DOC 221 Dinámica orbital y control de actitud

Solutions to Problems Lecture ADCS - VII

Problem 1:

(a) The spacecraft body-frame is obtained from the inertial frame by a principle rotation about vector $\mathbf{z}_G$ through angle 45^o. Therefore, the corresponding rotation matrix is

$$\mathbf{C}_{bG} = \mathbf{C}_z(45^o) = \begin{bmatrix} \cos 45^o & \sin 45^o & 0 \\ -\sin 45^o & \cos 45^o & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(b) We are given the ECI coordinates of vector $\mathbf{n}_e$ and $\mathbf{n}_s$ as

$$\mathbf{n}_{e,G} = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \quad , \quad \mathbf{n}_{s,G} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

They are in spacecraft coordinates

$$\mathbf{n}_{e,b} = \mathbf{C}_{bG}\mathbf{n}_{e,G} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix}$$

$$\mathbf{n}_{s,b} = \mathbf{C}_{bG}\mathbf{n}_{s,G} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix}$$

(c) The body frame triad is given by:

$$\vec{\mathbf{t}}_{1b} = \vec{\mathbf{n}}_{e,b} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix}$$

$$\vec{\mathbf{t}}_{2b} = \frac{\vec{\mathbf{n}}_{e,b} \times \vec{\mathbf{n}}_{s,b}}{|\vec{\mathbf{n}}_{e,b} \times \vec{\mathbf{n}}_{s,b}|} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

$$\vec{\mathbf{t}}_{3b} = \vec{\mathbf{t}}_{1b} \times \vec{\mathbf{t}}_{2b} = \begin{bmatrix} -1/\sqrt{2} \\ -1/\sqrt{2} \\ 0 \end{bmatrix}$$

The reference frame triad is given by:

$$\vec{\mathbf{t}}_{1i} = \vec{\mathbf{n}}_{e,G} = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$$

$$\vec{\mathbf{t}}_{2i} = \frac{\vec{\mathbf{n}}_{e,G} \times \vec{\mathbf{n}}_{s,G}}{|\vec{\mathbf{n}}_{e,G} \times \vec{\mathbf{n}}_{s,G}|} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

$$\vec{\mathbf{t}}_{3i} = \vec{\mathbf{t}}_{1i} \times \vec{\mathbf{t}}_{2i} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

(d) The rotation matrices are given by:

$$[\vec{\mathbf{t}}_{1b}\ \vec{\mathbf{t}}_{2b}\ \vec{\mathbf{t}}_{3b}] = \begin{bmatrix} -1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 0 & -1 & 0 \end{bmatrix}$$

$$[\vec{\mathbf{t}}_{1i}\ \vec{\mathbf{t}}_{2i}\ \vec{\mathbf{t}}_{3i}] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

(e) The rotation matrix C_{bG} given by the TRIAD method is:

$$\mathbf{C}_{bG} = [\vec{\mathbf{t}}_{1b}\ \vec{\mathbf{t}}_{2b}\ \vec{\mathbf{t}}_{3b}][\vec{\mathbf{t}}_{1i}\ \vec{\mathbf{t}}_{2i}\ \vec{\mathbf{t}}_{3i}]^T = \begin{bmatrix} -1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which agrees exactly with part **(a).**

Problem 2:

Applying the TRIAD algorithm, we construct the components of the vectors $\mathbf{t}_{jb}$ and $\mathbf{t}_{ji}$ with j=1,2,3 in both the body and inertial frames:

$$\mathbf{t}_{1b} = \begin{bmatrix} 0.8273 & 0.5541 & -0.0920 \end{bmatrix}^{\mathrm{T}}$$
$$\mathbf{t}_{2b} = \begin{bmatrix} -0.0023 & 0.1671 & 0.9859 \end{bmatrix}^{\mathrm{T}}$$
$$\mathbf{t}_{3b} = \begin{bmatrix} 0.5617 & -0.8155 & 0.1395 \end{bmatrix}^{\mathrm{T}}$$

and

$$\mathbf{t}_{1i} = \begin{bmatrix} -0.1517 & -0.9669 & 0.2050 \end{bmatrix}^{\mathrm{T}}$$
$$\mathbf{t}_{2i} = \begin{bmatrix} 0.2177 & -0.2350 & -0.9473 \end{bmatrix}^{\mathrm{T}}$$
$$\mathbf{t}_{3i} = \begin{bmatrix} 0.9641 & -0.0991 & 0.2462 \end{bmatrix}^{\mathrm{T}}$$

Using these results with the equation $\mathbf{C}_{bi} = [\vec{\mathbf{t}}_{1b}\ \vec{\mathbf{t}}_{2b}\ \vec{\mathbf{t}}_{3b}][\vec{\mathbf{t}}_{1i}\ \vec{\mathbf{t}}_{2i}\ \vec{\mathbf{t}}_{3i}]^T$, we obtain the approximate rotation matrix

$$\mathbf{C}_{bi} = [\vec{\mathbf{t}}_{1b}\ \vec{\mathbf{t}}_{2b}\ \vec{\mathbf{t}}_{3b}][\vec{\mathbf{t}}_{1i}\ \vec{\mathbf{t}}_{2i}\ \vec{\mathbf{t}}_{3i}]^T =$$

$$\begin{bmatrix} 0.4156 & -0.8551 & 03100 \\ -0.8339 & -0.4943 & -0.2455 \\ 0.3631 & -0.1566 & -0.9185 \end{bmatrix}$$

Applying this rotation matrix to $\mathbf{v}_{1i}$ gives $\mathbf{v}_{1b}$ exactly, because we used this condition in the formulation; however, applying it to $\mathbf{v}_{2i}$ does not give $\mathbf{v}_{2b}$ exactly. If we know a priori that sensor 2 is more accurate than sensor 1, then we can use $\mathbf{v}_2$ as the exact measurement, hopefully leading to a more accurate estimate of $\mathbf{C}_{bi}$.

Problem 3:

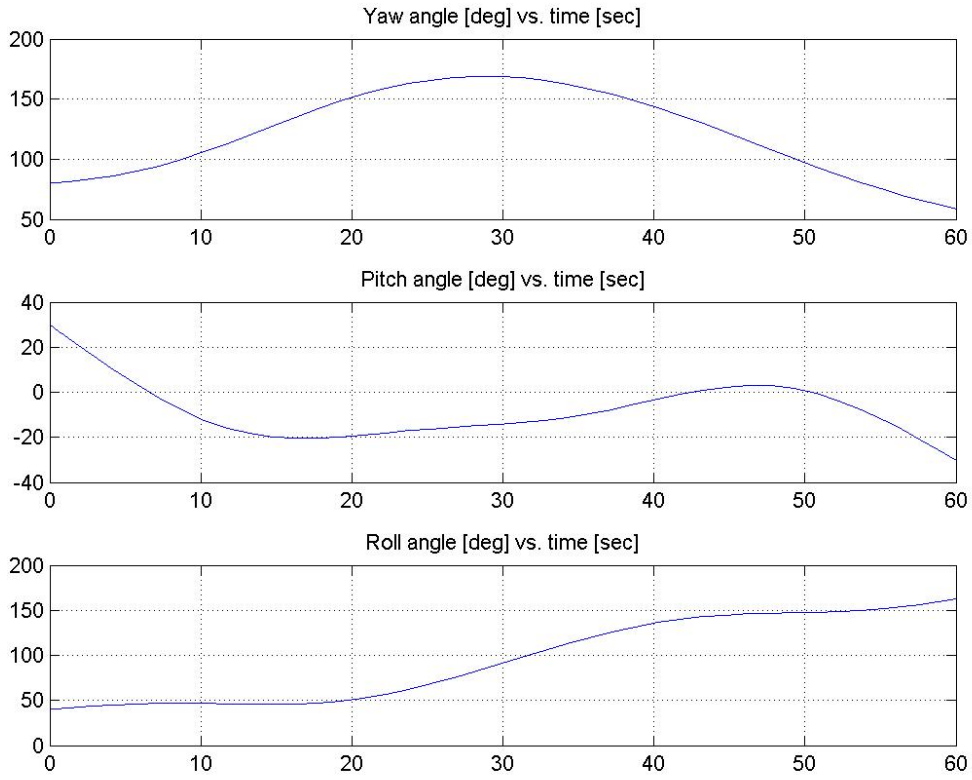
The differential kinematics equation for the 3-2-1 Euler angles is given by

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = \frac{1}{\cos\theta_2} \begin{bmatrix} \cos\theta_2 & \sin\theta_1\sin\theta_2 & \cos\theta_1\sin\theta_2 \\ 0 & \cos\theta_1\cos\theta_2 & -\sin\theta_1\cos\theta_2 \\ 0 & \sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

The initial condition is (Θ_1, Θ_2, Θ_3) = (80^o, 30^o, 40^o). Given the angular velocity vector:

$$\boldsymbol{\omega} = \begin{bmatrix} \sin(0.1t) \\ 0.01 \\ \cos(0.1t) \end{bmatrix} 5^o / s$$

We find the time history of yaw (Θ_1), pitch (Θ_2) and roll (Θ_3) angles shown in the figure below.



Matlab script:

```
function ProblemLecture7
% initial Euler angles in radians
theta1 = degtorad(80);
theta2 = degtorad(30);
theta3 = degtorad(40);

[time,angles] = ode45(@diff_eqn,[0 60], [theta1;theta2;theta3], odeset('RelTol',1e-4));

%plotting
subplot(311);plot(time,radtodeg(angles(:,1)));grid;title('Yaw angle [deg] vs. time [sec]')
subplot(312);plot(time,radtodeg(angles(:,2)));grid;title('Pitch angle [deg] vs. time [sec]')
subplot(313);plot(time,radtodeg(angles(:,3)));grid;title('Roll angle [deg] vs. time [sec]')

% kinemtaic differential equation for 3-2-1 sequence
function dot_angles = diff_eqn(time,angles)
theta1 = angles(1);
theta2 = angles(2);
theta3 = angles(3);
w = [sin(0.1*time); 0.01; cos(0.1*time)]*degtorad(5);

dot_angles = 1/cos(theta2)* [ cos(theta2)   sin(theta1)*sin(theta2)     cos(theta1)*sin(theta2)
                              0             cos(theta1)*cos(theta2)   -sin(theta1)*cos(theta2)
                              0             sin(theta1)                cos(theta1)       ]*w;
```

Problem 4:

(a) To translate from Euler angles to quaternion (Euler parameters), we first compute the corresponding rotation matrix from the 3-2-1 Euler angles as

$$\begin{bmatrix} 0.663414 & 0.55667 & -0.5 \\ 0.265584 & 0.449533 & 0.852869 \\ 0.699533 & -0.698597 & 0.150384 \end{bmatrix}$$

Next the quaternion (Euler parameters) are extracted by

$$q_4 = \pm\tfrac{1}{2}(1 + C_{11} + C_{22} + C_{33})^{\frac{1}{2}}$$

$$\vec{q} = \tfrac{1}{4q_4}\begin{bmatrix} C_{23} - C_{32} \\ C_{31} - C_{13} \\ C_{12} - C_{21} \end{bmatrix}$$

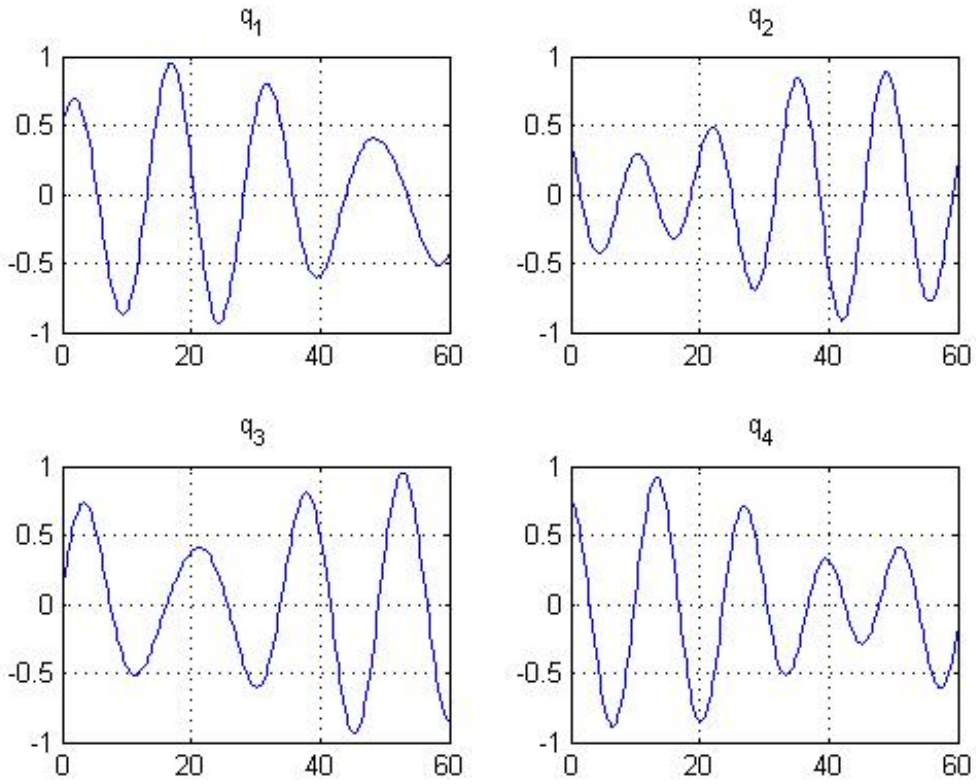
The corresponding initial quaternion (Euler parameters) is

$$[0.51563, 0.398665, 0.0967425, 0.752219]^{T}$$

(b) Numerically integrating the quaternion (Euler parameters) with the provided ω(t) using following equation

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

The results are shown in following figure



Matlab script:

```
function ProblemLecture7quat
% initial Euler q in radians
q_0 = [0.51563, 0.398665, 0.0967425, 0.752219];

[time,q] = ode45(@diff_eqn,[0 60], q_0, odeset('RelTol',1e-4));

%plotting
subplot(221);plot(time,q(:,1));grid;title('q_1')
subplot(222);plot(time,q(:,2));grid;title('q_2')
subplot(223);plot(time,q(:,3));grid;title('q_3')
subplot(224);plot(time,q(:,4));grid;title('q_4')
figure(2);plot(time,q(:,1).^2+q(:,2).^2+q(:,3).^2+q(:,4).^2);grid;title('|q|')
% kinematic differential equation for quaternion
function dot_q = diff_eqn(time,q)
w = [sin(0.1*time); 0.01; cos(0.1*time); 0.0]*degtorad(50);
w1 = w(1);
w2 = w(2);
w3 = w(3);
w4 = w(4);
dot_q = 0.5 * [   0   w3   -w2    w1
                -w3   0     w1    w2
                 w2  -w1    0     w3
                -w1  -w2   -w3    0 ]*q;
```

(c) The plot of the $|\mathbf{q}|$ expression is shown in the figure below. The length of the quaternion should be always one. In numerical integration such as Matlab ODE45, variable time step sizes are used to keep the numerical integration error small. The unit constraint violation however is growing over time and requires periodic corrections.