Table of Contents

Roshan Jaiswal-Ferri & Stefan Rosu

```
%Section - 01
%Aero 421 FP3: 4/2/25
%Note: This was prepared with the help of Dr. Mehiel's Template script
```

Workspace Prep

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

global mu
mu = 398600; % km^3/s^2
```

Part 1 - Mass Properties

```
% Mass properties for normal operations phase

% Calculate the total mass, inertia and Center of Mass of the MehielSat

% You will need to change this
zbar = 0.23438;
cm = [0; 0; zbar];
total_mass = 640;
J = [812.0396             0             0
       0      545.3729             0
       0             0      627.7083];

% fprintf('The spacecraft mass for the normal operations mode is:\n')
% display(total_mass)
% fprintf('The spacecraft center of mass for the normal operations mode
is:\n')
% display(cm)
% fprintf('The Inertia Matrix for the normal operations mode is:\n')
% display(J)
```

Part 2 - Geometric Properties of MehielSat during Normal Operations

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

% I constructed a matrix where the rows represent each surface of the
% MehielSat.  The first column stores the Area of the surface, the next
% three columns define the normal vector of that surface in F_b, and the
% final three columns store the center of presure of the surface (geometric
% center of the surface) in F_b.

% First get rhos vectors with respect to the center of the spacecraft bus
% the MehielSat BUS is a box
Areas = 4*ones(6,1);
normals = [1 0 0; -1 0 0; 0 1 0; 0 -1 0; 0 0 1; 0 0 -1];
cps = [1 0 0; -1 0 0; 0 1 0; 0 -1 0; 0 0 1; 0 0 -1];

% Append geometric properties for Solar Panel 1
Areas1 = 6*ones(2,1);
normals1 = [0 0 1; 0 0 -1];
cps1 = [0 2.5 0.005/2; 0 2.5 -0.005/2]; % Fix this

% Append geometric properties for Solar Panel 2
Areas2 = 6*ones(2,1);
normals2 = [0 0 1; 0 0 -1];
cps2 = [0 -2.5 0.005/2; 0 -2.5 -0.005/2]; % Do I need to use actual position
of geo-center, or just the axis?

% Append geometric properties for Sensor
Areas3 = [0.25*ones(4,1); 0.25*0.25];
normals3 = [1 0 0; 0 1 0; -1 0 0; 0 -1 0; 0 0 1];
cps3 = [0 0.25/2 1.5; 0.25/2 0 1.5; -0.25/2 0 1.5; 0 -0.25/2 1.5; 0 0 2];

% now subtract the center of mass to get the location of the rho vectors
% with respect to the center of mass
normals = normals - [0  0 zbar];
normals1 = normals1 - [0  0 zbar];
normals2 = normals2 - [0  0 zbar];
normals3 = normals3 - [0  0 zbar];

cps = cps - cm';
cps1 = cps1 - cm';
cps2 = cps2 - cm';
cps3 = cps3 - cm';


% Now build the matrix
surfaceProperties = [Areas cps normals;
                     Areas1 cps1 normals1;
                     Areas2 cps2 normals2;
                     Areas3 cps3 normals3];
```

Part 3 - Initialize Simulation States

```
% 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;

% Spacecraft Orbit Properties
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);

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


% No external command Torque
T_c = [0; 0; 0]; % Nm

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

Z_lvlh = -r_ECI_0/norm(r_ECI_0);
Y_lvlh = -cross(r_ECI_0,v_ECI_0)/norm(cross(r_ECI_0,v_ECI_0));
X_lvlh = cross(Y_lvlh,Z_lvlh);

C_LVLH_ECI_0 = [X_lvlh,Y_lvlh,Z_lvlh];

% 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];

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 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
```

```
% 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];
```

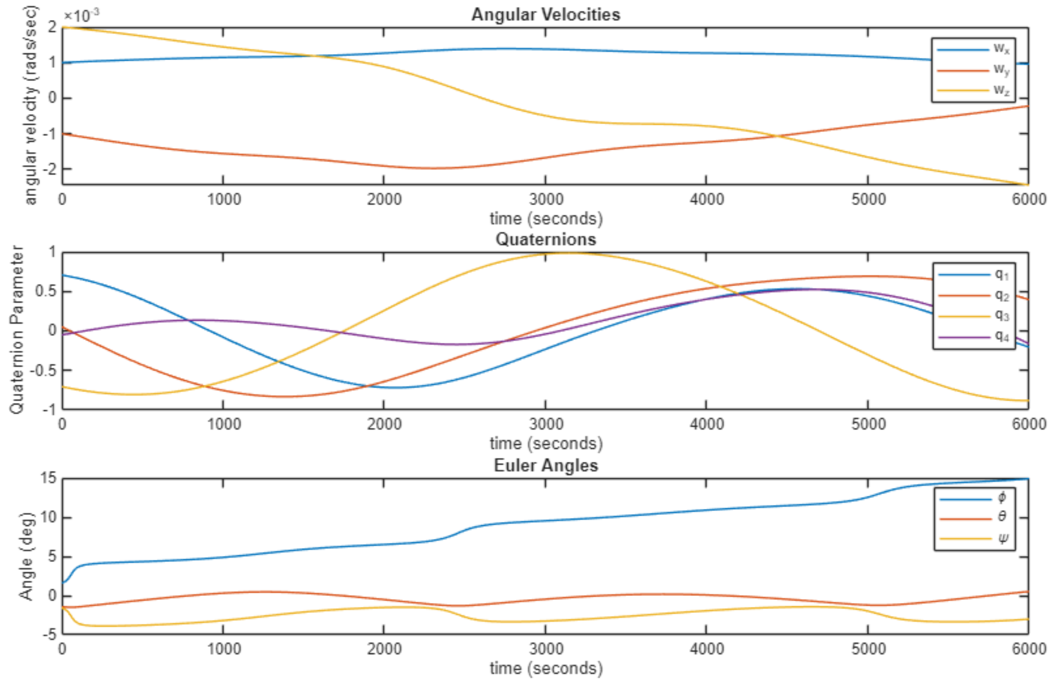
Part 4 - Simulate Results

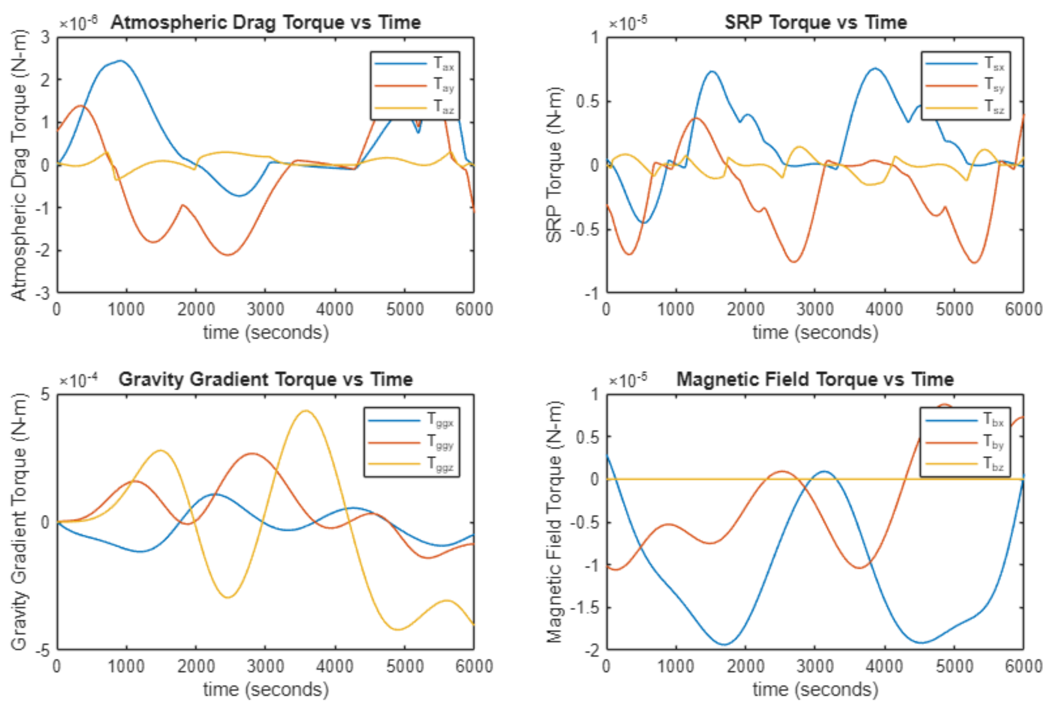
```
m_b = [0; 0; -0.5];
n_revs = 1; %revs %% CHANGE TO 5
tspan = n_revs * orbital_period;
out = sim('AERO421_FP3.slx');
```

Part 5 - Plot Results

```
% Plot Angular Velocities, Euler Angles and Quaternions

% Plot Disturbance torques in F_b

eul = squeeze(out.E_b_ECI.signals.values);
qua = squeeze(out.q_b_ECI.signals.values);
ang = squeeze(out.w_b_ECI.signals.values);


figure('Name','Angles & Rates')
subplot(3,1,1)
plot(out.tout,ang)
xlabel('time (seconds)')
ylabel('angular velocity (rads/sec)')
title('Angular Velocities')
legend('w_x','w_y','w_z')

subplot(3,1,2)
plot(out.tout,qua)
xlabel('time (seconds)')
ylabel('Quaternion Parameter')
title('Quaternions')
legend('q_1','q_2','q_3','q_4')

subplot(3,1,3)
plot(out.tout,eul)
xlabel('time (seconds)')
ylabel('Angle (deg)')
title('Euler Angles')
legend('\phi','\theta','\psi')

%

At = squeeze(out.T_a.signals.values);
SRPt = squeeze(out.T_srp.signals.values);
```

```
GGt = squeeze(out.T_gg.signals.values);
MFt = squeeze(out.T_b.signals.values);

figure('Name','Torques N Stuff')
subplot(2,2,1)
plot(out.tout,At)
xlabel('time (seconds)')
ylabel('Atmospheric Drag Torque (N-m)')
title('Atmospheric Drag Torque vs Time')
legend('T_a_x','T_a_y','T_a_z')

subplot(2,2,2)
plot(out.tout,SRPt)
xlabel('time (seconds)')
ylabel('SRP Torque (N-m)')
title('SRP Torque vs Time')
legend('T_s_x','T_s_y','T_s_z')

subplot(2,2,3)
plot(out.tout,GGt)
xlabel('time (seconds)')
ylabel('Gravity Gradient Torque (N-m)')
title('Gravity Gradient Torque vs Time')
legend('T_g_g_x','T_g_g_y','T_g_g_z')

subplot(2,2,4)
plot(out.tout,MFt)
xlabel('time (seconds)')
ylabel('Magnetic Field Torque (N-m)')
title('Magnetic Field Torque vs Time')
legend('T_b_x','T_b_y','T_b_z')
```





Published with MATLAB® R2024b