Table of Contents

Roshan Jaiswal-Ferri & Stefan Rosu

```
%Section - 01
%Aero 421 Sim Hack: 6/11/25
```

Workspace Prep

```
%warning off
format long      %Allows for more accurate decimals
close all;       %Clears all
clear all;       %Clears Workspace
clc;             %Clears Command Window
```

Description

The mehiel sat starts pointing nadir wher it has 30 seconds point at the target using command and absolute quaternions as well as the rotation matricies. During the 240 second time period it must stay facing at the target (while eventually pointing nadir right at the target) and then find the command quaternion to point nadir at the end of the active pointing phase.

Mass properties for normal operations phase (Question 2)

```
data = load("final_question.mat");

zbar = 0.23438;
cm = [0; 0; zbar];
mw = 1.4;
total_mass = 640;
total_mass = total_mass + 3*mw;
J = [812.0396 0 0
0 545.3729 0
0 0 627.7083];
eps = [0;sqrt(0.5);0];
```

```
eta = sqrt(0.5);
q_c = [eps; eta];

ts = data.t_slew; % settling time
zeta = 0.65; % Damping ratio
wn = log(0.02*sqrt(1-zeta^2))/-zeta/ts;
beta = atan(sqrt(1-zeta^2)/zeta);
tr = (pi-beta)/wn/sqrt(1-zeta^2);
syms Mp1
eqn = zeta == sqrt(log(Mp1)^2/(pi^2 + log(Mp1)^2));
Mp = double(solve(eqn, Mp1));
Kp = 2*J*eye(3)*wn^2;
Kd = J*eye(3)*2*zeta*wn;
max_T = 1; %Nms
```

Question 3

```
R1 = data.r_ECI_0;
R2 = data.r_ECI_1;
R3 = data.r_ECI_2;
R4 = data.r_ECI_3;
Rt = data.r_ECI_t;

V1 = data.v_ECI_0;
V2 = data.v_ECI_1;
V3 = data.v_ECI_2;
V4 = data.v_ECI_3;
Vt = data.v_ECI_t;

T1 = data.T_0;
T2 = data.T_1;
T3 = data.T_2;
T4 = data.T_3;
Tt = data.T_t;

[q1] = rotationM(R1,V1,T1)
[q2] = rotationM(R2,V2,T2)
[q3] = rotationM(R3,V3,T3)
[q4] = rotationM(R4,V4,T4)
[qt] = rotationM(Rt,Vt,Tt) %overhead

%multiply each by the old one to find absolute
```

Initial Mat

```
rV = R1; %Position Vector km
vV = V1; %Vel Vector km/s

Zlvlh = -(rV/norm(rV));
Ylvlh = -(cross(rV,vV)/norm(cross(rV,vV)));
Xlvlh = cross(Ylvlh,Zlvlh);

%Creating Matrix with new vectors
```

```
C_LVLH_ECI_0 = [Xlvlh, Ylvlh, Zlvlh];

% Initial Euler angles relating F_body and F_LVLH (given)
phi_0 = 0;
theta_0 = 0;
psi_0 = 0;
E_b_LVLH_0 = [phi_0; theta_0; psi_0];

% Initial Quaternion relating F_body and F_LVLH (given)
q_b_LVLH_0 = [0; 0; 0; 1];

% Compute initial C_LVLH_ECI_0, C_b_LHVL_0, and C_b_ECI_0 rotaiton matrices
%C_LVLH_ECI_0 = [x_LVLH'; y_LVLH'; z_LVLH'];
C_b_LVLH_0 =
rotx(rad2deg(phi_0))'*roty(rad2deg(theta_0))'*rotz(rad2deg(psi_0))';
C_b_ECI_0 = C_b_LVLH_0*C_LVLH_ECI_0;

% Initial Euler angles relating body to ECI
E_b_ECI_1 = C2EulerAngles(C_b_ECI_0);

% Initial quaternion relating body to E
q_b_ECI_1 = rotm2quat(C_b_ECI_0);
```

Rotation while overhead

```
rV = Rt; %Position Vector km
vV = Vt; %Vel Vector km/s

Zlvlh = -(rV/norm(rV));
Ylvlh = -(cross(rV,vV)/norm(cross(rV,vV)));
Xlvlh = cross(Ylvlh,Zlvlh);

%Creating Matrix with new vectors

C_LVLH_ECI_0 = [Xlvlh, Ylvlh, Zlvlh];

% Initial Euler angles relating F_body and F_LVLH (given)
phi_0 = 0;
theta_0 = 0;
psi_0 = 0;
E_b_LVLH_0 = [phi_0; theta_0; psi_0];

% Initial Quaternion relating F_body and F_LVLH (given)
q_b_LVLH_0 = [0; 0; 0; 1];

% Compute initial C_LVLH_ECI_0, C_b_LHVL_0, and C_b_ECI_0 rotaiton matrices
%C_LVLH_ECI_0 = [x_LVLH'; y_LVLH'; z_LVLH'];
C_b_LVLH_0 =
rotx(rad2deg(phi_0))'*roty(rad2deg(theta_0))'*rotz(rad2deg(psi_0))';
C_b_ECI_0 = C_b_LVLH_0*C_LVLH_ECI_0;

% Initial Euler angles relating body to ECI
```

```
E_b_ECI_t = C2EulerAngles(C_b_ECI_0);

% Initial quaternion relating body to E
q_b_ECI_t = rotm2quat(C_b_ECI_0);
```

Questions 4 & 6

```
Cq1 = quatMult(q_b_ECI_1',q1)
Cq2 = quatMult(q_b_ECI_t',qt)
```

```
Cq1 =

   0.513688739131143
  -0.508049628119357
  -0.641805578862395
  -0.257089582828137


Cq2 =

   0.179184138816702
  -0.738673460628894
  -0.508789701669503
  -0.404212323456507
```

Question 5

First, I would make t_stare into a linspace Then I would propage r and v using orbits functions and ode 45 to get a function of r and v for the duration of the t_stare time Then from here, I would calculate the new quaternion at every r as a function of time. Then I would feed this into our control law in simulink, and use the clock function to make sure that the times would line up. From here it would be a standard feed back using quaternion multiple to get the the error quaternion between the commanded quaternion and the current quaternion, as shown below: q_e = quatMult(q, quatConjugate(q_c)); Mc = -Kp*q_e(1:3,1) - Kd*omega;

```
% Define the sun in F_ECI and residual dipole moment in F_b
sun_ECI = [0 0 -1];

% Current JD - has to be on the solar equinox, why? - we'll use 3/20/2024
% from https://aa.usno.navy.mil/data/JulianDate
% Need this so we can convert from F_ECEF to F_ECI and to F_b for the
% magnetic field model
JD_0 = 2460390;

m_b = [0; 0; -0.5];

% Spacecraft Orbit Properties (given)
mu = 398600; % km^3/s^2
h = 53335.2; % km^2/s
e = 0; % none
Omega = 0*pi/180; % radians
inc = 98.43*pi/180; % radians
```

```
omega = 0*pi/180; % radians
nu = 0*pi/180; % radians

a = h^2/mu/(1 - e^2);
orbital_period = 2*pi*sqrt(a^3/mu);

% Torque free scenario (Given)
T = [0;0;0];

% Set/Compute initial conditions
% intial orbital position and velocity
[r_ECI_0, v_ECI_0] = coes2rvd(a,e,rad2deg(inc),0,omega,nu,mu);

% Compute inital F_LVLH basis vectors in F_ECI components based on F_LVLH
% definition

rV = r_ECI_0; %Position Vector km
vV = v_ECI_0; %Vel Vector km/s

Zlvlh = -(rV/norm(rV));
Ylvlh = -(cross(rV,vV)/norm(cross(rV,vV)));
Xlvlh = cross(Ylvlh,Zlvlh);

%Creating Matrix with new vectors

C_LVLH_ECI_0 = [Xlvlh, Ylvlh, Zlvlh];

% Initial Euler angles relating F_body and F_LVLH (given)
phi_0 = 0;
theta_0 = 0;
psi_0 = 0;
E_b_LVLH_0 = [phi_0; theta_0; psi_0];

% Initial Quaternion relating F_body and F_LVLH (given)
q_b_LVLH_0 = [0; 0; 0; 1];

% Compute initial C_LVLH_ECI_0, C_b_LHVL_0, and C_b_ECI_0 rotaiton matrices
%C_LVLH_ECI_0 = [x_LVLH'; y_LVLH'; z_LVLH'];
C_b_LVLH_0 =
rotx(rad2deg(phi_0))'*roty(rad2deg(theta_0))'*rotz(rad2deg(psi_0))';
C_b_ECI_0 = C_b_LVLH_0*C_LVLH_ECI_0;

% Initial Euler angles relating body to ECI
E_b_ECI_0 = C2EulerAngles(C_b_ECI_0);

% Initial quaternion relating body to E
q_b_ECI_0 = rotm2quat(C_b_ECI_0);

% Initial body rates of spacecraft (given)
w_b_ECI_0 = [0.001; -0.001; 0.002];

% Set simulation time period
N = 10; % Number of Orbits
```

```
tspan = orbital_period*N;
%tspan = 300;
```

Function

```
function [q] = rotationM(R,V,T)
    P = T - R;
    phi = acos(dot(-R,P)/(norm(R)*norm(P)));

    ahat = sin(phi/2)*(cross(R,T)/norm(cross(R,T)));
    eta = cos(phi/2);

    q = [ahat;eta];
end
```

```
q1 =

  -0.382301163730618
   0.166069657000762
  -0.081916556149822
   0.905293525362033


q2 =

  -0.344388258004056
   0.149008172093951
  -0.073392004519776
   0.924016723912896


q3 =

   0.346473142256169
  -0.145493961826100
   0.070270015470177
   0.924040039009730


q4 =

   0.385210731261753
  -0.161195325443835
   0.077583671591288
   0.905322889072322


qt =

                   0
                   0
   0.000000010536712
```

```
1.000000000000000
```

Published with MATLAB® R2024b