov}\left(\vec{r}_{k\,|\,k-1}, \vec{r}_{k\,|\,k-1}\right) = A_{k-1} R_{k-1} A_{k-1}^T + \Xi_{k-1} (dt)^2 / 4,$$

where

$$A_{k-1} = 1_{3\times 3} + [\times\hat{\omega}_{k-1}]dt,\ R_{k-1} = \mathrm{Cov}(\vec{r}_{k-1}, \vec{r}_{k-1}),\ \Xi_{k-1} = \mathrm{Cov}(\xi_{k-1}, \xi_{k-1}).$$

The update/correct step takes the form

$$L_k = R_{k\,|\,k-1} C_k^T \left(C_k R_{k\,|\,k-1} C_k^T + \Theta_k\right)^{-1},$$

$$\vec{r}_k = L_k \begin{bmatrix} u_k - \hat{u}_{k\,|\,k-1} \\ v_k - \hat{v}_{k\,|\,k-1} \\ \vdots \end{bmatrix},$$

$$\check{p}_k = \check{p}_{k\,|\,k-1} \otimes \check{r}_k = \check{p}_{k\,|\,k-1} \otimes \begin{bmatrix} \sqrt{1 - \|\vec{r}_k\|^2} \\ \vec{r}_k \end{bmatrix},$$

where

$$C_k = \frac{\partial}{\partial \vec{r}_k} \begin{bmatrix} \hat{u}_k \\ \vdots \end{bmatrix} = 2 \begin{bmatrix} \left[\hat{u}_{k\,|\,k-1}\times\right] \\ \vdots \end{bmatrix}, \quad \Theta_k = \begin{bmatrix} \Theta_{u,k} & 0 \\ 0 & \ddots \end{bmatrix},$$

$$\hat{u}_{k\,|\,k-1} = \check{p}_{k\,|\,k-1}^{-1} \otimes g \otimes \check{p}_{k\,|\,k-1},\ \ldots\ .$$

The covariance matrix at the end of the update/correct step is given as usual by

$$R_k = R_{k\,|\,k-1} - L_k C_k R_{k\,|\,k-1}.$$

The extended Kalman filter solves the non-linear estimation problem by linearising state and/or measurement equations and applying the standard Kalman filter formulas to the resulting linear estimation problem. By treating non-linear processes with a linearized approximation, errors are being created that may have an accumulative effect inducing numerical instabilities

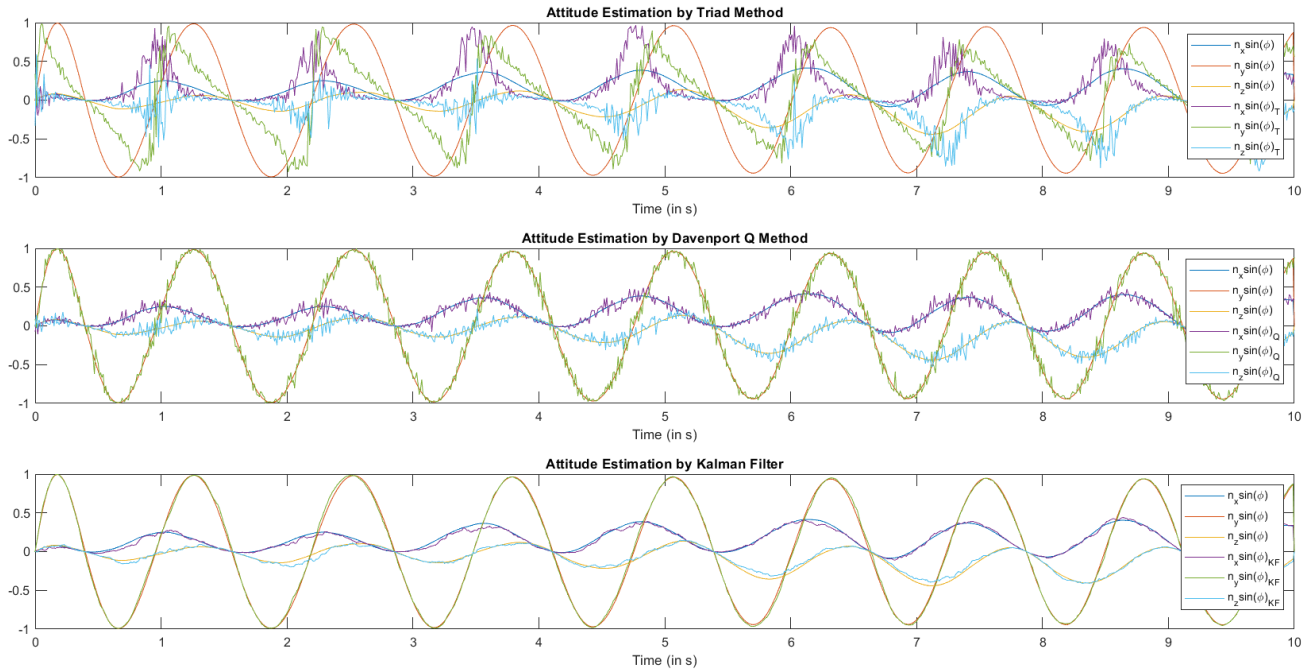
---

Assumptions

- The satellite is modeled as a single rigid body. We are also not taking into account slosh of liquid fuels or cryogens,
- The gyroscope measurements come associated with a Gaussian noise of rms 1 rad/s in each component and the home and star trackers measure the direction with a Gaussian noise of rms 0.1 in each component and additionally, a uniform sampling jitter of $\pm$0.01s in the line of apparent motion.
- One important assumption used in KF is that the process and measurement uncertainties are white noises, which means they are Gaussian distributions with known covariances and with mean values equal to zero

- Another critical assumption is that the process, transition, or system dynamic model is linear and has the form as
  $x_{k+1} = A_k x_k + u_k + \xi_k$, where $x \in \mathbb{R}^n$.
- The measurement or observation model is also assumed to be linear and can be described as
  $z_k = C_k x_k + \theta_k$, where $z \in \mathbb{R}^m$.
- We will assume that the disturbance and noise at any given time instant are generated from white, zero-mean Gaussian processes. While the disturbance and noise components at a given instant are internally correlated through Ξ and Θ, we will assume they are uncorrelated to any other quantity. That is,
  $\mathrm{Cov}(\xi_k, \xi_k) = \Xi_k$ and $\mathrm{Cov}(\theta_k, \theta_k) = \Theta_k$, but $\mathrm{Cov}(\xi_k, \xi_i) = 0$, $\mathrm{Cov}(\theta_k, \theta_i) = 0$, $\mathrm{Cov}(\theta_i, \xi_j) = 0$, $\mathrm{Cov}(\theta_i, x_j) = 0$, $\mathrm{Cov}(\xi_i, x_j) = 0$

Observations and Results



It is visible that the accuracy has improved in Davenport's method when compared with the results obtained by implementing the TRIAD algorithm. Moreover, the Kalman Filter gives the most accurate results.

It is observed that the TRIAD algorithm takes 0.031635 seconds, Davenport's Q- method takes 0.079652 seconds, and Extended Kalman Filter takes 0.144790 seconds to estimate the attitude of the spacecraft.

It is also imperative to note that the TRIAD and Davenport methods can only estimate the attitude. However, the Extended Attitude Kalman Filter can be used to estimate both attitude and angular velocities.

Conclusion

Attitude determination algorithms are deterministic methods. They are characterized by low computational efforts, less accuracy and amount of memory in comparison with attitude estimation algorithms. This is due to its independence of dynamic equation. However, this latter is used by attitude estimation algorithms. The use of this equation increases the accuracy of these algorithms. However, they are highly time-consuming (the execution time of the EKF is approximately 4.6 times the execution time of TRIAD Method) because it needs much calculation (calculation of the covariance matrix), and its structure uses dynamic models. In addition, the memory requirement of such algorithms is the largest.

Appendix – Matlab Code

```
%Project Part 2 - ME20B052 Sneha M S
% g,h,k --> direction of centre of Earth, Star A and Star B from the body frame of the satellite respectively
% <variable>_wn --> data without sensor noise
clear *; close all;

%% Modelling the System
J=[480 0 0; 0 640 0; 0 0 720]; %Moment of Inertia Tensor
n=zeros(3,1001); %Moments provided by Thrusters;
g=zeros(3,1001);
k=zeros(3,1001);
h=zeros(3,1001);
Q=zeros(4,1001); %Rotation Quaternion
w=zeros(3,1001); %Angular velocity
a=zeros(3,1001); %Angular Acceleration
g_wn=zeros(3,1001);
k_wn=zeros(3,1001);
h_wn=zeros(3,1001);
w_wn=zeros(3,1001);

%disp("Initialize the altitude quaternion");
%Q(1,1)=input("Enter the value of q0: ");
%Q(2,1)=input("Enter the value of q1: ");
%Q(3,1)=input("Enter the value of q2: ");
%Q(4,1)=input("Enter the value of q3: ");
Q(:,1)=[1;0;0;0];
% Q(1,1)=Q(1,1);
% Q(2,1)=Q(2,1);
% Q(3,1)=Q(3,1);

%disp("Initialize the angular velocity");
%w(1,1)=input("Enter the value of wx: ");
```

```
%w(2,1)=input("Enter the value of wy: ");
%w(3,1)=input("Enter the value of wz: ");
w_wn(1,1)=1;
w_wn(2,1)=10;
w_wn(3,1)=1;

%disp("Enter the direction of Star A measured in ground frame");
%h1=input("Enter the value of hx: ");
%h2=input("Enter the value of hy: ");
%h3=input("Enter the value of hz: ");
h0=[0;1;0]; %With respect to ground frame
% ru=rand;
% rn=normrnd(0,1,3,1);

h_wn(:,1)=quat_to_matrix(Q(:,1))'*h0;
h(:,1)=h_wn(:, 1) + 0.1*randn(3, 1);

%disp("Enter the direction of Star B measured in ground frame");
%k1=input("Enter the value of kx: ");
%k2=input("Enter the value of ky: ");
%k3=input("Enter the value of kz: ");
k0=[0;-1;0]; %With respect to ground frame
% ru=rand;
% rn=normrnd(0,1,3,1);
k_wn(:,1)=quat_to_matrix(Q(:,1))'*k0;
k(:,1)=k_wn(:, 1) + 0.1*randn(3, 1);

g0=[0;0;1]; %With respect to ground frame
% ru=rand;
% rn=normrnd(0,1,3,1);
g_wn(:,1)=quat_to_matrix(Q(:,1))'*g0;
g(:,1)=g_wn(:, 1) + 0.1*randn(3, 1);

Kp=[1000 0 0; 0 0 0; 0 0 1000];
w_cd=[0 5 0]';
Kd=2000;

dt=0.01;
total_time=10;
time=0;
for i =1:(total_time/dt)
    time=time+dt;
    rn=normrnd(0,1,3,1);
    w(:,i)=w_wn(:,i)+rn;
    %n(:,i+1)=-Kp*Q(2:4,i);
    n(:,i+1)=-Kp*Q(2:4,i)-Kd.*(w(:,i)-w_cd);
    a(:,i)=J\(n(:,i)-cross(w_wn(:,i),(J*w_wn(:,i))));
    w_wn(:,i+1)=w_wn(:,i)+a(:,i).*dt;
    temp=[0,w_wn(:,i+1)'];
    q_dot=0.5.*quatprod(Q(:,i), temp);
    Q(:,i+1)=Q(:,i)+(q_dot).*dt;
    Q(:,i+1)=Norm(Q(:,i+1));
    C=quat_to_matrix((Q(:,i+1))');
    g_wn(:,i+1)=uni(C'*g0);
    h_wn(:,i+1)=uni(C'*h0);
    k_wn(:,i+1)=uni(C'*k0);
    rn=normrnd(0,1,3,1);
```

```
    g(:,i+1)=C'*g0+(0.1.*rn);
    rn=normrnd(0,1,3,1);
    h(:,i+1)=C'*h0+(0.1.*rn);
    rn=normrnd(0,1,3,1);
    k(:,i+1)=C'*k0+(0.1.*rn);
end

% % Plotting Graphs
axis=0:dt:total_time;
% axis(:,201)=[ ];
% t=tiledlayout(2,2);
% title(t,"Sensor Outputs");
%
% nexttile
% plot(axis,g,axis,g_wn);
% title('Home Tracker')
% xlabel("Time (in s)")
% legend('ĝx','ĝy','ĝz','gx','gy','gz')
%
% nexttile
% plot(axis,h,axis,h_wn);
% title('Star A Tracker')
% xlabel("Time (in s)")
% legend('ĥx','ĥy','ĥz','hx','hy','hz')
%
% nexttile
% plot(axis,k,axis,k_wn);
% title('Star B Tracker')
% xlabel("Time (in s)")
% legend('ǩx','ǩy','ǩz','kx','ky','kz')
%
% nexttile
% plot(axis,w);
% hold on
% plot(axis,w_wn);
% title('Gyroscope Readings')
% xlabel("Time (in s)")
% ylabel("ω(t) (in rad/s)")
% legend('ŵx','ŵy','ŵz','ωx','ωy','ωz')

%% Attitude Estimation from Sensor Data - Triad Algorithm
%Home Tracker Sensor Output - g
%Star A Tracker Sensor output - h
%Measurements made in ground frame - g0 and h0
tic
Q_triad=zeros(4,1001);
C_triad=zeros(3,3,1001);
D_triad=zeros(3,1001);
G=[uni(g0),uni(cross(g0,h0)),uni(cross(g0,cross(g0,h0)))];
for i=1:(total_time/dt)+1
    u=g(:,i);
    v=h(:,i);
    B=[uni(u),uni(cross(u,v)),uni(cross(u,cross(u,v)))];
    C_triad(:,:,i)=G*transpose(B);
    Q_triad(:,i)=Norm(Matrix_to_Quat(C_triad(:,:,i)));
    D_triad(:,i)=(2*Q_triad(1,i)).*(Q_triad(2:4,i))';
end
```

```
toc

%% Attitude Estimation from Sensor Data - Davenport's Q Method
% Vectors - h0,h (Star A); g0,g (Earth); k0,k (Star B)
tic
Q_qmethod=zeros(4,1001);
D_qmethod=zeros(3,1001);
w1=1/3; w2=1/3; w3=1/3;
v1_N=g0;
v2_N=h0;
v3_N=k0;
for i=1:(total_time/dt)+1
    v1_B=g(:,i);
    v2_B=h(:,i);
    v3_B=k(:,i);
    B=w1.*(v1_B*v1_N')+w2.*(v2_B*v2_N')+w3.*(v3_B*v3_N');
    sigma=trace(B);
    S=B+B';
    Z=[B(2,3) - B(3,2) ;
    B(3,1) - B(1,3) ;
    B(1,2) - B(2,1) ];
    K=[sigma, Z'; Z, (S-sigma*eye(3))];
    [eigvec,eigval]=eig(K);
    max=eigval(1,1);
    ind=1;
    for j=2:4
        if(eigval(j,j)>max)
            max=eigval(j,j);
            ind=j;
        end
    end
    Q_qmethod(:,i)=eigvec(:,ind)/norm(eigvec(:,ind));
    D_qmethod(:,i)=(2*Q_qmethod(1,i)).*(Q_qmethod(2:4,i))';
end
toc

%% Attitude Estimation - Kalman Filter
tic
%Covariance Matrix
E=eye(3);
O=0.1.*eye(9);
Q_kalman=zeros(4,1001);
Q_kalman(:,1)=[1;0;0;0];
D_kalman=zeros(3,1001);
r=zeros(4,1001);
r(:,1)=[1;0;0;0];
R=zeros(3,3,1001);
for i=1:(total_time/dt)
    phi=mag(w(:,i))*dt/2;
    temp=quatprod(Q_kalman(:,i),[cos(phi); uni(w(:,i)).*sin(phi)]);
    r_temp=[1;0;0;0];
    A=eye(3) - (crossprodv(w(:,i))).*dt;
    R_temp=A*R(:,:,i)*A' + E*dt^2./4; %Predict Step
    s=qinverse(temp);
    z_temp=[quatprod(quatprod(s,[0;g0]),temp); quatprod(quatprod(s,[0;h0]),temp); quatprod(quatprod(s,[0;k0]),temp)];
    u_temp=[z_temp(2:4);z_temp(6:8);z_temp(10:12)];
    C=2.*[crossprodv(u_temp(1:3));crossprodv(u_temp(4:6));crossprodv(u_temp(7:9))];
```

```
    L=R_temp*C'*inv(C*R_temp*C'+O);  %Update Step
    R(:,:,i+1)=R_temp-L*C*R_temp;
    r(2:4,i+1)=L*[g(:,i+1)-u_temp(1:3);h(:,i+1)-u_temp(4:6); k(:,i+1)-u_temp(7:9)];
    r(1,i+1)=sqrt(1-(mag(r(2:4,i+1)))^2);
    Q_kalman(:,i+1)=quatprod(temp, r(:,i+1));
    D_kalman(:,i)=(2*Q_kalman(1,i)).*(Q_kalman(2:4,i))';
end
toc

%% Comparing Estimated Attitude and True Attitude
t2=tiledlayout(3,1);
D=zeros(3,1001);
for i=1:(total_time/dt)
    D(:,i)=(2*Q(1,i)).*(Q(2:4,i))';
end

nexttile
plot(axis,D,axis,D_triad);
title('Attitude Estimation by Triad Method')
xlabel("Time (in s)")
legend('n_xsin(\phi)','n_ysin(\phi)','n_zsin(\phi)','n_xsin(\phi)_T','n_ysin(\phi)_T','n_zsin(\phi)_T');

nexttile
plot(axis,D,axis,D_qmethod);
title('Attitude Estimation by Davenport Q Method')
xlabel("Time (in s)")
legend('n_xsin(\phi)','n_ysin(\phi)','n_zsin(\phi)','n_xsin(\phi)_Q','n_ysin(\phi)_Q','n_zsin(\phi)_Q');

nexttile
plot(axis,D,axis,D_kalman);
title('Attitude Estimation by Kalman Filter')
xlabel("Time (in s)")
legend('n_xsin(\phi)','n_ysin(\phi)','n_zsin(\phi)','n_xsin(\phi)_{KF}','n_ysin(\phi)_{KF}','n_zsin(\phi)_{KF}');

%% Functions
%To find the rotation Matrix corresponding to the Quaternion
function C = quat_to_matrix(q)
    C(1,1)=(q(1)^2)+(q(2)^2)-(q(3)^2)-(q(4)^2);
    C(2,2)=(q(1)^2)-(q(2)^2)+(q(3)^2)-(q(4)^2);
    C(3,3)=(q(1)^2)-(q(2)^2)-(q(3)^2)+(q(4)^2);
    C(1,2)=2*((q(2)*q(3))-(q(4)*q(1)));
    C(2,1)=2*((q(2)*q(3))+(q(4)*q(1)));
    C(1,3)=2*((q(2)*q(4))+(q(3)*q(1)));
    C(3,1)=2*((q(2)*q(4))-(q(3)*q(1)));
    C(2,3)=2*((q(3)*q(4))-(q(2)*q(1)));
    C(3,2)=2*((q(3)*q(4))+(q(2)*q(1)));
end

%To find the product of two quaternions
function prod = quatprod(p,q)
    p0=p(1)*q(1)-p(2)*q(2)-p(3)*q(3)-p(4)*q(4);
    p1=p(1)*q(2)+p(2)*q(1)+p(3)*q(4)-p(4)*q(3);
    p2=p(1)*q(3)-p(2)*q(4)+p(3)*q(1)+p(4)*q(2);
    p3=p(1)*q(4)+p(2)*q(3)-p(3)*q(2)+p(4)*q(1);
    prod=[p0;p1;p2;p3];
end
```

```
%To Find the Cross Product Vector
function V = crossprodv(u)
    V(1,1)=0;
    V(1,2)=-u(3);
    V(1,3)=u(2);
    V(2,1)=u(3);
    V(2,2)=0;
    V(2,3)=-u(1);
    V(3,1)=-u(2);
    V(3,2)=u(1);
    V(3,3)=0;
end

%To normalize a quaternion
function n=Norm(q)
    sum=sqrt(q(1)^2+q(2)^2+q(3)^2+q(4)^2);
    n=q./sum;
end

%Obtaining a Unit Vector
function a = uni(b)
norm=0;
for i=1:3
    norm=norm+(b(i)^2);
end
norm=sqrt(norm);
a=b./norm;
end

%Obtaining Magnitude of a Vector
function a = mag(b)
norm=0;
for i=1:3
    norm=norm+(b(i)^2);
end
norm=sqrt(norm);
a=norm;
end

function q=Matrix_to_Quat(C)
    q(1)=0.25*sqrt(1+C(1,1)+C(2,2)+C(3,3));
    q(2)=(C(3,2) - C(2,3))/4/q(1);
    q(3)=(C(1,3) - C(3,1))/4/q(1);
    q(4)=(C(2,1) - C(1,2))/4/q(1);
end

function qinv = qinverse(q)
    qinv(1,1)=q(1,1);
    qinv(2,1)=-q(2,1);
    qinv(3,1)=-q(3,1);
    qinv(4,1)=-q(4,1);
end
```