Aero 421 Final Project Discussion

Henry Macanas
Michael Johnston
Eric Ekstrom

Thrusters:

Three cold gas thrusters by MOOG were chosen to detumble the spacecraft. The three thrusters were chosen based on how much torque would need to be applied about the x, y, and z body axes of the spacecraft in order to stabilize it. Using these thrusters it was determined that the amount of fuel each thruster uses is 0.000074, 0.000135, and 0.000443 kg and the total amount of fuel used by the thrusters is 0.000651 kg. See far left thruster in the link below for data specifications.

http://www.moog.com/content/dam/moog/literature/Space_Defense/Spacecraft/Propulsion/ColdGasThrusters_0717.pdf

Reaction Wheels:

Reaction wheels were used on the spacecraft in order to counteract the various disturbance torques it encounters during orbit. Spinning wheels on a spacecraft make the inertia about each spin axis appear more oblate, making it harder for disturbances to influence the spacecraft's attitude. Additionally whenever the spacecraft attitude diverged from within 2% of nominal, the reaction wheels were accelerated in order to reorient the vehicle. This gradual accumulation of angular momentum eventually needs to be "dumped" due to the wheel's limited angular velocity. Although this maximum value was not reached during the 5 periods we modeled, momenting dumping can be achieved through the use of the previously mentioned cold-gas thrusters.

Three RW1 reaction wheels by Blue Canyon Technologies were chosen keep the spacracecraft sensor pointing nadir. These wheels were chosen based on angular momentum each wheel would need to counteract in order to achieve a desired attitude of the spacecraft in order to maintain a nadir pointing orientation. Based on simulated reaction wheel angular momentum plots, it was determined that the maximum amount of angular momentum experience about any axis was approximately .85 Nms and the maximum absolute angular momentum accumulated by all wheels was approximately 1.4 Nms. Each wheel is capable of providing 1.5Nms of angular momentum, making them sufficient to maneuver this spacecraft. See link below for reaction wheel data specifications.

http://bluecanyontech.com/wp-content/uploads/2018/01/DataSheet_RW_07_F.pdf

Table of Contents

421 Final Project

Henry Macanas Eric Ekstrom Michael Johnston

Body parameters

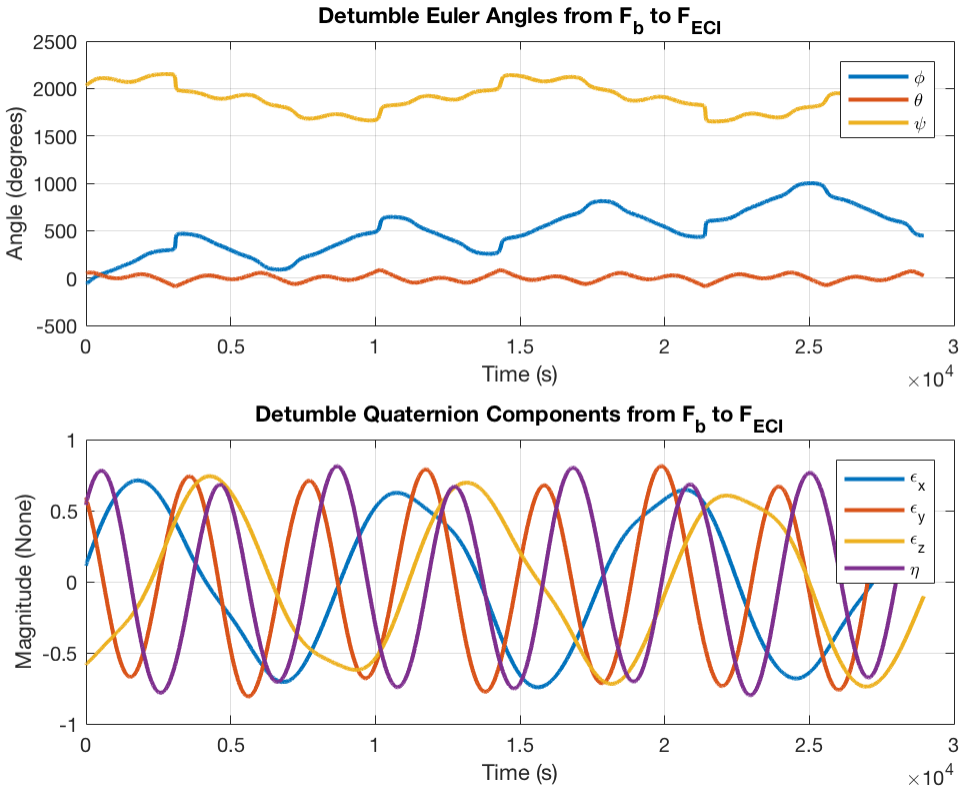
-- Initial COM & I calcs

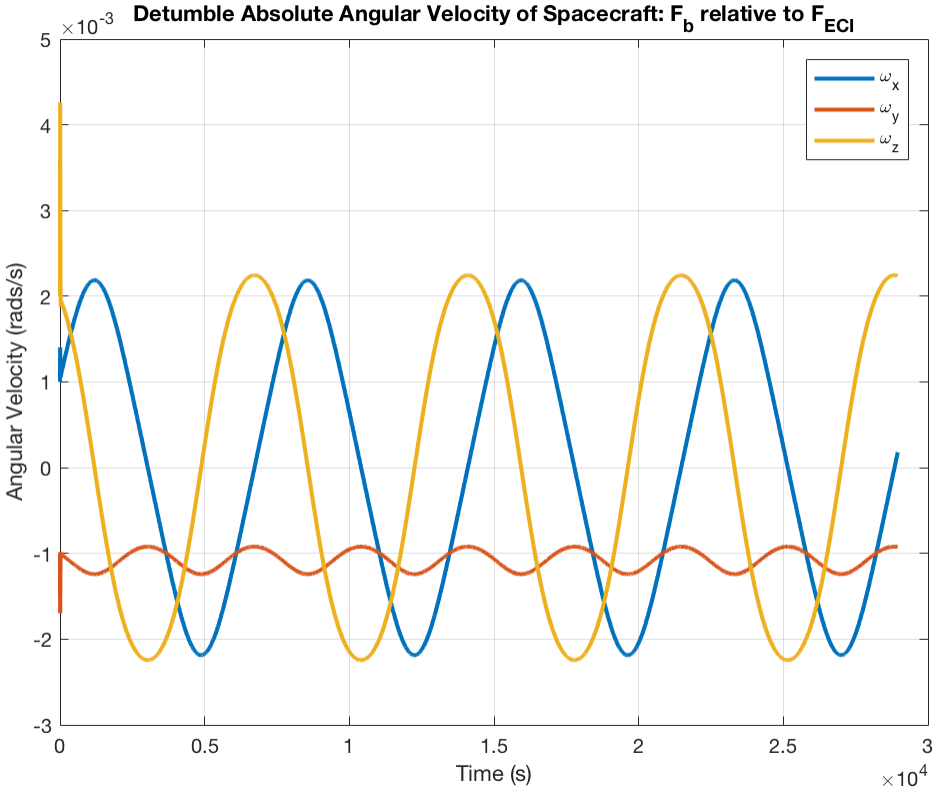
Orbital parameters

rigid body prop

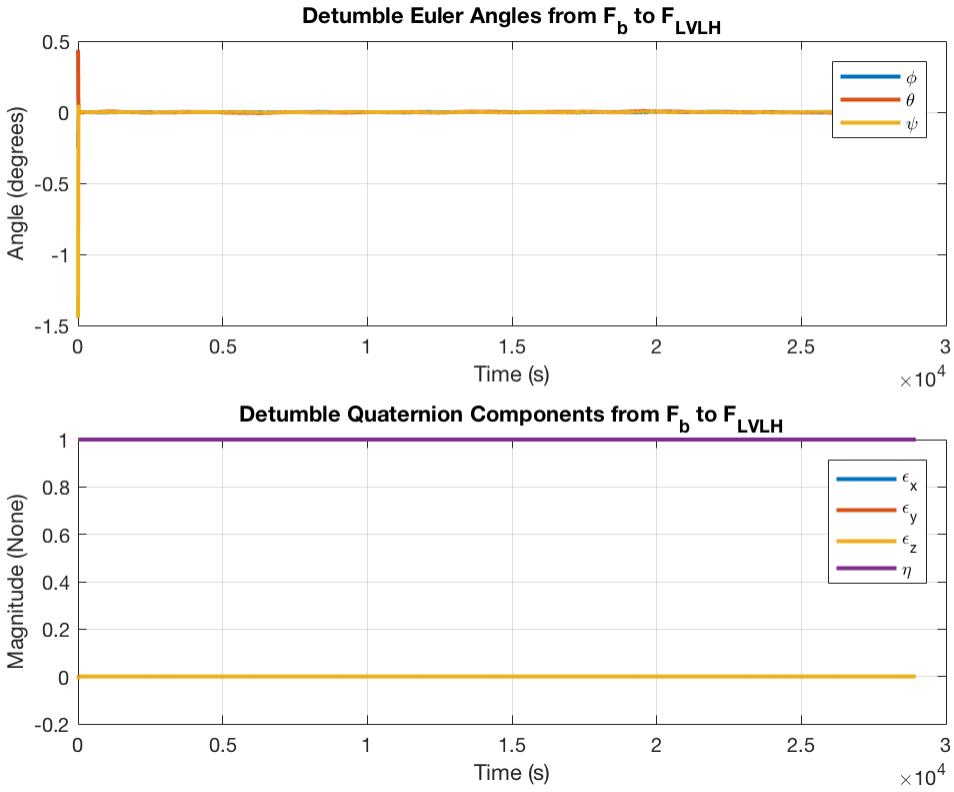
lvlh to eci tranformation

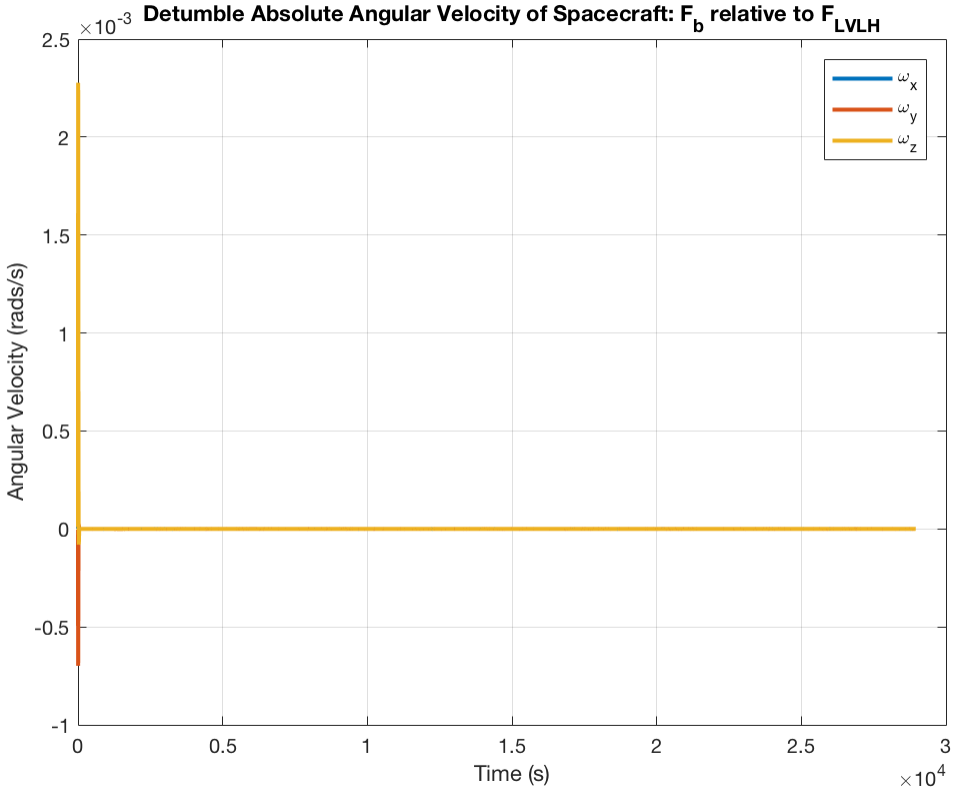
Body Relativeto ECI Plots




Detumble Absolute Angular Velocity of Spacecraft: F_b relative to F_ECI
×10^-3
5
4
3
2
1
0
-1
-2
-3
Angular Velocity (rads/s)
0
0.5
1
1.5
2
2.5
3
Time (s)
×10^4
ω_x
ω_y
ω_z

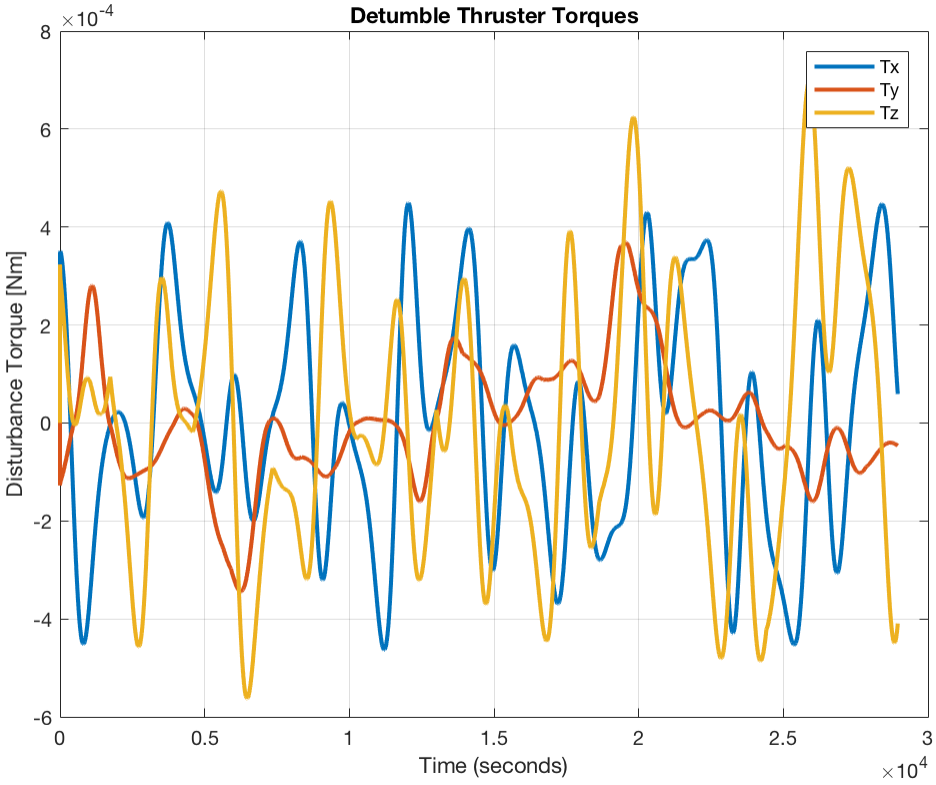
Body Relative to LVLH Plots



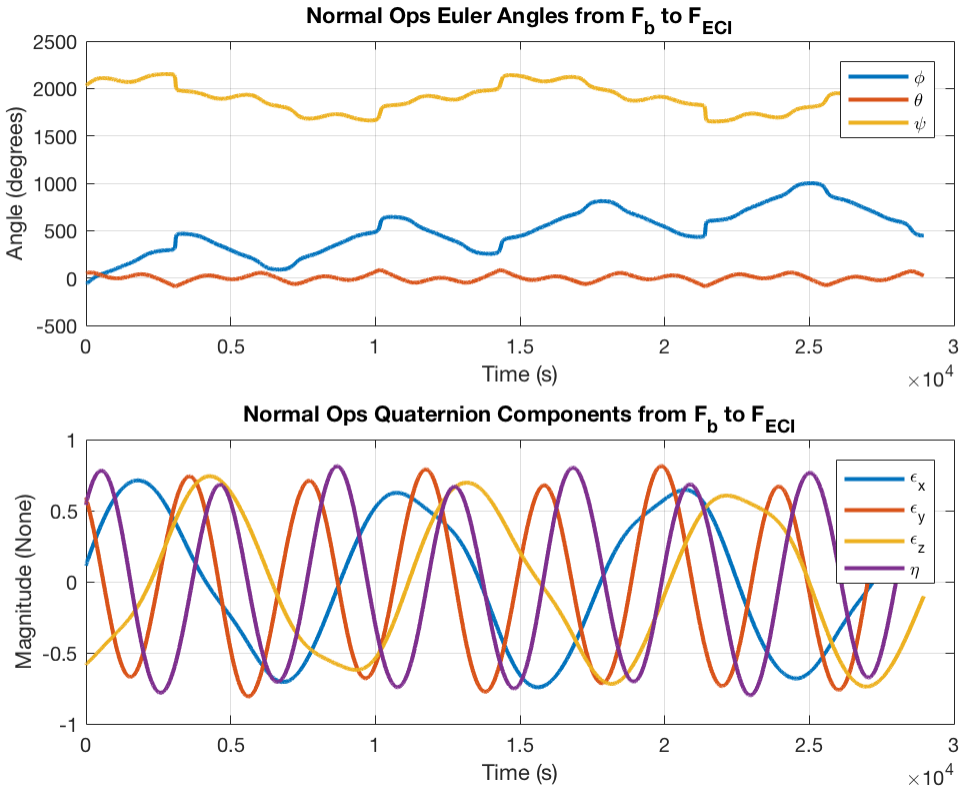


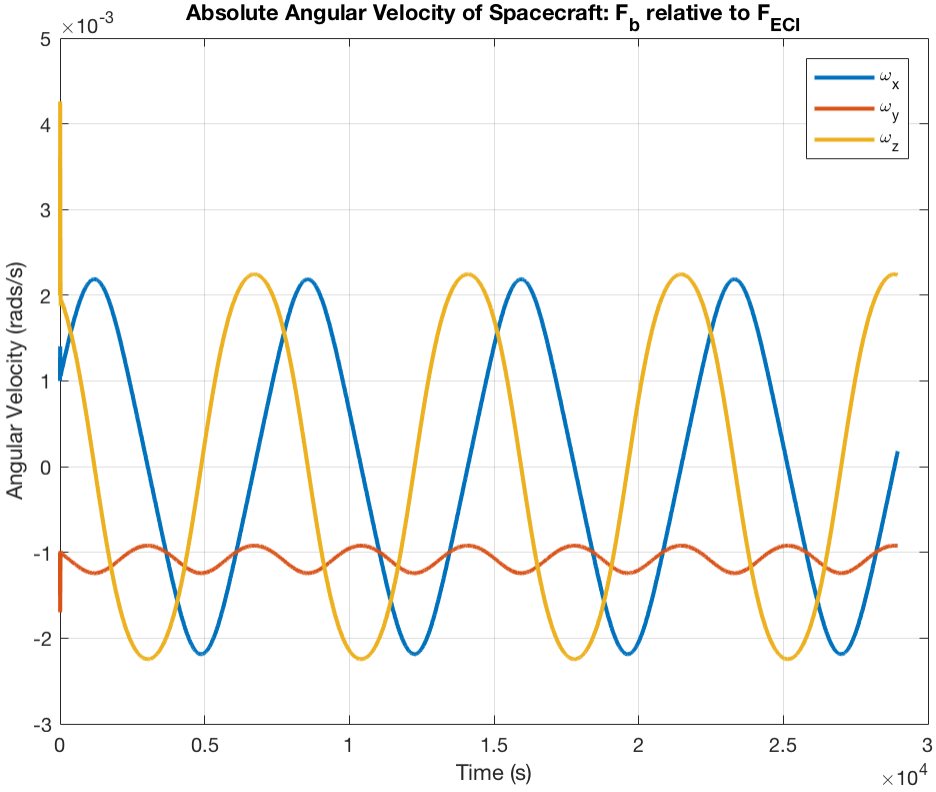
Total Torque

The amount of mass each thruster uses is 0.009404, 0.004387, and 0.010769 kg.
The total amount of mass used by the thrusters is 0.024560 kg.

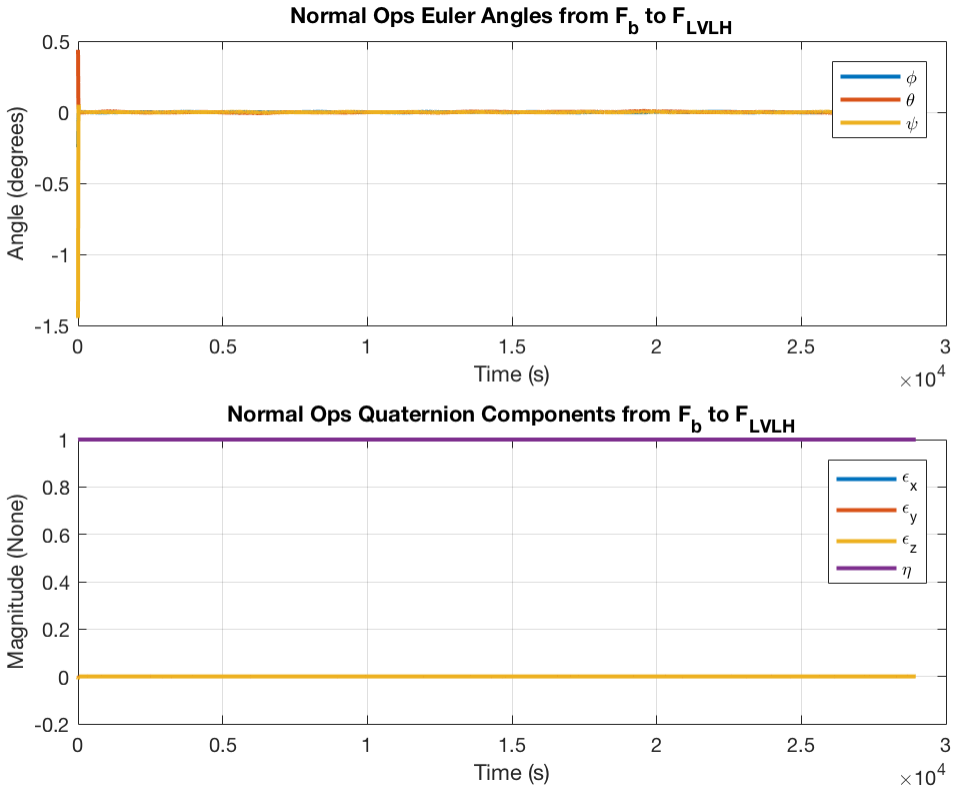

Detumble Thruster Torques
×10^-4
8
6
4
2
0
-2
-4
-6
Disturbance Torque [Nm]
0
0.5
1
1.5
2
2.5
3
×10^4
Time (seconds)
Tx
Ty
Tz

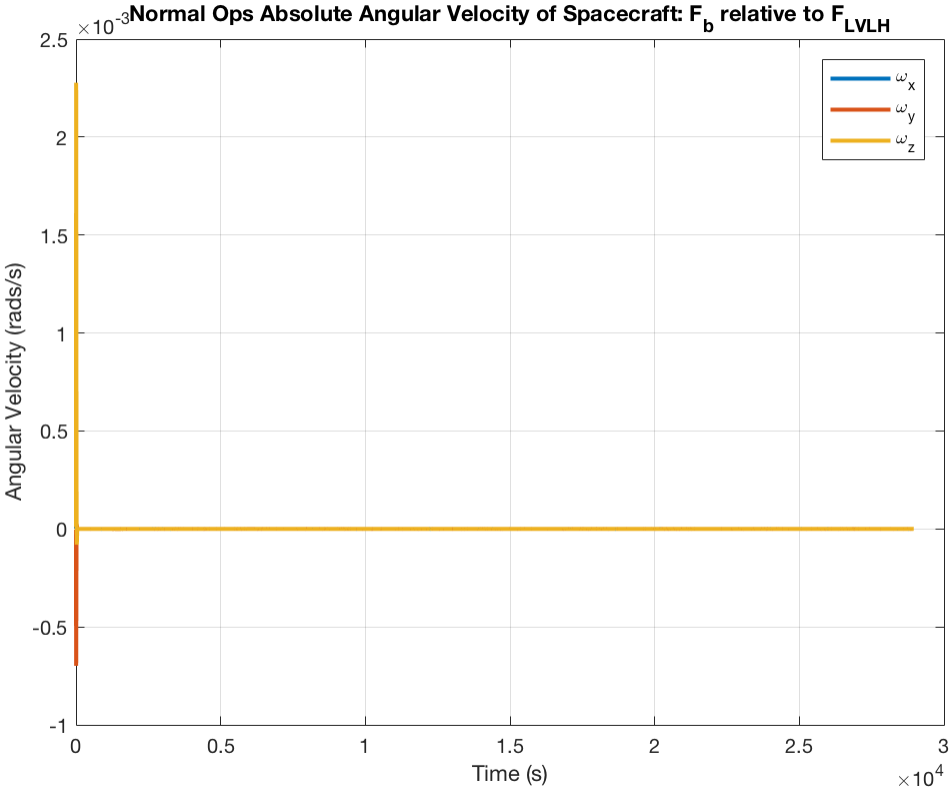
Body Relativeto ECI Plots

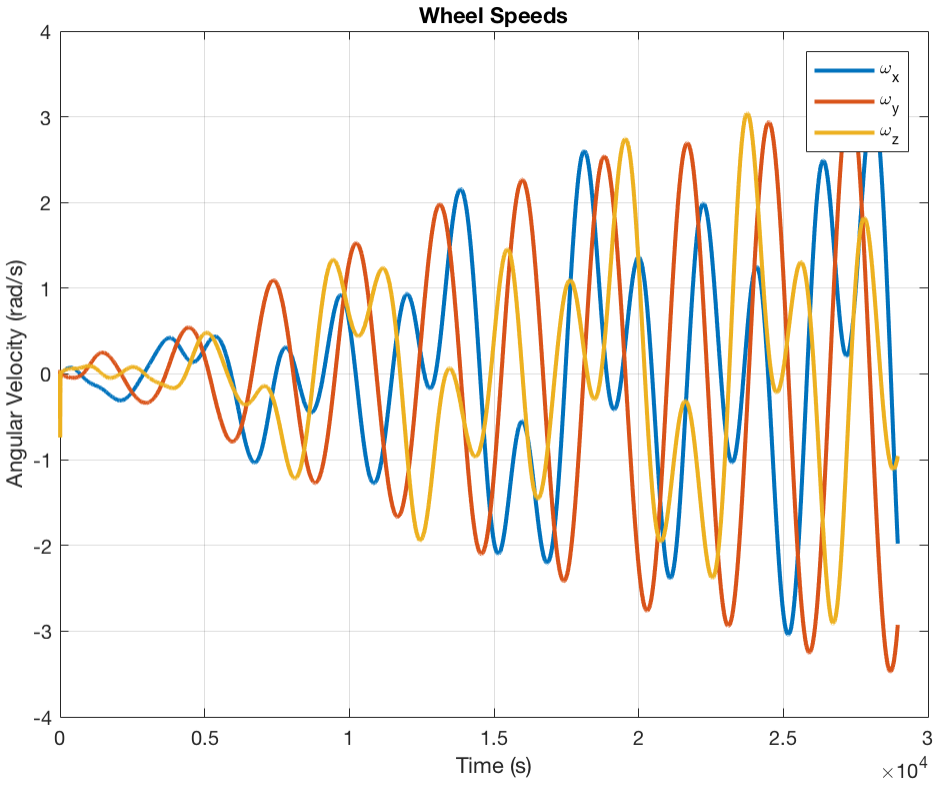



Absolute Angular Velocity of Spacecraft: F_b relative to F_ECI
×10^-3
5
4
3
2
1
0
-1
-2
-3
Angular Velocity (rads/s)
0
0.5
1
1.5
2
2.5
3
×10^4
Time (s)
ω_x
ω_y
ω_z

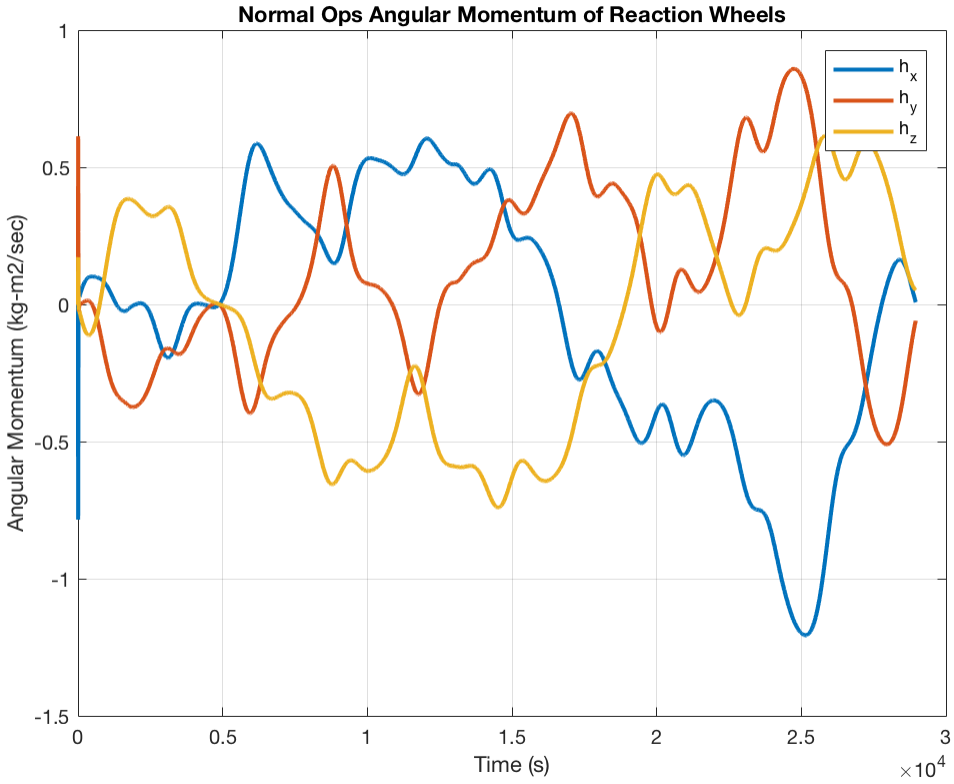
Body Relative to LVLH Plots

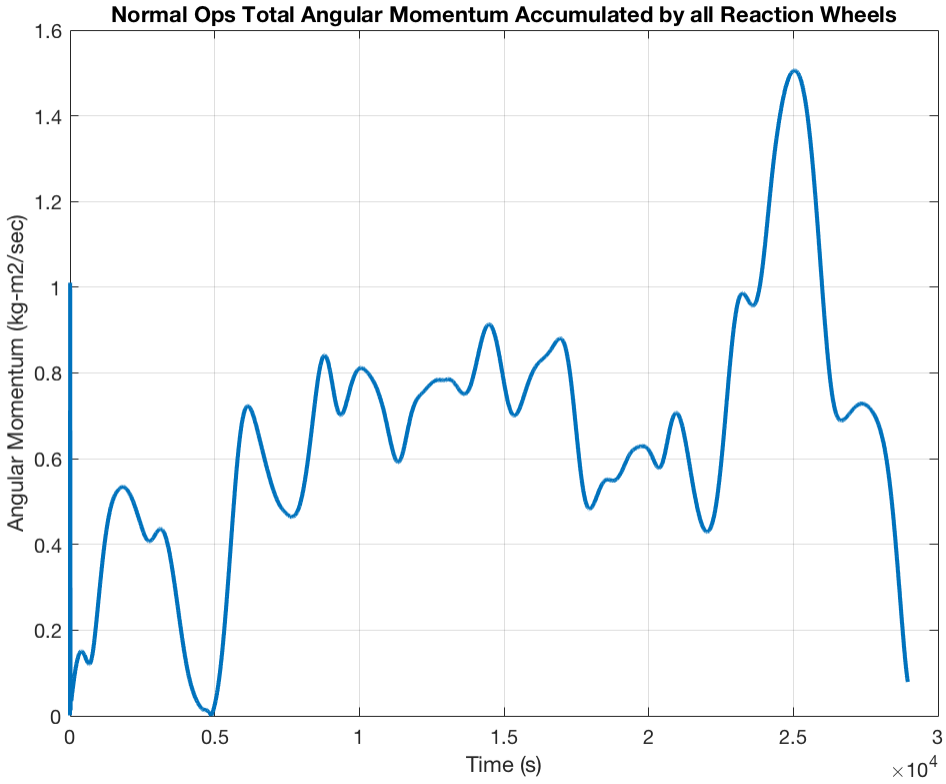



Normal Ops Absolute Angular Velocity of Spacecraft: F_b relative to F_LVLH
×10^-3
2.5
2
1.5
1
0.5
0
-0.5
-1
Angular Velocity (rads/s)
ω_x
ω_y
ω_z
0
0.5
1
1.5
2
2.5
3
Time (s)
×10^4

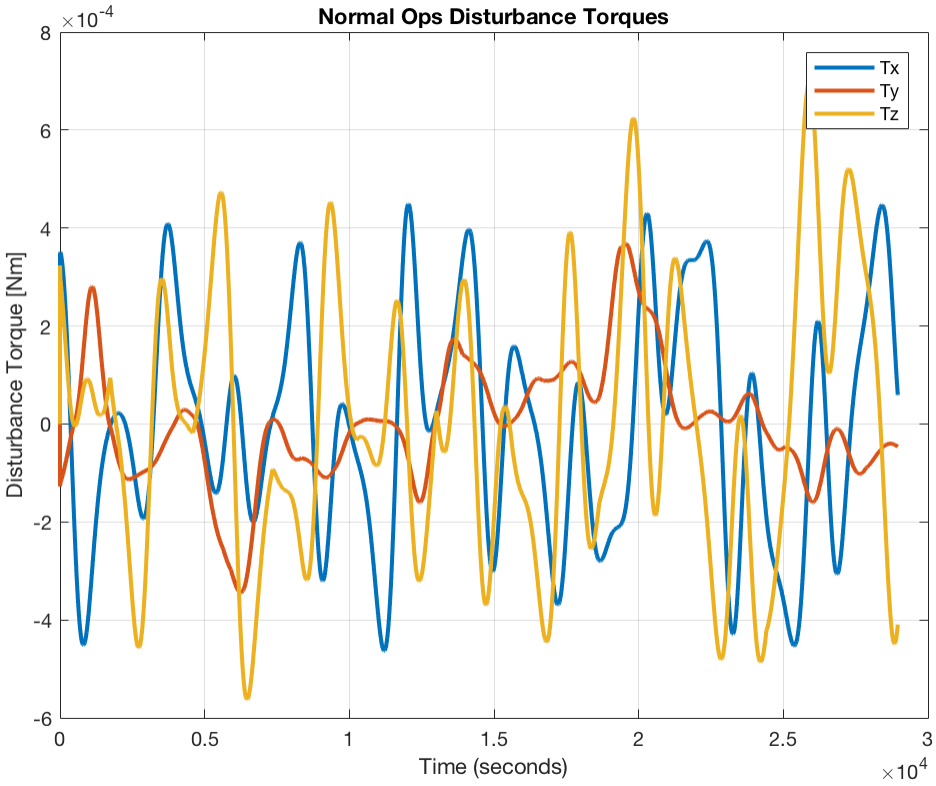

Wheel Speeds
4
3
2
1
0
-1
-2
-3
-4
Angular Velocity (rad/s)
ω_x
ω_y
ω_z
0
0.5
1
1.5
2
2.5
3
Time (s)
×10^4

Angular Momentum Accumulated

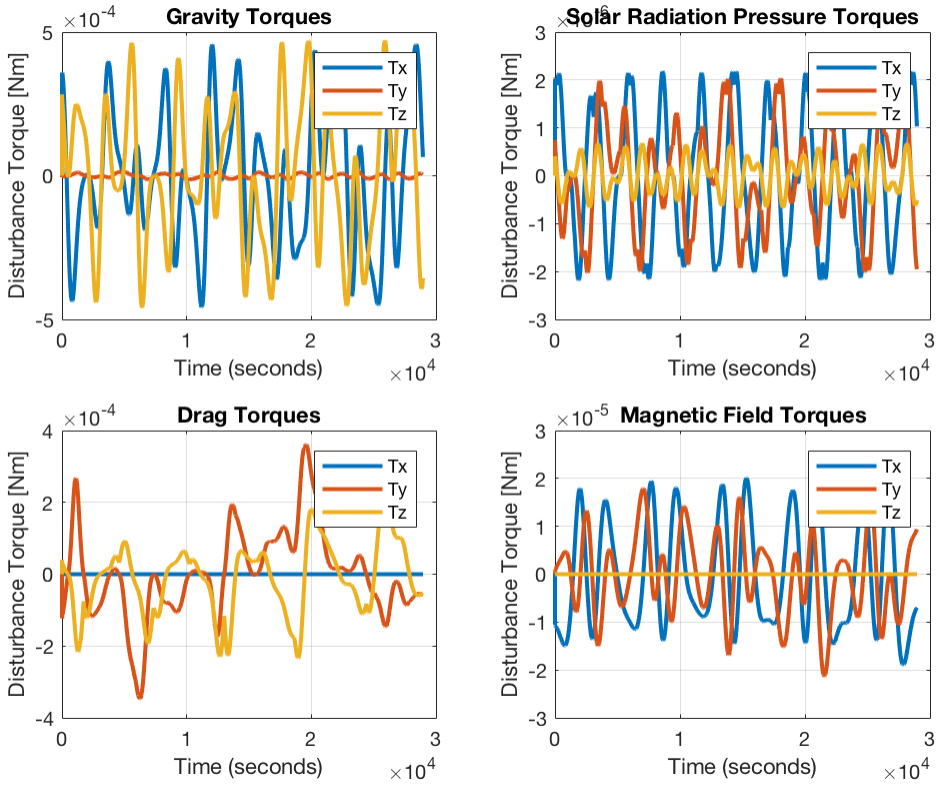



Normal Ops Total Angular Momentum Accumulated by all Reaction Wheels
Angular Momentum (kg-m2/sec)
Time (s)
×10^4
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
0
0.5
1
1.5
2
2.5
3

Total Torque



Individual Torques



Published with MATLAB® R2016b

Table of Contents

421 Final Project

Henry Macanas Eric Ekstrom Michael Johnston

```
clc, close all, clear all

addpath('overhead_functions/')
```

Body parameters

-- Initial COM & I calcs

```
[COM0,I0] = getCOM(100.75,zeros(3,1),[2;2;2],['s']); % COM & I for
 initial configuration

dimRxn = [0.075,0.065,0]; % Reaction wheel dimensions (r,h,~) [m]
mRxn = 1.6; % Mass of reaction wheels [kg]
[~,Iw] = getCOM(mRxn, zeros(3,1), dimRxn', 'disk');
Iw = Iw(1,1)*eye(3); % Inertia matrix of all three wheels

% -- Deployed COM & I calcs
Rbb = zeros(3,1); % Distance from bus COM to bus COM
Rbsp = [0; 2.5; 0]; % Distance from bus COM to solar panel COM
Rbsens = [0; 0; 1.5]; % Distance from bus COM to sensor COM
masses = [50,20,20,10,mRxn*ones(1,3)]; % Component masses [kg]
dims = [2,2,2; 2,3,0.05; 2,3,0.05; 0.25,0.25,1;
 dimRxn;dimRxn;dimRxn]'; % Component dimensions
shapes = ['';'';'';'disk';'disk';'disk']; % Shapes of components
[COM, I] = getCOM(masses,...
 [Rbb Rbsp -Rbsp Rbsens],...
 dims,shapes); % Spacecraft center of mass

consts.I = I0;
consts.COM = COM0;
```

```
% inertial parameters
consts.n   = [1 0 -1  0 0  0    1 1 -1 -1 0 0  0  0    1 -1 0  0;
              0 1  0 -1 0  0    0 0  0  0 0 0  0  0    0  0 1 -1;
              0 0  0  0 1 -1    0 0  0  0 1 1 -1 -1    0  0 0  0]; %
 Bus    Solar Panels    Sensor
consts.rho = [1 0 -1  0 0  0      1    1   -1   -1    0    0     0
 0    .125 -.125    0     0;
              0 1  0 -1 0  0    2.5 -2.5  2.5 -2.5  2.5 -2.5   2.5
 -2.5       0     0 .125 -.125;
              0 0  0  0 1 -1      0    0    0    0 .025 .025 -.025
 -.025     1.5   1.5  1.5   1.5] - consts.COM; % Bus    Solar Panels
  Sensor
consts.A   = [4 4 4 4 4 4    .15 .15 .15 .15 6 6 6 6
  .25 .25 .25 .25]; % Bus    Solar Panels    Sensor
```

Orbital parameters

```
COEs.a = 6370 + 600; % 600km altitude
COEs.ecc = 0;
COEs.inc = deg2rad(40); % 40deg inc
COEs.RAAN = 0;
COEs.arg = 0;
COEs.TA = 0;

% Initial R and V vectors
[R,V] = COE2RV(COEs,398600);
% retrieves period
[ ~,~,~,~,~,~,P,~ ] = coes(R,V);

% angular velcoity of lvlh frame
w_lvlh_eci_eci = cross(R,V)/norm(R)^2; % this is wrong, should be a
 3x1 not 1x1
```

rigid body prop

lvlh to eci tranformation

```
C_lvlh_eci= eci2lvlh(R,V);
C_body_lvlh = eye(3);
C_body_eci = C_body_lvlh*C_lvlh_eci;

% initial states inn reference to eci
w0_body_eci = [0.5;-2.0;3.0]*10^(-3);
r0_eci_eci = R;
v0_eci_eci = V;
euler_angles0_eci = euler_angs(C_body_eci);
q0_eci_eci = quaternion(C_body_eci);

% states relative to lvlh
euler_angs0_lvlh = [0;0;0];
w_body_eci = w0_body_eci;
```

```
w_body_lvlh0 = w_body_eci - C_body_eci*w_lvlh_eci_eci;
q0_body_lvlh = [0;0;0;1];
w0_wheel = [0;0;0];
w0_wheel_lvlh = w_body_lvlh0 + C_body_lvlh'*w0_wheel;

% State vector
state =
 [euler_angles0_eci;w0_body_eci;q0_eci_eci;r0_eci_eci;v0_eci_eci;...

 euler_angs0_lvlh;w_body_lvlh0;q0_body_lvlh;w0_wheel;w0_wheel_lvlh];

% gains
Ts = 3*P; % [s]
zeta = 0.7292;
wn = 4.4/(Ts*zeta);
kd = 2*consts.I*zeta*wn;
kp = 2*consts.I*wn^2;
%--ode call

mission = 'detumble';
tspan = [0 5*P];
options =
 odeset('RelTol',1e-8,'AbsTol',1e-8, 'OutputFcn',@(t,y,flag,varargin)
 odeOutFunc(t,y,flag));
% [tnew, statenew] =
 ode45(@normOpsOde,tspan,state,options,mission,consts,kd,kp,Iw);
% Save and load solutions for speed
% save('soln','tnew','statenew', 'Torques')
load('soln')
```

Body Relativeto ECI Plots

```
figure
subplot(2,1,1)
plot(tnew,rad2deg(statenew(:,1:3)),'LineWidth',2)
title('Detumble Euler Angles from F_b to F_{ECI}')
xlabel('Time (s)')
ylabel('Angle (degrees)')
legend('\phi','\theta','\psi')

subplot(2,1,2)
plot(tnew,statenew(:,7:10),'LineWidth',2)
title('Detumble Quaternion Components from F_b to F_{ECI}')
xlabel('Time (s)')
ylabel('Magnitude (None)')
legend('\epsilon_x','\epsilon_y','\epsilon_z','\eta')

figure
set(groot,'DefaultAxesXGrid','on', 'DefaultAxesYGrid','on')
plot(tnew,statenew(:,4:6),'LineWidth',2)
title('Detumble Absolute Angular Velocity of Spacecraft: F_b relative
 to F_{ECI}')
xlabel('Time (s)')
```

```
ylabel('Angular Velocity (rads/s)')
legend('\omega_x','\omega_y','\omega_z')
```

Body Relative to LVLH Plots

```
figure
subplot(2,1,1)
plot(tnew,rad2deg(statenew(:,17:19)),'LineWidth',2)
title('Detumble Euler Angles from F_b to F_{LVLH}')
xlabel('Time (s)')
ylabel('Angle (degrees)')
legend('\phi','\theta','\psi')

subplot(2,1,2)
plot(tnew,statenew(:,23:26),'LineWidth',2)
title('Detumble Quaternion Components from F_b to F_{LVLH}')
xlabel('Time (s)')
ylabel('Magnitude (None)')
legend('\epsilon_x','\epsilon_y','\epsilon_z','\eta')

figure
plot(tnew,statenew(:,20:22),'LineWidth',2)
title('Detumble Absolute Angular Velocity of Spacecraft: F_b relative
 to F_{LVLH}')
xlabel('Time (s)')
ylabel('Angular Velocity (rads/s)')
legend('\omega_x','\omega_y','\omega_z')
```

Total Torque

```
figure
plot(Torques.tot(:,1),Torques.tot(:,2:4), 'lineWidth', 2)
grid on
title('Detumble Thruster Torques')
xlabel('Time (seconds)'), ylabel('Disturbance Torque [Nm]')
legend('Tx', 'Ty', 'Tz')

m = propMass(tnew,Torques.tot); % [kg] for each thruster
massTotal = sum(m); % [kg] total

fprintf('The amount of mass each thruster uses is %f, %f, and %f kg.
 \n The total amount of mass used by the thrusters is %f kg. \n',
[m(1),m(2),m(3),massTotal])
fprintf('\n')
%--ode call

% gains
Ts = 30; % [s]
consts.I = I; % Update inertia matrix
consts.COM = COM; % Update center of mass
wn = 4.4/(Ts*zeta);
kd = 2*consts.I*zeta*wn;
```

```
kp = 2*consts.I*wn^2;
state = statenew(end,:); % Update state to last state of detumble
state(4:6) = [0.001; -0.001; 0.002]; % Update omega

mission = 'normops';
tspan = [0 5*P];
options =
 odeset('RelTol',1e-6,'AbsTol',1e-6, 'OutputFcn',@(t,y,flag,varargin)
 odeOutFunc(t,y,flag));
% [tnew, statenew] =
 ode45(@normOpsOde,tspan,state,options,mission,consts,kd,kp,Iw);
% Save and load solutions for speed
% save('soln','tnew','statenew', 'Torques')
% load('soln')

h = zeros(length(statenew),3);
mag_h = zeros(length(statenew),1);
for i = 1:length(statenew)
 h(i,:) = cross(diag(consts.I,0), statenew(i,4:6));
    mag_h(i,:) = norm(h(i,:));
end

hW = zeros(length(statenew),3);
for i = 1:length(statenew)
 hW(i,:) = cross(diag(Iw), statenew(i,30:32));
    mag_hW(i,:) = norm(hW(i,:));
end
```

Body Relativeto ECI Plots

```
figure
subplot(2,1,1)
plot(tnew,rad2deg(statenew(:,1:3)),'LineWidth',2)
title('Normal Ops Euler Angles from F_b to F_{ECI}')
xlabel('Time (s)')
ylabel('Angle (degrees)')
legend('\phi','\theta','\psi')

subplot(2,1,2)
plot(tnew,statenew(:,7:10),'LineWidth',2)
title('Normal Ops Quaternion Components from F_b to F_{ECI}')
xlabel('Time (s)')
ylabel('Magnitude (None)')
legend('\epsilon_x','\epsilon_y','\epsilon_z','\eta')

figure
set(groot,'DefaultAxesXGrid','on', 'DefaultAxesYGrid','on')
plot(tnew,statenew(:,4:6),'LineWidth',2)
title('Absolute Angular Velocity of Spacecraft: F_b relative to
 F_{ECI}')
xlabel('Time (s)')
ylabel('Angular Velocity (rads/s)')
legend('\omega_x','\omega_y','\omega_z')
```

Body Relative to LVLH Plots

```
figure
subplot(2,1,1)
plot(tnew,rad2deg(statenew(:,17:19)),'LineWidth',2)
title('Normal Ops Euler Angles from F_b to F_{LVLH}')
xlabel('Time (s)')
ylabel('Angle (degrees)')
legend('\phi','\theta','\psi')

subplot(2,1,2)
plot(tnew,statenew(:,23:26),'LineWidth',2)
title('Normal Ops Quaternion Components from F_b to F_{LVLH}')
xlabel('Time (s)')
ylabel('Magnitude (None)')
legend('\epsilon_x','\epsilon_y','\epsilon_z','\eta')

figure
plot(tnew,statenew(:,20:22),'LineWidth',2)
title('Normal Ops Absolute Angular Velocity of Spacecraft: F_b
 relative to F_{LVLH}')
xlabel('Time (s)')
ylabel('Angular Velocity (rads/s)')
legend('\omega_x','\omega_y','\omega_z')

% Wheel speeds
figure
plot(tnew, statenew(:,27:29), 'lineWidth', 2)
grid on
title('Wheel Speeds')
xlabel('Time (s)'), ylabel('Angular Velocity (rad/s)')
legend('\omega_x','\omega_y','\omega_z')
```

Angular Momentum Accumulated

```
figure
plot(tnew, hW, 'lineWidth', 2)
grid on
title('Normal Ops Angular Momentum of Reaction Wheels')
xlabel('Time (s)'), ylabel('Angular Momentum (kg-m2/sec)')
legend('h_x','h_y','h_z')

figure
plot(tnew, mag_hW, 'lineWidth', 2)
grid on
title('Normal Ops Total Angular Momentum Accumulated by all Reaction
 Wheels')
xlabel('Time (s)'), ylabel('Angular Momentum (kg-m2/sec)')

% % Wheel speeds
% figure
% plot(tnew, statenew(:,30:32), 'lineWidth', 2)
```

```
% grid on
% title('Normal Ops Wheel Speeds (rel to LVLH)')
% xlabel('Time (s)'), ylabel('Angular Velocity (rad/s)')
% legend('\omega_x','\omega_y','\omega_z')
```

Total Torque

```
figure
plot(Torques.tot(:,1),Torques.tot(:,2:4), 'lineWidth', 2)
grid on
title('Normal Ops Disturbance Torques')
xlabel('Time (seconds)'), ylabel('Disturbance Torque [Nm]')
legend('Tx', 'Ty', 'Tz')
```

Individual Torques

```
figure
subplot(2,2,1)
plot(Torques.grav(:,1),Torques.grav(:,2:4), 'lineWidth', 2)
grid on
title('Gravity Torques')
xlabel('Time (seconds)'), ylabel('Disturbance Torque [Nm]')
legend('Tx', 'Ty', 'Tz')

subplot(2,2,2)
plot(Torques.srp(:,1),Torques.srp(:,2:4), 'lineWidth', 2)
grid on
title('Solar Radiation Pressure Torques')
xlabel('Time (seconds)'), ylabel('Disturbance Torque [Nm]')
legend('Tx', 'Ty', 'Tz')

subplot(2,2,3)
plot(Torques.drag(:,1),Torques.drag(:,2:4), 'lineWidth', 2)
grid on
title('Drag Torques')
xlabel('Time (seconds)'), ylabel('Disturbance Torque [Nm]')
legend('Tx', 'Ty', 'Tz')

subplot(2,2,4)
plot(Torques.mag(:,1),Torques.mag(:,2:4), 'lineWidth', 2)
grid on
title('Magnetic Field Torques')
xlabel('Time (seconds)'), ylabel('Disturbance Torque [Nm]')
legend('Tx', 'Ty', 'Tz')
```

Published with MATLAB® R2016b