Modelling and Estimation of Spacecraft Attitude Kinematics and Kinetics

Attitude Estimation and Control

08-12-2023

Sumukh Porwal
Roll No.: **ME20B041**
B. Tech. Mechanical Engineering
Indian Institute of Technology Tirupati

Guide:
Dr. Yujendra Mitikiri
Department of Mechanical Engineering
Indian Institute of Technology Tirupati

Contents

List of Figures

1 Introduction

1.1 System description

A satellite has the following principal-axes moments of inertia about its centre-of-mass

$$J = \begin{bmatrix} 480 & 0 & 0 \\ 0 & 640 & 0 \\ 0 & 0 & 960 \end{bmatrix} kg - m^2,$$

and is equipped with the following sensors and actuators:

- thrusters that can generate moments $n = \begin{bmatrix} n_1 & n_2 & n_3 \end{bmatrix}^T$ up to a maximum of 80, 80, 80 $kN - m$ about the satellite's principal x, y, and z directions,
- a gyroscope that can measure body-frame components of the angular velocity $\vec{\omega}$ up to 20 rad/s 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 if A is within 60 degrees of the body-frame $+y$ axis, 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 if B is within 60 degrees of the body-frame $-y$ axis, 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 if the Earth is within 60 degrees of the body-frame $+z$ axis, with a Gaussian noise of rms 0.1 in each component and additionally, a uniform sampling jitter of ± 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.

Initial Conditions:

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

1.2 System Dynamics

Quaternions: the (unit-magnitude) axis, $\vec{n}$, and angle, Φ, of rotation, are combined as a column vector $\check{q}$, with a scalar part q_0, and vector part $\vec{q}$:

$$\check{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ \vec{q} \end{bmatrix} = \begin{bmatrix} cos\Phi/2 \\ \vec{n} sin\Phi/2 \end{bmatrix},$$

$$\begin{aligned} & \vec{n}^T \vec{n} = 1 \\ \implies & \check{q}^T \check{q} = cos^2\Phi/2 + \vec{n}^T \vec{n} \ sin^2\Phi/2 = 1 \\ & \check{p} \otimes \check{q} = \begin{bmatrix} p_0 \\ \vec{p} \end{bmatrix} \otimes \begin{bmatrix} q_0 \\ \vec{q} \end{bmatrix} = \begin{bmatrix} p_0 q_0 - \vec{p}^T \vec{q} \\ p_0 \vec{q} + q_0 \vec{p} + \vec{p} \times \vec{q} \end{bmatrix} \\ & \check{q}^{-1} = \begin{bmatrix} q_0 \\ -\vec{q} \end{bmatrix} \end{aligned}$$

The time derivative of a rotation quaternion is given in terms of its body-frame angular velocity and ground-frame angular velocity $\vec{\chi}$ by

$$\dot{\check{q}} = \frac{1}{2}\check{q} \otimes \vec{\omega} = \frac{1}{2}\vec{\chi} \otimes \check{q}$$

The ground frame angular velocity $\vec{\chi}$ is given by the following relation:

$$\vec{\chi} = \check{q} \otimes \vec{\omega} \otimes \check{q}^{-1}$$

In the next step the quaternion is updated as

$$\check{q}_{t+1} = \check{q}_t + \dot{\check{q}} \times ts$$

where, ts = differential time step

Quaternions must abide the unit length constraint, so the updated quaternion has to be divided by its magnitude.

$$\check{q} = \frac{\check{q}}{||\check{q}||}$$

For the body frame angular velocity $\vec{\omega}$, the time derivative is given by:

$$\dot{\vec{\omega}} = J^{-1}(\vec{n} - (\vec{\omega} \times J\vec{\omega}))$$

where $\vec{n}$ = Applied Moment

From the obtained time derivative of angular velocity, the angular velocity is obtained

$$\vec{\omega} = \vec{\omega} + \dot{\vec{\omega}} \times ts$$

2 Methods for Attitude Estimation

There are two major classes of attitude estimation depending upon whether we use kinematic measurements or purely geometric measurements. We have geometric attitude estimation if all the measurements correspond to geometric quantities, and geometro-kinematic attitude estimation if we may additionally measure the angular velocity. We will use two geometric attitude estimators, namely Harold Black's Vector Triad method and Paul Davenport's q-method, and one geometro-kinematic attitude estimator, namely the Extended Kalman Filter applied to the attitude space.

2.1 Triad Method

Triad algorithm problem statement: Estimate the attitude matrix $\hat{C}$ from the body-frame measurements of the components , u and v of two vectors, whose components in the ground-frame, g and h, are already known (example, acceleration due to gravity and geo-magnetic field intensity). Construct the orthogonal triads,

$$X = \begin{bmatrix} \frac{g}{|g|} & \frac{g \times h}{|g \times h|} & \frac{g \times (g \times h)}{|g \times (g \times h)|} \end{bmatrix} = \begin{bmatrix} \frac{g}{|g|} & \frac{g \times h}{|g \times h|} & \frac{gg^T h - hg^T g)}{|g||g \times h|} \end{bmatrix}$$

for the ground-frame, and

$$Y = \begin{bmatrix} \frac{u}{|u|} & \frac{u \times v}{|u \times v|} & \frac{u \times (u \times v)}{|u \times (u \times v)|} \end{bmatrix}, = \begin{bmatrix} \frac{u}{|u|} & \frac{u \times v}{|u \times v|} & \frac{uu^T v - vu^T u)}{|u||u \times u|} \end{bmatrix}$$

for the body frame.

The estimate for the attitude matrix is,

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

It can be easily verified that,

$$X = \hat{C}Y,$$

$$\frac{g}{|g|} = \hat{C}\frac{u}{|u|},$$

$$\frac{h}{|h|} = \hat{C}\frac{v}{|v|},\ if\ u^T v = g^T h$$

2.2 Quest 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 groundframe, g, h, ..., are already known (example, acceleration due to gravity, geo-magnetic field intensity, sun tracker, etc), so as to minimize the weighted quadratic error function

$$E(\check{q}) = a|u - \hat{u}|^2 + b|v - \hat{v}|^2 + ...,$$

with measurement weights a, b, ...
where,

$$\hat{u} = \hat{q}^{-1} \otimes \check{g} \otimes \hat{q},\ \hat{v} = \hat{q}^{-1} \otimes \check{h} \otimes \hat{q},\ ...$$

Define the weighted error energy function $E(\check{q})$ as a function of the quaternion estimate $\check{q}$, with weight parameters a, b, ..., summing to 1

$$a + b + ... = 1$$

and the given measurements u, v, ..., g, h, ..., as:

$$\begin{aligned}
E(\check{q}) &= a|u - \hat{q}^{-1} \otimes \check{g} \otimes \hat{q}|^2 + b|v - \hat{q}^{-1} \otimes \check{h} \otimes \hat{q}|^2 + \dots \\
&= a|u|^2 + a|\hat{q}^{-1} \otimes \check{g} \otimes \hat{q}|^2 - 2au^T(\hat{q}^{-1} \otimes \check{g} \otimes \hat{q}) \\
&\quad + b|v|^2 + b|\hat{q}^{-1} \otimes \check{h} \otimes \hat{q}|^2 - 2bv^T(\hat{q}^{-1} \otimes \check{h} \otimes \hat{q}) + \dots \\
&= a|u|^2 + a|g|^2 - 2a(\hat{q} \otimes \check{u})^T(\check{g} \otimes \hat{q}) \\
&\quad b|v|^2 + b|h|^2 - 2b(\hat{q} \otimes \check{v})^T(\check{h} \otimes \hat{q}) + \dots \\
&= 2a - 2a(\hat{q} \otimes \check{u})^T(\check{g} \otimes \hat{q}) + 2b - 2b(\hat{q} \otimes \check{v})^T(\check{h} \otimes \hat{q}) + \dots \\
&= 2(a + b + \dots) - 2(a(\hat{q} \otimes \check{u})^T(\check{g} \otimes \hat{q}) + b(\hat{q} \otimes \check{v})^T(\check{h} \otimes \hat{q}) + \dots)
\end{aligned}$$

The error energy functional E may be scaled by the irrelevant factor of two, negated, and shifted to the origin, and expressed as a quadratic form in $\hat{q}$:

$$\begin{aligned}
J(\hat{q}) = a + b + \dots - E/2 &= a(\hat{q} \otimes \check{u})^T(\check{g} \otimes \hat{q}) + b(\hat{q} \otimes \check{v})^T(\check{h} \otimes \hat{q}) + \dots \\
&= a\hat{q}^T[\otimes\check{u}]^T[\check{g}\otimes]\hat{q} + b\hat{q}^T[\otimes\check{v}]^T[\check{h}\otimes]\hat{q} + \dots \\
&= \hat{q}^T D\hat{q} \qquad (2.1)
\end{aligned}$$

where D is **Davenport's matrix**. On account of the negation, minimizing E is equivalent to maximizing J. But, of course, we use calculus only to extremize $J(\hat{q})$, and then choose the solution which maximizes (rather than minimizes) J. Expressing the cost function J in the form of (2.1) is useful because it isolates the unknown attitude estimate $\hat{q}$ from the weights and measurements which are collected together in the matrix D.

Davenport's matrix D may be simplified to

$$\begin{aligned}
D &= a[\otimes\check{u}]^T[\check{g}\otimes] + b[\otimes\check{v}]^T[\check{h}\otimes] + \dots \\
&= a\begin{bmatrix} 0 & u^T \\ -u & [u\times] \end{bmatrix}\begin{bmatrix} 0 & -g^T \\ g & [g\times] \end{bmatrix} + b\begin{bmatrix} 0 & v^T \\ -v & [v\times] \end{bmatrix}\begin{bmatrix} 0 & -h^T \\ h & [h\times] \end{bmatrix} + \dots \\
&= a\begin{bmatrix} u^Tg & u^T[g\times] \\ u \times g & ug^T + [u\times][g\times] \end{bmatrix} + b\begin{bmatrix} v^Th & v^T[h\times] \\ v \times g & vg^T + [v\times][h\times] \end{bmatrix} + \dots \\
&= a\begin{bmatrix} u^Tg & (u \times g)^T \\ u \times g & ug^T + gu^T + u^Tg1_{3\times3} \end{bmatrix} + b\begin{bmatrix} v^Tg & (v \times h)^T \\ v \times h & vh^T + hv^T + v^Th1_{3\times3} \end{bmatrix} + \dots
\end{aligned}$$

In order to determine the optimal attitude estimate, we use the method of Lagrange multipliers (described in the previous subsection) to maximize the quadratic form in (2.1) subject to the normalization constraint for an attitude quaternion

$$\hat{q}^T\hat{q} = 1$$

The auxilliary cost function

$$J_a = \hat{q}^T D\hat{q} + \lambda(1 - \hat{q}^T\hat{q})$$

and apply the first-order optimality conditions to obtain

$$J_a = 2\hat{q}^{*T} D - 2\lambda \hat{q}^{*T}$$
$$\implies D\hat{q}^* = \lambda \hat{q}^*$$

The above is an eigenvalue-eigenvector equation for Davenport's matrix D. Since D is symmetric, all eigenvalues are real. Further, since D is traceless, we will necessarily have at least one positive real root. The value of the objective function J_a at the optimal value is

$$J_a = \hat{q}^{*T} D \hat{q}^* = \hat{q}^{*T} \lambda \hat{q}^* = \lambda$$

Thus maximizing J_a is equivalent to maximizing the eigenvalue λ, and **the optimal attitude estimate is the eigenvector of Davenport's matrix corresponding to the maximum eigenvalue**.

2.3 Attitude Kalman Filter

Attitude Kalman Filter (AKF) problem statement: Given the attitude system equations,

$$\dot{\check{q}} = \frac{1}{2}\check{q} \otimes \omega, \ \hat{\omega} = \omega + \xi$$
$$u = \hat{q}^{-1} \otimes g \otimes \hat{q} + \theta_u, \ v = \hat{q}^{-1} \otimes h \otimes \hat{q} + \theta_v, \ \ldots$$

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 groundframe, g, h, ..., are already known (example, acceleration due to gravity, geo-magnetic field intensity, sun tracker, etc), given statistics $E[\xi] = 0$, $E[\theta] = 0$, $Cov(\xi, \xi) = \Xi$, and $Cov(\theta, \theta) = \Theta$ for the noise in the measurements, and the initial estimate $\check{p}_0$, so as to minimize a quadratic residual error.

Kalman filtering (KF) is a very popular tool for treating measurement data and making good estimations out of them. KF could be defined as an iterative algorithm, which assumes that the measurement consists of a true value corrupted by a noise or error and in a mathematical way it makes an educated guess of which part of the observation that is noise and which that is the closest we can get to the true value. For doing so, the algorithm needs more information that comes partly from the knowledge on the process, in the form of the equations of motions for example, and partly from other observations/measurements.

Important characteristics of KF is that it is a discrete process, data is sampled at intervals of time δt, and it is 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 Kalman filter may be appropriately expanded to estimate attitude based on measurements of attitude kinematics and direction.

A time step of 0.01 seconds has been taken.

Let the estimated and residual rotation at any time-step be given by the quaternion $\check{p}$ 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 now implemented upon the small incremental rotation $\check{r}$. The linearization in the attitude equations is accomplished by assuming that $\check{r}$ is very nearly equal to the identity quaternion.

The **predict step** takes the form

$$\begin{aligned}
\check{p}_{k|k-1} &= \check{p}_{k-1} \otimes \check{\varphi}_{k-1}, \\
\check{\varphi}_{k-1} &= \begin{bmatrix} cos(|\hat{\omega}_{k-1}|dt/2) \\ sin(|\hat{\omega}_{k-1}|dt/2)\frac{\hat{\omega}_{k-1}}{|\hat{\omega}_{k-1}|} \end{bmatrix}, \\
\check{r}_{k|k-1} &= \check{1}, \\
R_{k|k-1} &= Cov(\vec{r}_{k|k-1}, \vec{r}_{k|k-1}) = A_{k-1} R_{k-1} A_{k-1}^T + \Xi_{k-1}(dt^2)/4,
\end{aligned}$$

where,

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

The **update/correct step** takes the form

$$\check{p}_k = \check{p}_{k|k-1} \otimes E[\check{r}_k] = \check{p}_{k|k-1} \otimes \begin{bmatrix} \sqrt{1 - |E[r_{v,k}]^2|} \\ E[r_{v,k}] \end{bmatrix},$$

where,

$$\begin{aligned}
E[r_{v,k}] &= (R_{k|k-1}^{-1} + C_k^T \Theta_k^{-1} C_k) C_k^T \Theta_k^{-1} \begin{bmatrix} u_k - \hat{u}_{k|k-1} \\ v_k - \hat{v}_{k|k-1} \\ \vdots \end{bmatrix} \\
C_k &= \frac{\partial}{\partial r_{v,k}} \begin{bmatrix} \hat{u}_k \\ \hat{v}_k \\ \vdots \end{bmatrix} = 2 \begin{bmatrix} [\hat{u}_{k|k-1} \times] \\ [\hat{v}_{k|k-1} \times] \\ \vdots \end{bmatrix}, \\
\Theta_k &= \begin{bmatrix} \Theta_{u,k} & 0 & \cdots \\ 0 & \Theta_{v,k} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}, \\
\hat{u}_{k|k-1} &= \check{p}_{k|k-1}^{-1} \otimes g \otimes \check{p}_{k|k-1}, \ \ldots
\end{aligned}$$

The update/correct step is conceptually performed after the direction measurements are made available. The estimates for the residual rotation $\check{r}_k$ and the attitude $\hat{p}_k$ after this step are also called **a-posteriori** estimates, since they are computed after incorporating the information contained in direction measurements. The covariance matrix for $\hat{p}_k$ at the end of the update step is given as usual by

$$R_k = (R_{k|k-1}^{-1} + C_k^T \Theta_k^{-1} C_k)^{-1} = R_{k|k-1} - R_{k|k-1} C_k^T (C_k R_{k|k-1} C_k^T + \Theta_k)^{-1} C_k R_{k|k-1}$$

The covariance in the a-posteriori estimate $\hat{p}_k$ is reduced with respect to the covariance of the **a-priori** estimate $\hat{p}_{k|k-1}$ since we have utilized the information contained in the direction measurements to filter out the errors in the measurements.

3 Results and Discussion

3.1 Triad Method

The Triad algorithm, utilizing both home-tracker and star A tracker sensors, stands out for its simplicity and real-time applicability. Its reliance on multiple sensors ensures redundancy, fostering a dependable attitude estimation method even in scenarios with minimal external disturbances.

Limitations:

However, the algorithm may face challenges in handling sensor noise and calibration errors, potentially impacting accuracy. In situations with limited star visibility, maintaining precision becomes more challenging due to reduced celestial reference points.

Comparative Advantages and Disadvantages:

Compared to other methods, the Triad algorithm's computational efficiency and suitability for real-time processing are evident. The redundancy provided by multiple sensors enhances the system's fault tolerance. Despite its advantages, the Triad algorithm's sensitivity to sensor biases and misalignments is a notable challenge. Regular recalibration becomes essential to mitigate cumulative inaccuracies over time.

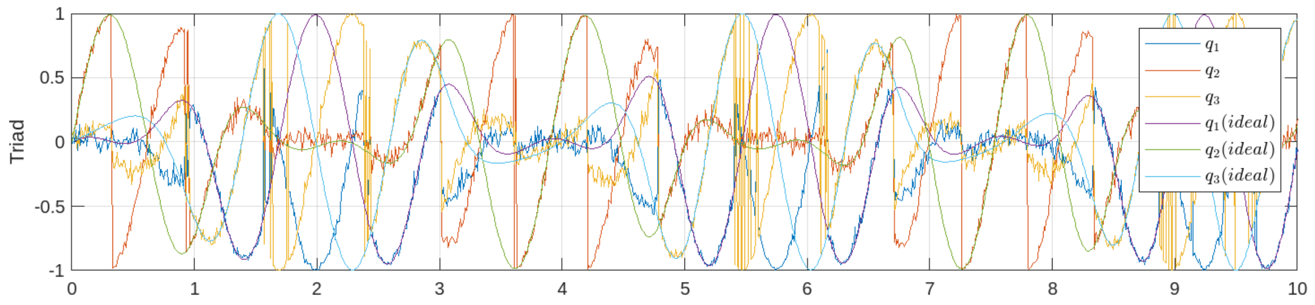

Figure 1: Triad Method with external moment, $\vec{n} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$

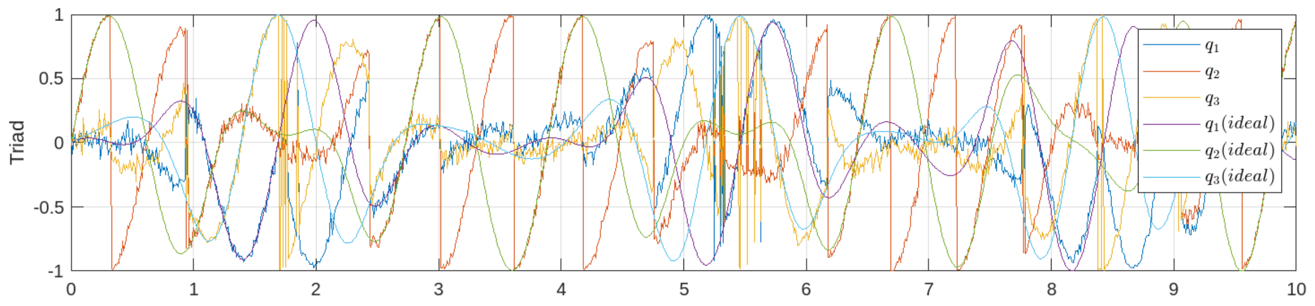

Figure 2: Triad Method with external moment, $\vec{n} = -K_P q_v$

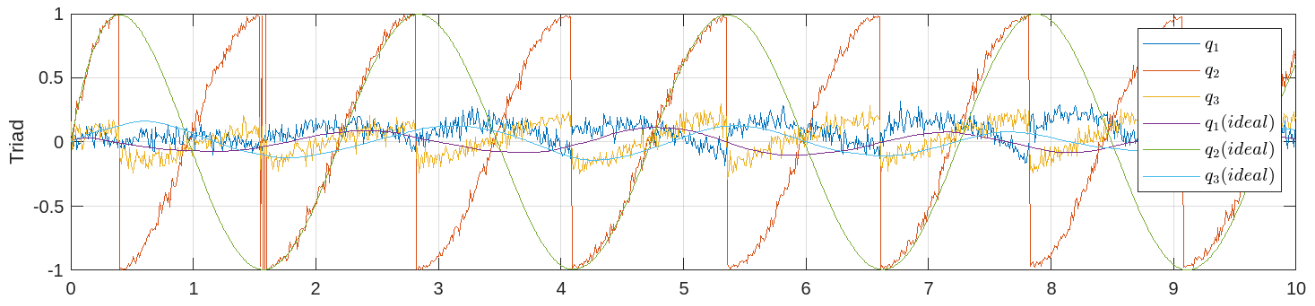

Figure 3: Triad Method with external moment, $\vec{n} = -K_P q_v - K_D(\hat{\omega} - \omega_{cmd})$

3.2 Q method

The Q-method, incorporating moments proportional to the change in attitude, is characterized by improved stability, particularly in scenarios with external disturbances. Its ability to capture sudden changes in attitude makes it a favorable choice for applications where rapid responses are critical.

Limitations:

However, the Q-method may encounter difficulties in scenarios involving continuous rotations, potentially overemphasizing abrupt changes while neglecting gradual variations. The reliance on moments proportional to attitude changes introduces sensitivity to noise.

Comparative Advantages and Disadvantages:

In comparison to methods without moment consideration, the Q-method provides a more nuanced understanding of attitude dynamics. Its versatility in handling varying environmental conditions sets it apart. Yet, the complexity of the Q-method and the requirement for a deep understanding of system dynamics pose implementation challenges. Calibration and parameter tuning become critical aspects, adding to the overall complexity.

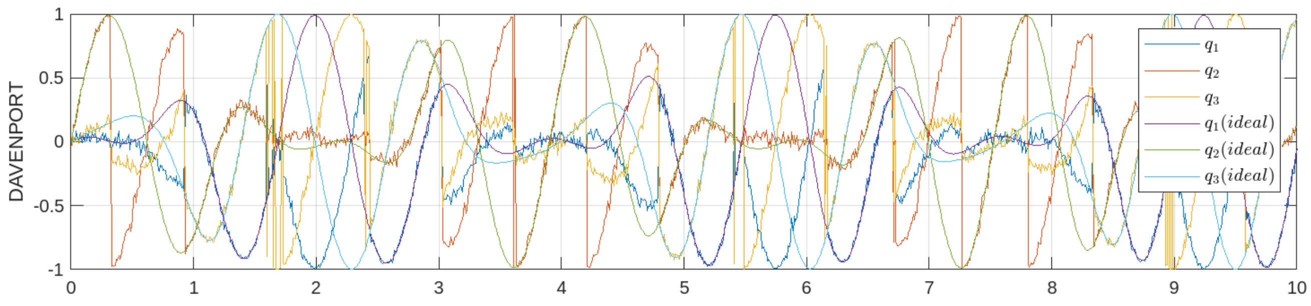

Figure 4: Q Method with external moment, $\vec{n} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$

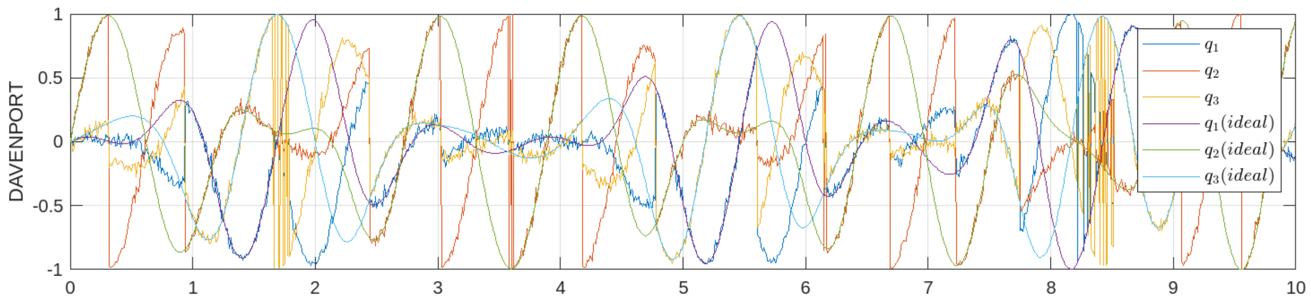

Figure 5: Q Method with external moment, $\vec{n} = -K_P q_v$

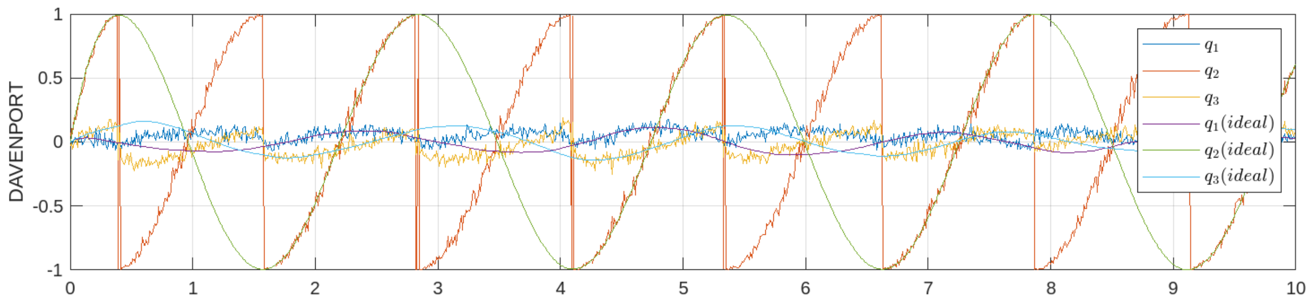

Figure 6: Q Method with external moment, $\vec{n} = -K_P q_v - K_D(\hat{\omega} - \omega_{cmd})$

3.3 Attitude Kalman Filter

The Attitude Kalman Filter (AKF), with moments proportional to both attitude changes and angular velocity, is notable for its adaptability in dynamic scenarios. It excels in providing comprehensive attitude estimation, incorporating both orientation and rate of change.

Limitations:

However, the AKF's performance heavily depends on accurate knowledge of system dynamics and noise characteristics. In scenarios with poorly understood dynamics, achieving optimal convergence may be challenging, impacting attitude estimates.

Comparative Advantages and Disadvantages:

Compared to other methods, the AKF offers superior performance in scenarios with continuous and dynamic attitude changes. Its integration of angular velocity information enhances accuracy, particularly in applications requiring precise control.

Yet, the computational demands of the AKF may limit its real-time applicability. The need for accurate parameterization and tuning further underscores the importance of a well-understood system for optimal performance.

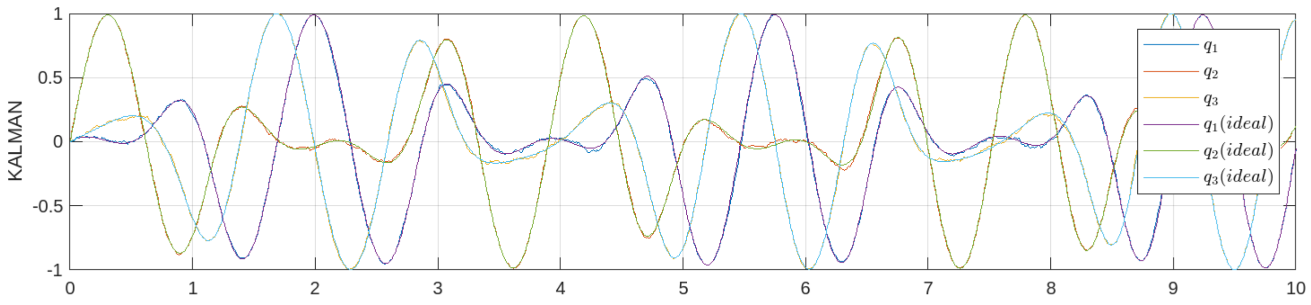

Figure 7: Attitude Kalman Filter with external moment, $\vec{n} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$

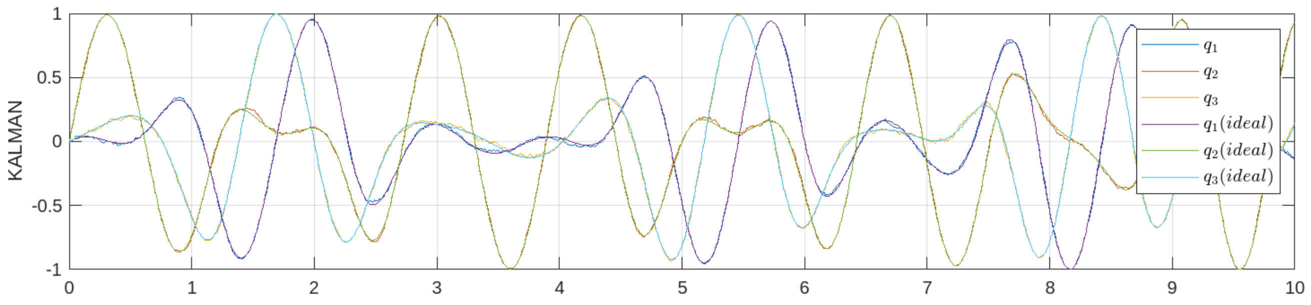

Figure 8: Attitude Kalman Filter with external moment, $\vec{n} = -K_P q_v$

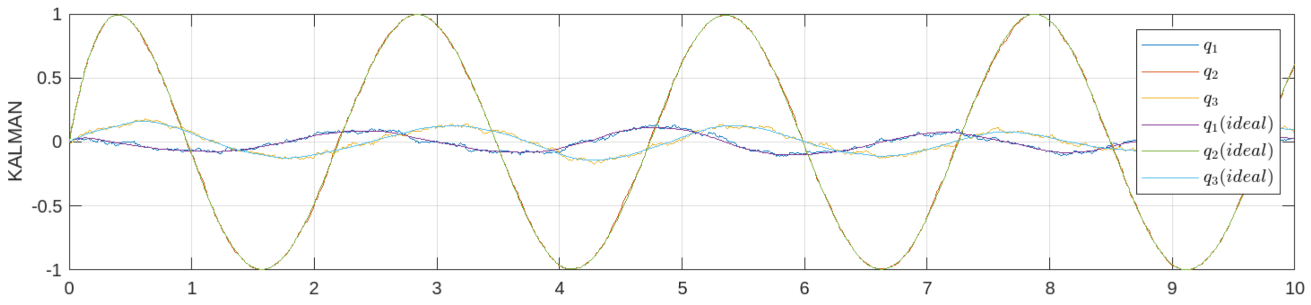

Figure 9: Attitude Kalman Filter with external moment, $\vec{n} = -K_P q_v - K_D(\hat{\omega} - \omega_{cmd})$

4 Appendix

```
clear *; close all;
g=[0; 0; 1];
h=[0; 1; 0];
k=[0; -1; 0];

% Body moment of inertias
J=diag([480,640,960]);

% moments
Kp=diag([1000, 0, 1000]);
Kd=2000;

nts = 1000; tf = 10; ts = tf/nts;
t=0:ts:tf;

%noise and disturbances
v_rms_sensor=0.1;
v_rms_gyro=1;

%initial conditions
q_o=[1; 0; 0; 0];
omg_o=[1; 10; 1];
st_o=[omg_o; q_o];
st=zeros(7,nts+1);
st(:,1)=st_o;
omg_meas=zeros(3,nts+1);
omg_meas(:,1)=st(1:3,1)+v_rms_gyro*randn(3,1); %noise
qhat_triad=zeros(4,nts+1);
qhat_qmethod=zeros(4,nts+1);
u_meas=zeros(3,nts+1);
v_meas=zeros(3,nts+1);
w_meas=zeros(3,nts+1);
n=-Kp*st(5:7,1)-Kd*(omg_meas(:,1)-[0; 5; 0]);

% Triad method Algorithm{
for i=1:nts
    st_dot=zeros(7,1);
    st_dot(1:3)=J^-1*(-cross(st(1:3,i),J*st(1:3,i))+n);
    st_dot(4:7)=1/2*[-st(5:7,i)'*st(1:3,i); st(4,i)*st(1:3,i)+cross(st(5:7,i),st
        (1:3,i))];
    st(:,i+1)=st(:,i) + st_dot*ts;
    st(4:7,i+1) = st(4:7,i+1)/norm(st(4:7,i+1)); % normalize attitude quaternion
    omg_meas(:,i+1)=st(1:3,i+1)+v_rms_gyro*randn(3,1); % noise
    C=(st(4,i)^2-st(5:7,i)'*st(5:7,i))*eye(3) + 2*(st(5:7,i)*st(5:7,i)') + 2*st(4,i
        )*vec_cross(st(5:7,i));
    u_meas(:,i)=C'*g+ v_rms_sensor*randn(3,1); % noise
    v_meas(:,i)=C'*h+ v_rms_sensor*randn(3,1); % noise
    w_meas(:,i)=C'*k+ v_rms_sensor*randn(3,1); % noise

    % C(attitude matrix) obtained from triad algorithm
    C_triad=Triad(g,h,u_meas(:,i),v_meas(:,i));

    % q obtained from triad algorithm
    qhat_triad(:,i)= quat_from_C(C_triad);

    %moment updated for next time step
    n=-Kp*qhat_triad(2:4,i)-Kd*(omg_meas(:,i)-[0; 5; 0]);
end
```

```
C=(st(4,i+1)^2-st(5:7,i+1)'*st(5:7,i+1))*eye(3)+2*(st(5:7,i+1)*st(5:7,i+1)')+2*st
    (4,i+1)*vec_cross(st(5:7,i+1));

u_meas(:,i+1)=C'*g+ v_rms_sensor*randn(3,1); %  noise
v_meas(:,i+1)=C'*h+ v_rms_sensor*randn(3,1); %  noise
w_meas(:,i+1)=C'*k+ v_rms_sensor*randn(3,1); %  noise

C_triad=Triad(g,h,u_meas(:,i+1),v_meas(:,i+1));
qhat_triad(:,i+1)= quat_from_C(C_triad);
%}

%{qmethod
for i=1:nts
    st_dot(1:3)=J^-1*(-cross(st(1:3,i),J*st(1:3,i))+n);
    st_dot(4:7)=1/2*[-st(5:7,i)'*st(1:3,i); st(4,i)*st(1:3,i)+cross(st(5:7,i),st
        (1:3,i))];
    st(:,i+1)=st(:,i) + st_dot*ts;
    st(4:7,i+1) = st(4:7,i+1)/norm(st(4:7,i+1)); % normalize attitude quaternion

    omg_meas(:,i+1)=st(1:3,i+1)+v_rms_gyro*randn(3,1); % noise

    C=(st(4,i)^2-st(5:7,i)'*st(5:7,i))*eye(3) + 2*(st(5:7,i)*st(5:7,i)') + 2*st(4,i
        )*vec_cross(st(5:7,i));
    u_meas(:,i)=C'*g+ v_rms_sensor*randn(3,1); % noise
    v_meas(:,i)=C'*h+ v_rms_sensor*randn(3,1); % noise
    w_meas(:,i)=C'*k+ v_rms_sensor*randn(3,1); % noise

    % Davenport matrix
    D=1/3*[0 u_meas(:,i)'; -u_meas(:,i) vec_cross(u_meas(:,i))]*[0 -g'; g vec_cross
        (g)] + 1/3*[0 v_meas(:,i)'; -v_meas(:,i) vec_cross(v_meas(:,i))]*[0 -h'; h
        vec_cross(h)] + 1/3*[0 w_meas(:,i)'; -w_meas(:,i) vec_cross(w_meas(:,i))
        ]*[0 -k'; k vec_cross(k)];

    [q,A_k_1]=eig(D); % q is a matrix whose columns are eigen vectors, A is
        diagonal matrix with eigen values
    [~, s]=max(diag(A_k_1)); % ~ stores max value and s stores its index, diag(A)
        will give [lambda1, lambda2, lambda3]

    qhat_qmethod(:,i)=q(:,s)*sign(q(1, s)); % q obtained from q_method

    %moment updated for next time step
    n=-Kp*qhat_qmethod(2:4,i)-Kd*(omg_meas(:,i)-[0; 5; 0]);
end

C=(st(4,i+1)^2-st(5:7,i+1)'*st(5:7,i+1))*eye(3)+2*(st(5:7,i+1)*st(5:7,i+1)')+2*st
    (4,i+1)*vec_cross(st(5:7,i+1));

u_meas(:,i+1)=C'*g+ v_rms_sensor*randn(3,1); %  noise
v_meas(:,i+1)=C'*h+ v_rms_sensor*randn(3,1); %  noise
w_meas(:,i+1)=C'*k+ v_rms_sensor*randn(3,1); %  noise

D=1/3*[0 u_meas(:,i)'; -u_meas(:,i) vec_cross(u_meas(:,i))]*[0 -g'; g vec_cross(g)]
     + 1/3*[0 v_meas(:,i)'; -v_meas(:,i) vec_cross(v_meas(:,i))]*[0 -h'; h
    vec_cross(h)] + 1/3*[0 w_meas(:,i)'; -w_meas(:,i) vec_cross(w_meas(:,i))]*[0 -k
    '; k vec_cross(k)];
[q,A_k_1]=eig(D);
[~, s]=max(diag(A_k_1));
qhat_qmethod(:,i+1)=q(:,s)*sign(q(1, s));
%}

% {Kalman Filter
phat_k_k_1=zeros(4,nts+1);
```

```
phat_k_k_1(:,1)=q_o;
phat_k=phat_k_k_1;
r_kv=zeros(4,nts+1);
R_k_k_1=zeros(4,nts+1);
r_kv(1,:)=1;
uhat_k_k_1=zeros(4,nts+1);
vhat_k_k_1=zeros(4,nts+1);
what_k_k_1=zeros(4,nts+1);
R_k_1=zeros(3);
n=-Kp*q_o(2:4,1)-Kd*(omg_meas(:,1 )-[0; 5; 0]);
E = v_rms_gyro^(2)*eye(3);
Tht = v_rms_sensor^(2)*eye(9);
for j=1:nts
    st_dot(1:3)=J^-1*(-cross(st(1:3,i),J*st(1:3,i))+n);
    st_dot(4:7)=1/2*[-st(5:7,i)'*st(1:3,i); st(4,i)*st(1:3,i)+cross(st(5:7,i),st
        (1:3,i))];
    st(:,i+1)=st(:,i) + st_dot*ts;
    st(4:7,i+1) = st(4:7,i+1)/norm(st(4:7,i+1)); % normalize attitude quaternion

    omg_meas(:,i+1)=st(1:3,i+1)+v_rms_gyro*randn(3,1); % noise

    % Predict Step:
    phat_k_k_1(:,j+1)=quat_multiply(phat_k(:,j),[cos(sqrt(omg_meas(:,j)'*omg_meas
        (:,j))*ts/2); sin(sqrt(omg_meas(:,j)'*omg_meas(:,j))*ts/2)*omg_meas(:,j)/(
        sqrt(omg_meas(:,j)'*omg_meas(:,j)))]);
    A_k_1=eye(3)+cross_vec(omg_meas(:,j))*ts;
    R_k_k_1=A_k_1*R_k_1*A_k_1'+E*ts^2/4;

    % Update step:
    uhat_k_k_1(:,j+1)=quat_multiply(quat_multiply([phat_k_k_1(1,j+1); -phat_k_k_1
        (2:4,j+1)],[0;g]),phat_k_k_1(:,j+1));
    vhat_k_k_1(:,j+1)=quat_multiply(quat_multiply([phat_k_k_1(1,j+1); -phat_k_k_1
        (2:4,j+1)],[0;h]),phat_k_k_1(:,j+1));
    what_k_k_1(:,j+1)=quat_multiply(quat_multiply([phat_k_k_1(1,j+1); -phat_k_k_1
        (2:4,j+1)],[0;k]),phat_k_k_1(:,j+1));

    Ck=2*[vec_cross(uhat_k_k_1(2:4,j+1)); vec_cross(vhat_k_k_1(2:4,j+1)); vec_cross
        (what_k_k_1(2:4,j+1))];

    Lk=((R_k_k_1^(-1) + Ck'*Tht^(-1)*Ck)^(-1))*Ck'*Tht^(-1);
    r_kv(2:4,j+1)=Lk*[u_meas(:,j+1)-uhat_k_k_1(2:4,j+1); v_meas(:,j+1)-vhat_k_k_1
        (2:4,j+1); w_meas(:,j+1)-what_k_k_1(2:4,j+1)];

    phat_k(:,j+1)=quat_multiply(phat_k_k_1(:,j+1),r_kv(:,j+1));
    phat_k(:,j+1)=phat_k(:,j+1)/norm(phat_k(:,j+1)); % normalize q
    n=-Kp*phat_k(2:4,j)-Kd*(omg_meas(:,j)-[0; 5; 0]);
    R_k_1 = R_k_k_1 - R_k_k_1*Ck'*(Ck*R_k_k_1*Ck' + Tht)^(-1)*Ck*R_k_k_1;
end
%}

% plot for triad method
subplot(3,1,1)
plot(t,qhat_triad(2:4,:))
hold on
plot(t,st(5:7,:))
hold off
grid on
ylabel('Triad')
legend({'$q_1$','$q_2$','$q_3$','$q_1(ideal)$','$q_2(ideal)$','$q_3(ideal)$'}, ...
'Interpreter', 'latex', 'FontSize', 10);

%plot for qmethod
```

```
subplot(3,1,2)
plot(t,qhat_qmethod(2:4,:))
hold on
plot(t,st(5:7,:))
hold off
grid on
ylabel('DAVENPORT')
legend({'$q_1$','$q_2$','$q_3$','$q_1(ideal)$','$q_2(ideal)$','$q_3(ideal)$'}, ...
'Interpreter', 'latex', 'FontSize', 10);
%}

subplot(3,1,3)
%plot for Kalman filter
plot(t,phat_k(2:4,:))
hold on
plot(t,st(5:7,:))
hold off
grid on
ylabel('KALMAN')
legend({'$q_1$','$q_2$','$q_3$','$q_1(ideal)$','$q_2(ideal)$','$q_3(ideal)$'}, ...
'Interpreter', 'latex', 'FontSize', 10);

function q=quat_from_C(C)
% for obtaining quaternion from C matrix
c_phi=(sum(diag(C))-1)/2;
s_phi=sqrt(1-c_phi^2);
nx=(C-C')/(2*s_phi);
qvec = sqrt((1-c_phi)/2)*[nx(3,2); nx(1,3); nx(2,1)];
q=[sqrt((c_phi+1)/2); qvec]; %
end

function v=vec_cross(a)
%for obtaining [vx] of a vector
v=[0 -a(3) a(2); a(3) 0 -a(1); -a(2) a(1) 0];
end

function C_t=Triad(g,h,u,v)
%for obtaining the attitude matrix by triad method
X=[g/sqrt(sum(g.^2)), cross(g,h)/sqrt(sum(cross(g,h).^2)), ...
cross(g,cross(g,h))/(sqrt(sum(g.^2))*sqrt(sum(cross(g,h).^2)))];
Y=[u/sqrt(sum(u.^2)), cross(u,v)/sqrt(sum(cross(u,v).^2)), ...
cross(u,cross(u,v))/(sqrt(sum(u.^2))*sqrt(sum(cross(u,v).^2)))];
% attitude matrix
C_t=X*Y';
end

function Q=quat_multiply(a,b)
%quaternion muliplication
Q=[a(1)*b(1)-a(2:4)'*b(2:4); a(1)*b(2:4)+b(1)*a(2:4)+cross(a(2:4),b(2:4))];
end

function f=cross_vec(a)
%for obtaining [xv] of a vector
v=[0 -a(3) a(2); a(3) 0 -a(1); -a(2) a(1) 0];
f=v';
end
```

Listing 1: MATLAB code