Midterm - ASE 389P.8 Satellite Control Systems

Corey Marcus

March 9, 2023

1 Part 1

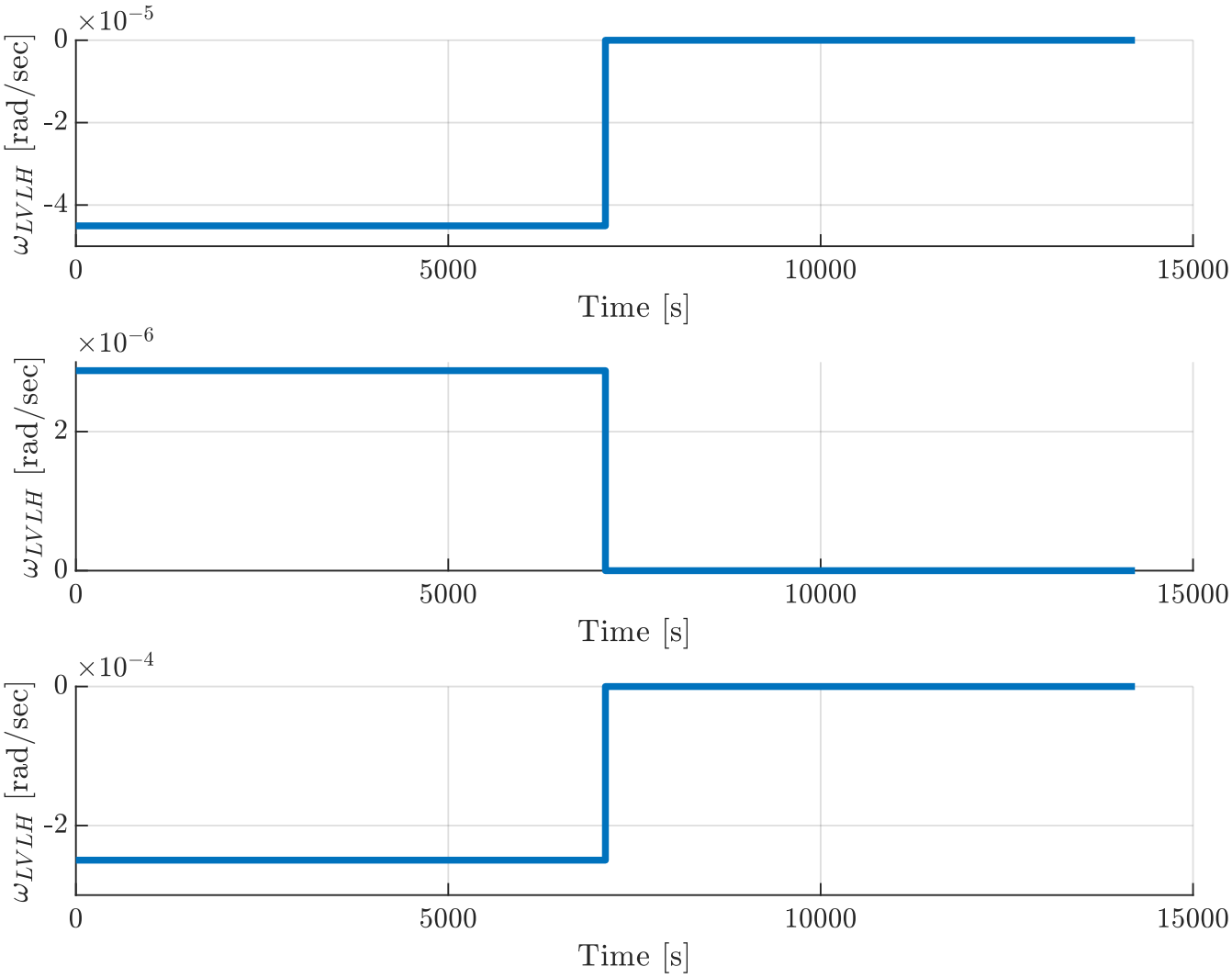


Figure 1: The nominal LVLH to body angular rates. We begin with a constant rate maneuver, followed by a LVLH hold.

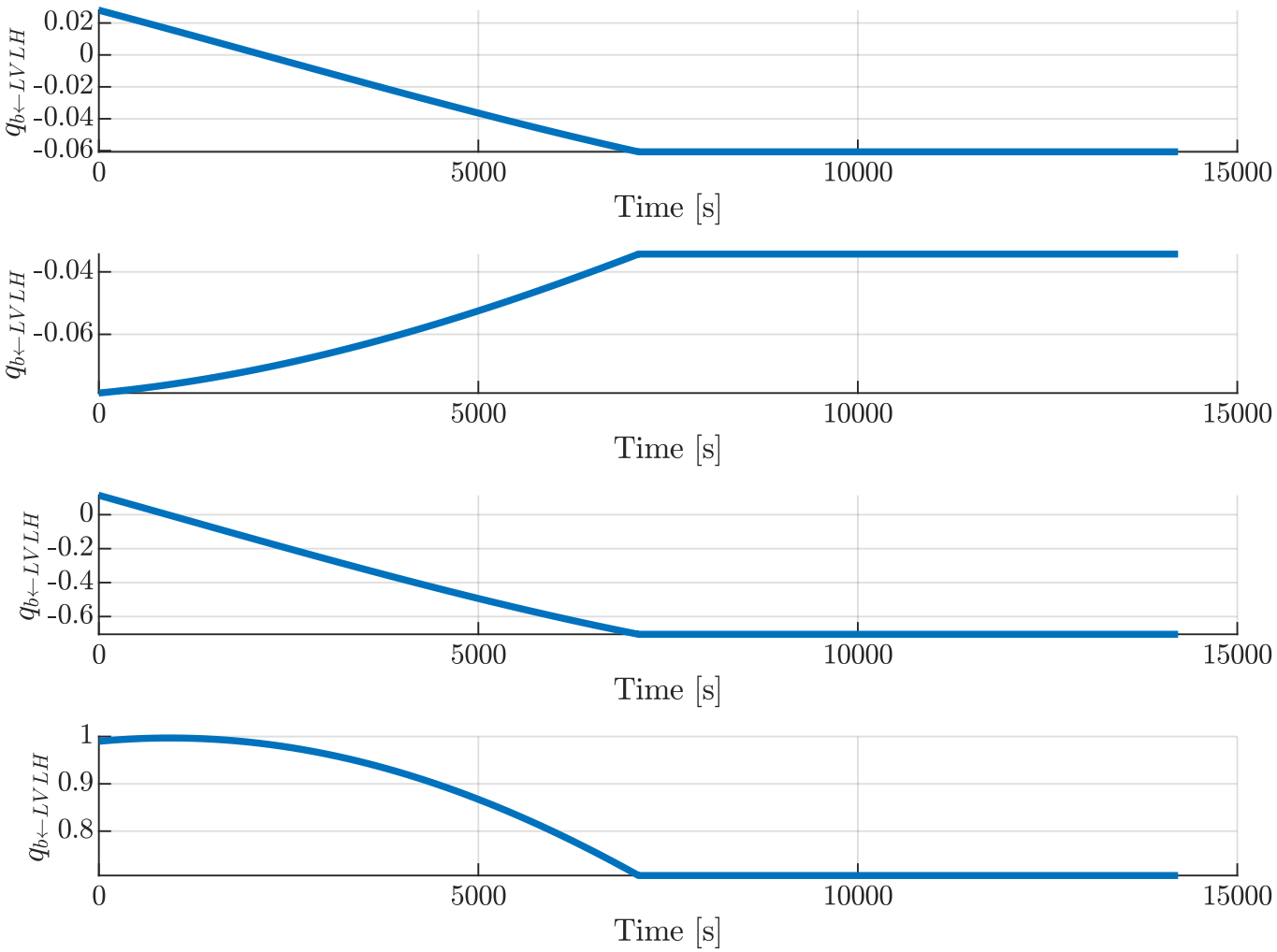


Figure 2: The nominal LVLH to body quaternion. Here we see that the quaternion is commanded to be constant at the end of the maneuver.

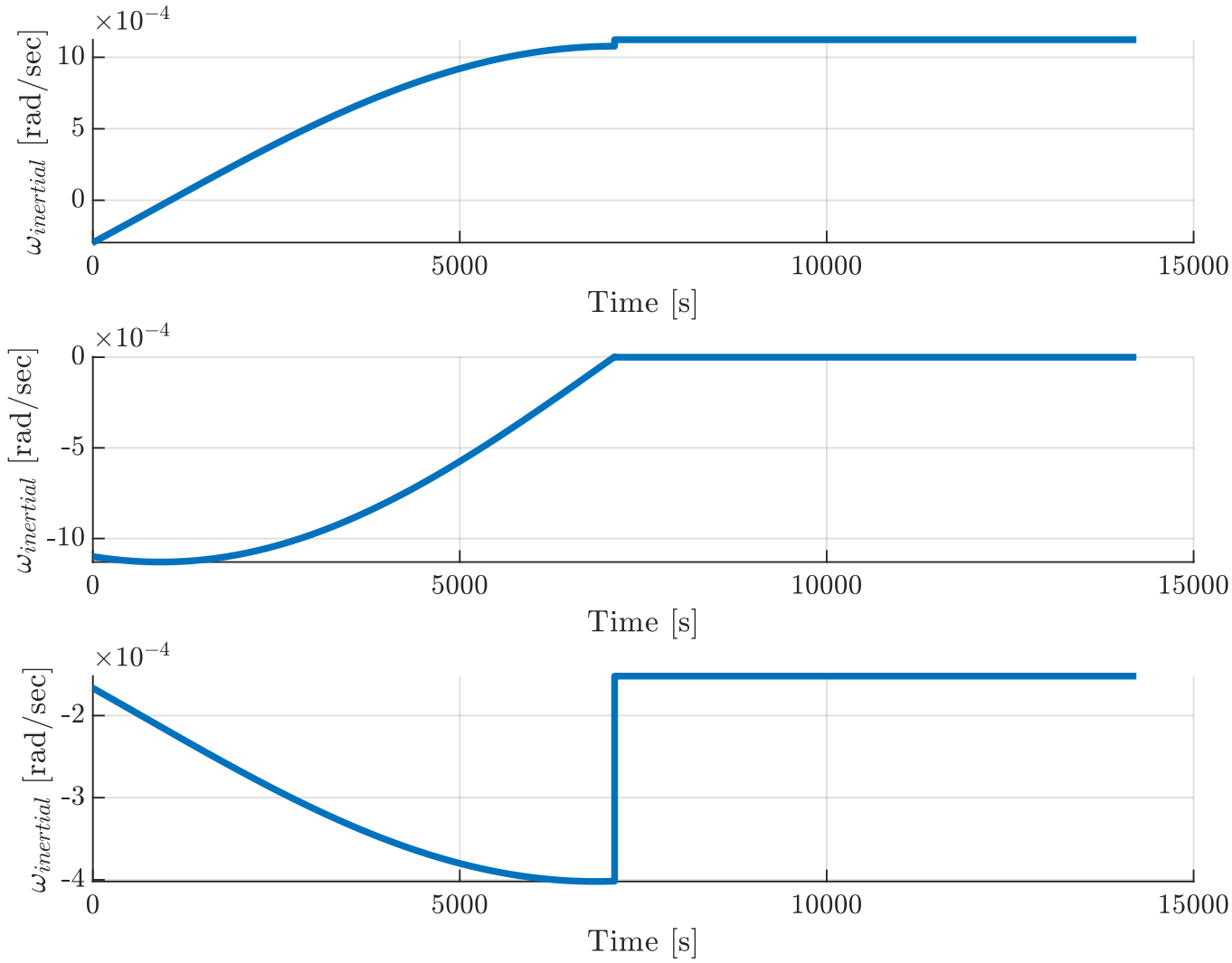


Figure 3: The nominal inertial to body angular rates. These are not zero after the maneuver because the LVLH frame is rotating.

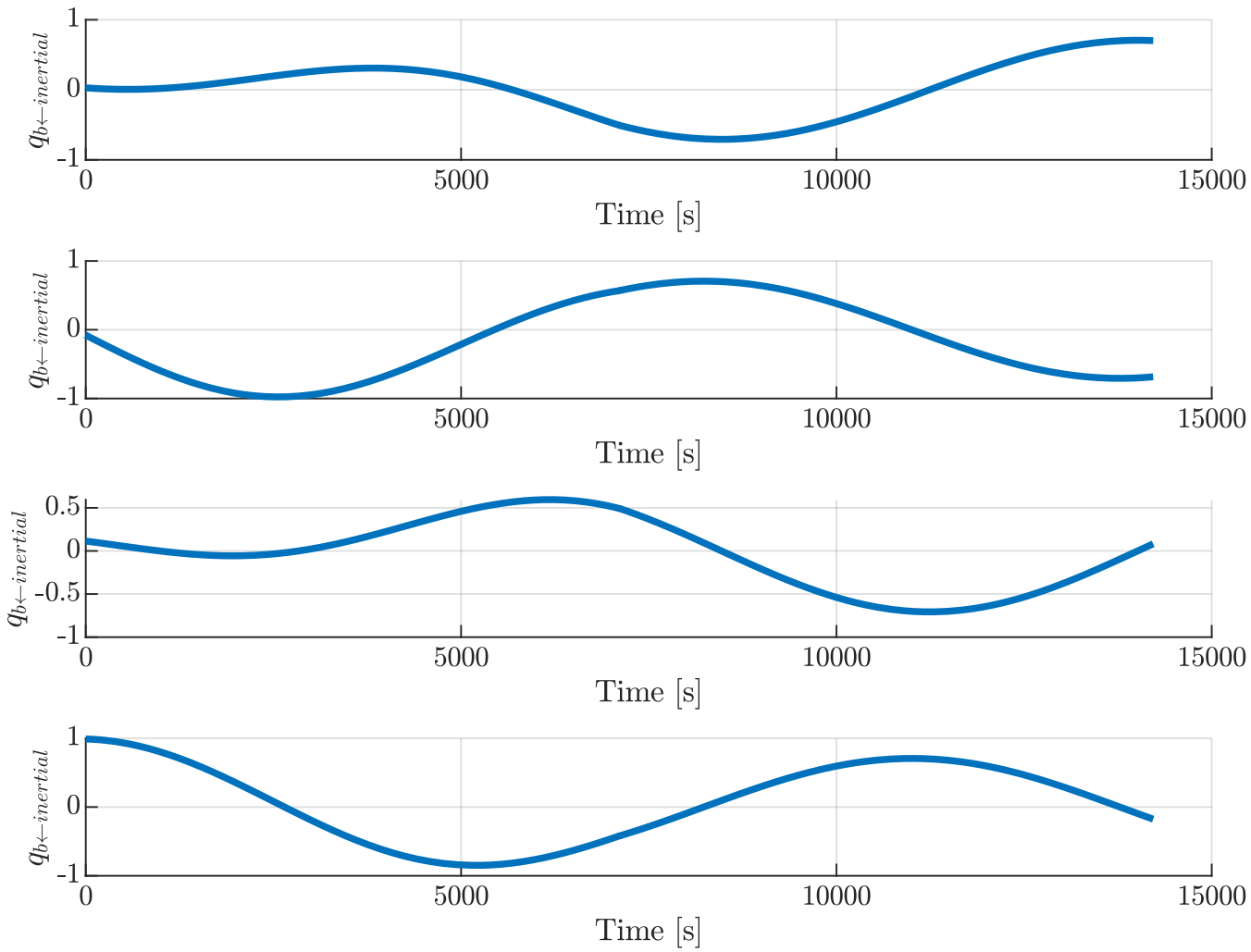


Figure 4: The nominal inertial to body quaternion. This is not constant after the maneuver because the LVLH frame is rotating.

2 Part 2

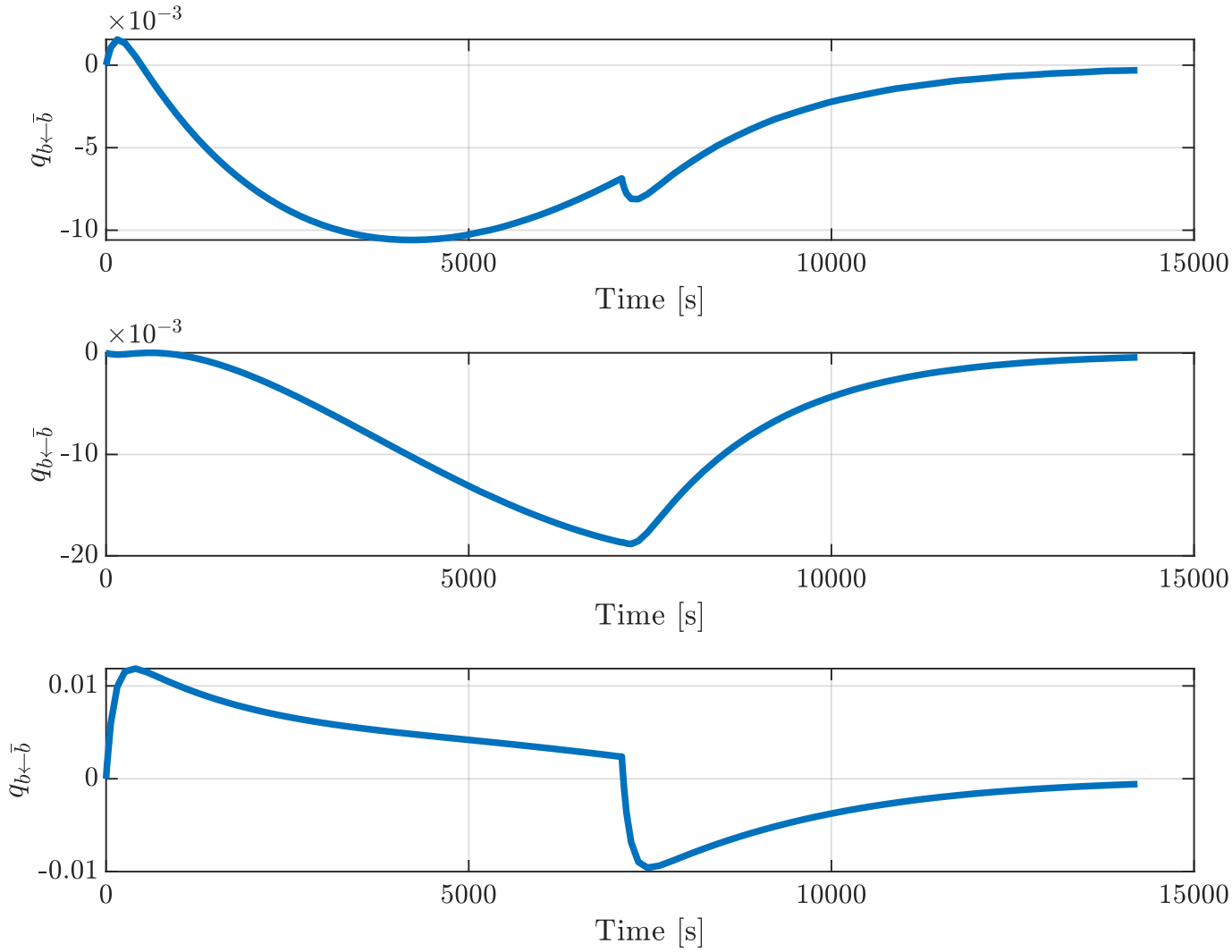


Figure 5: The attitude control error as represented by the vector components of the error quaternion. The commanded body rate profile is discontinuous at the beginning and end of the maneuver, resulting in overshoot and poor tracking performance after those moments.

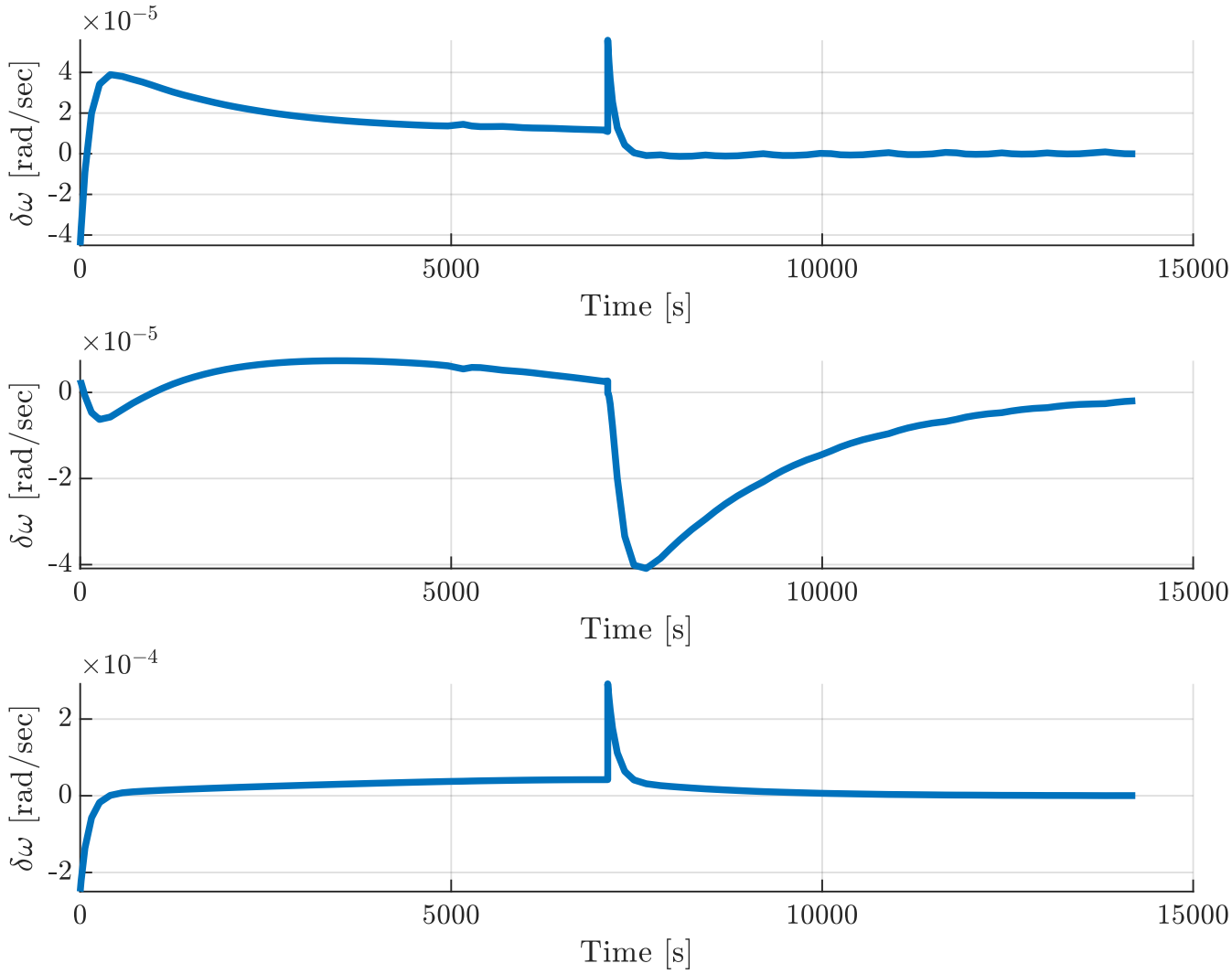


Figure 6: The attitude rate error. The commanded body rate profile is discontinuous at the beginning and end of the maneuver, resulting in overshoot and poor performance after those moments.

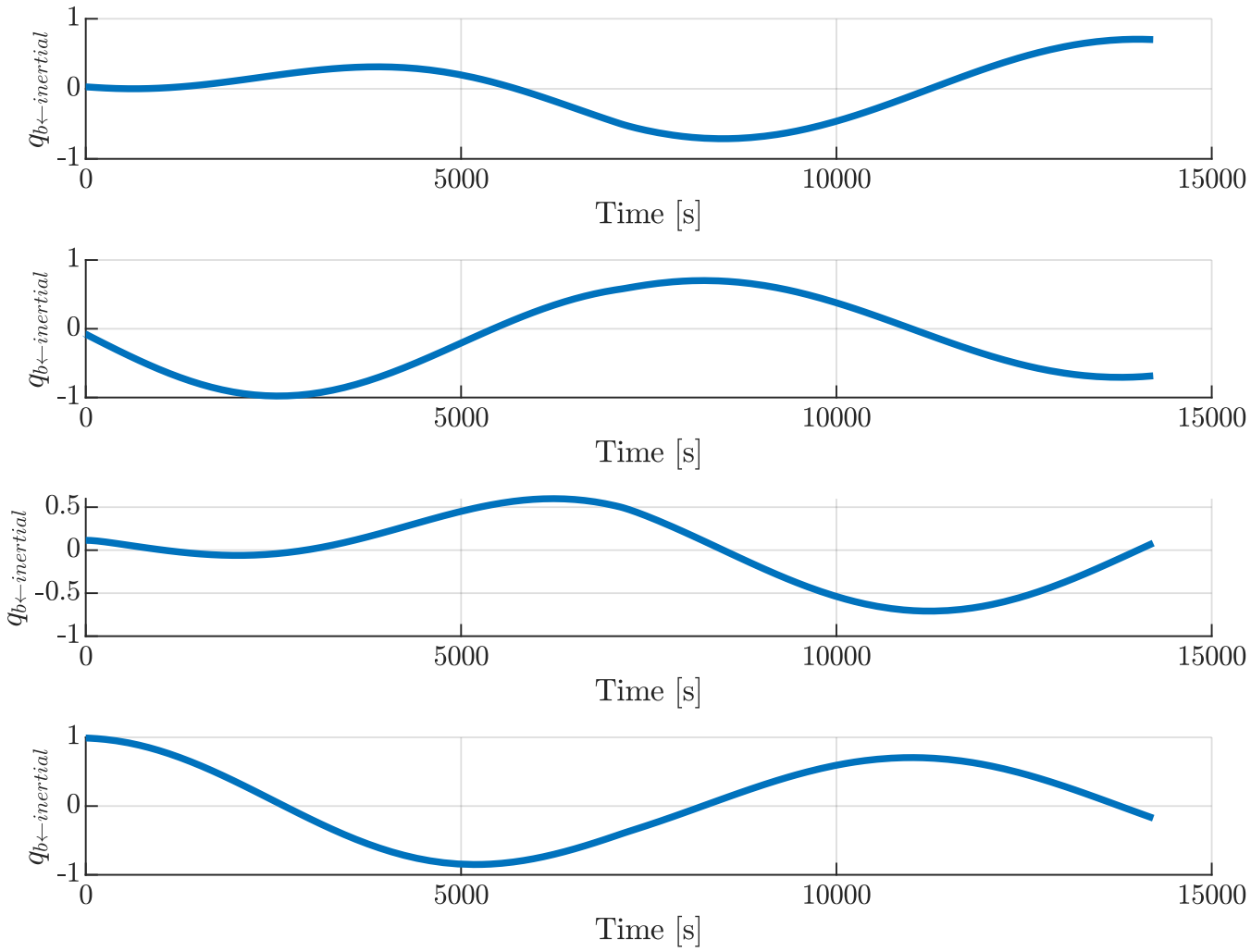


Figure 7: The true inertial to body quaternion.

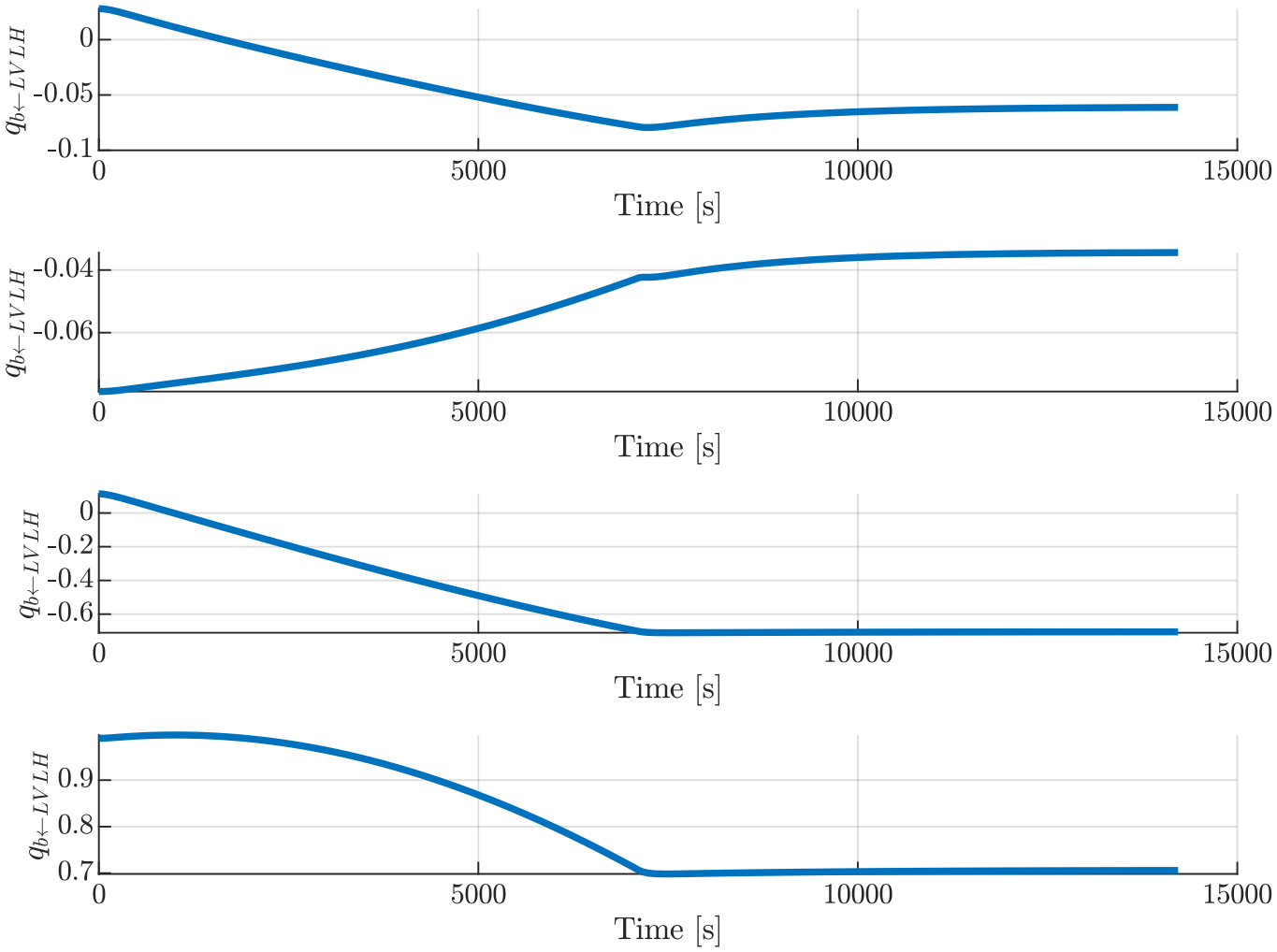


Figure 8: The true LVLH to body quaternion.

3 Part 3

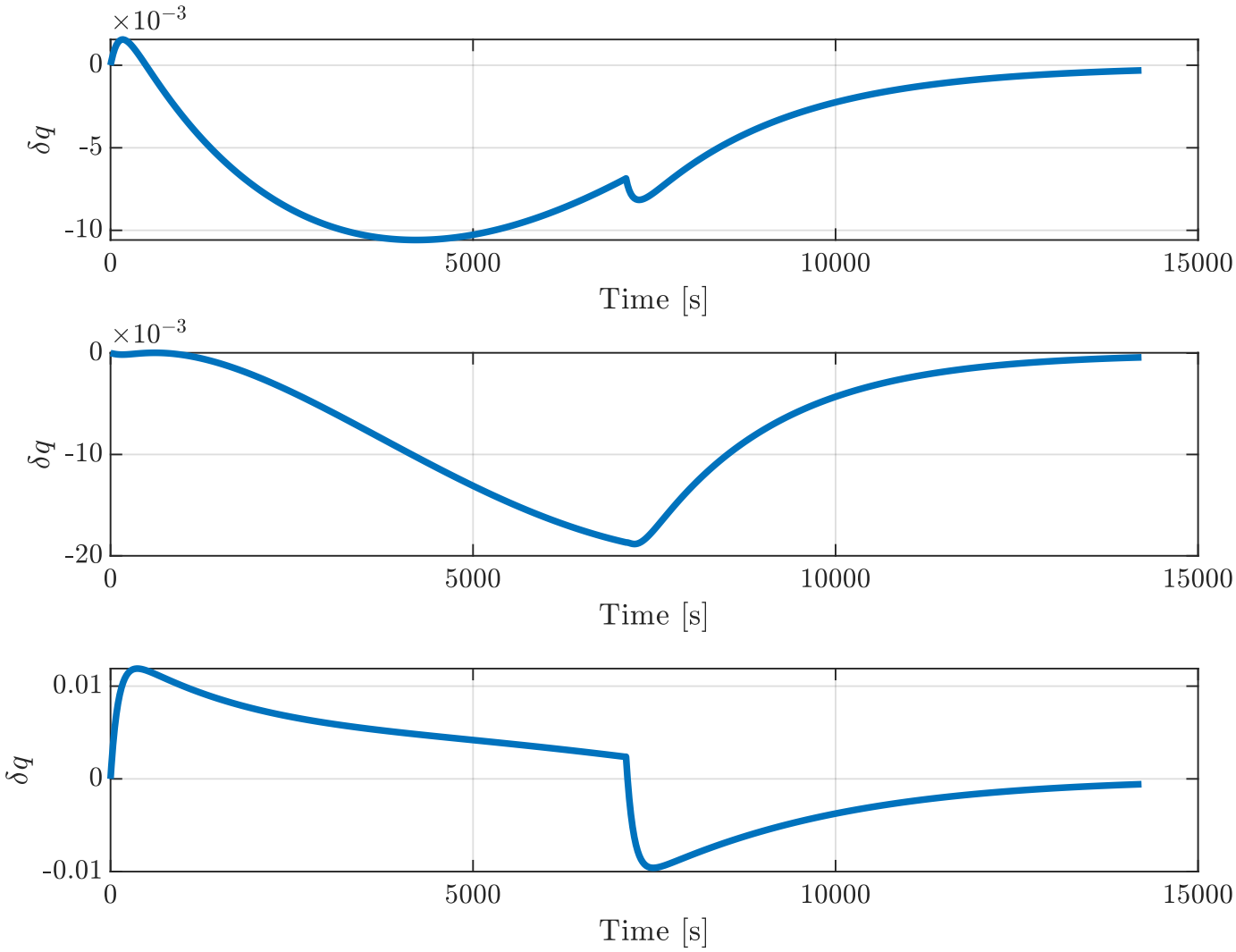


Figure 9: The attitude control error as represented by the vector components of the error quaternion. The commanded body rate profile is discontinuous at the beginning and end of the maneuver, resulting in overshoot and poor performance after those moments.

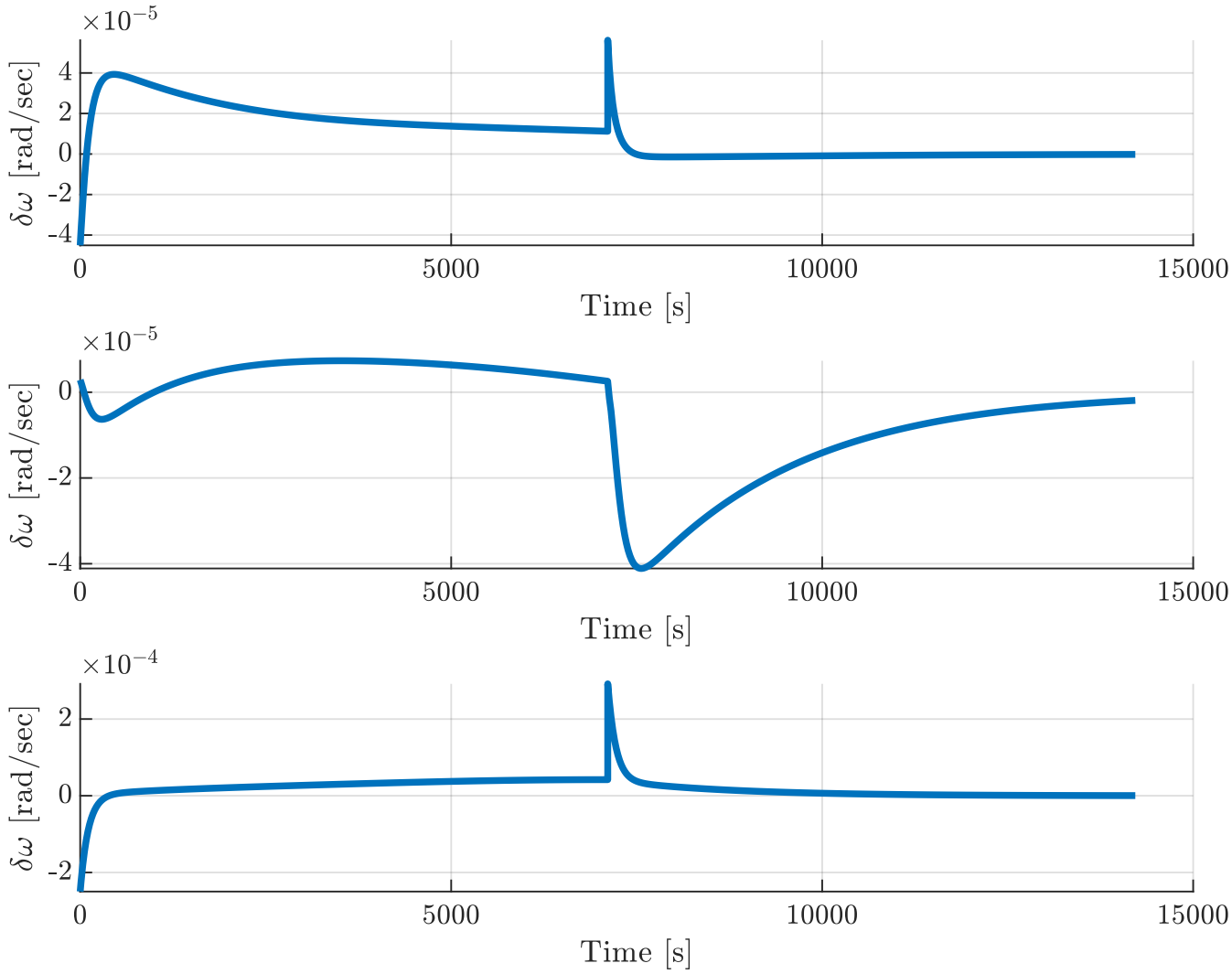


Figure 10: The attitude rate error. The commanded body rate profile is discontinuous at the beginning and end of the maneuver, resulting in overshoot and poor performance after those moments.

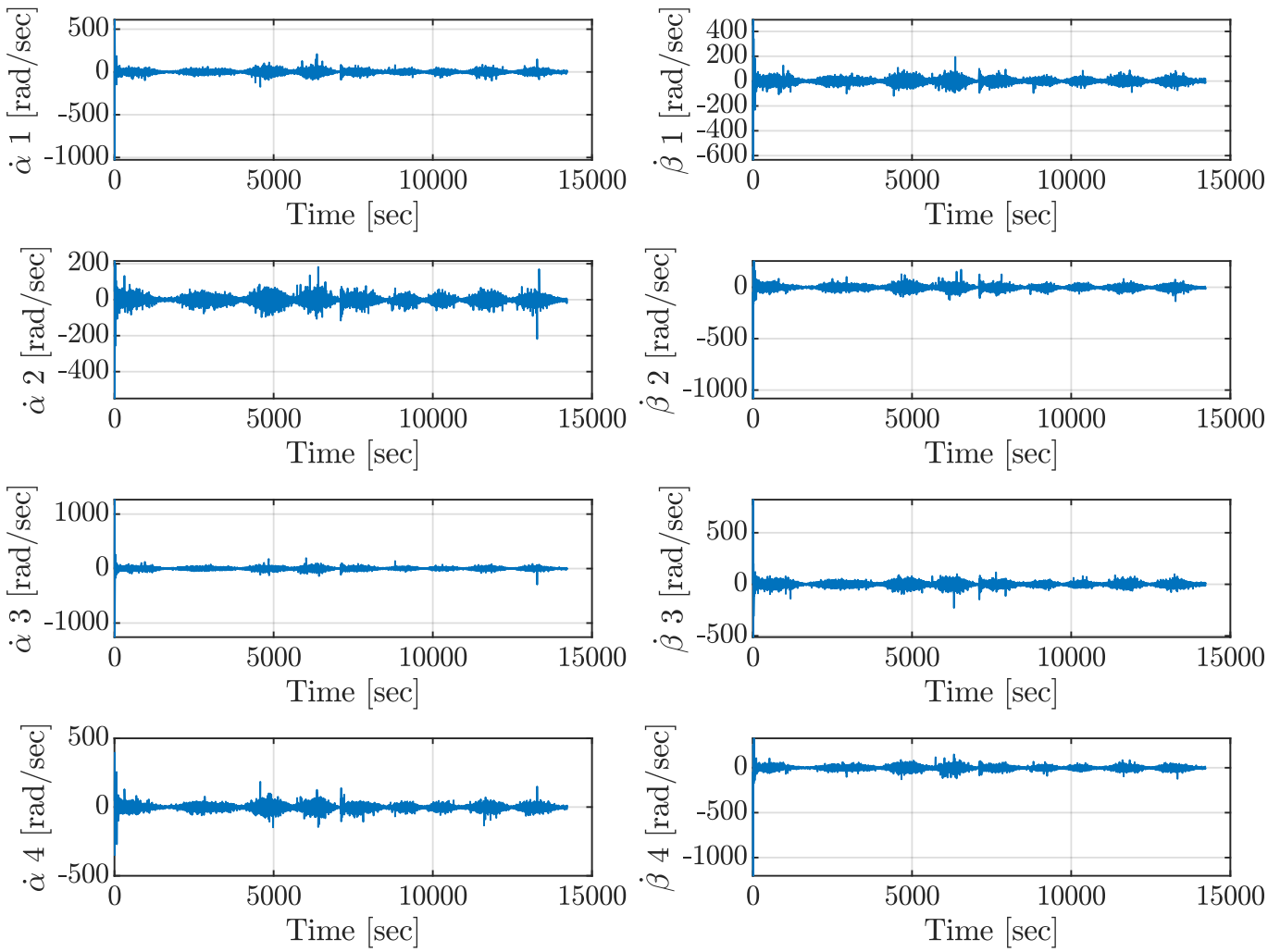


Figure 11: The gimbal rates for each CMG. These rates are extremely high and physically unrealizable. This occurs because our KD controller gains are very high and the system contains no logic to saturate the commanded gimbal rate.

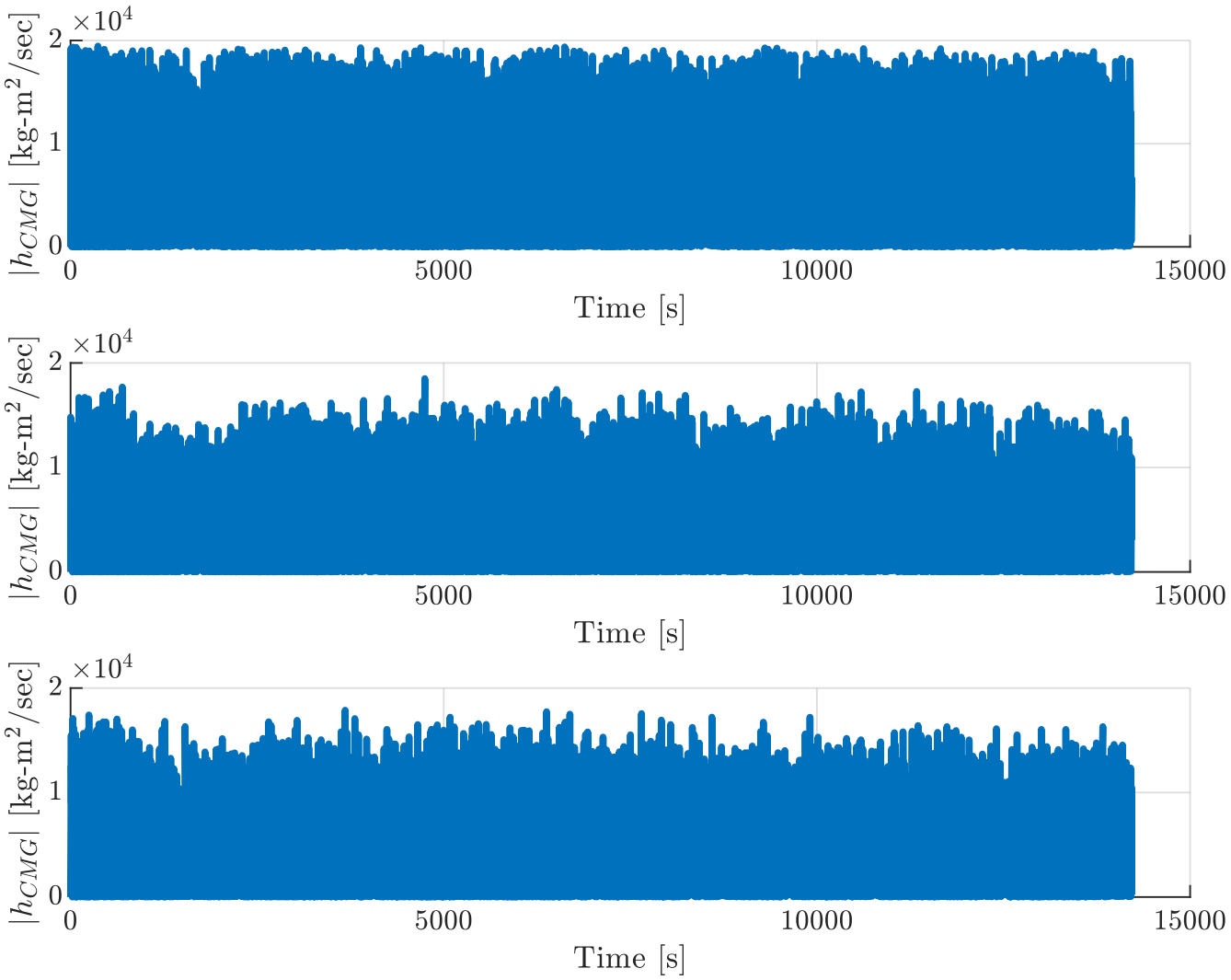


Figure 12: The magnitude of each element of the angular momentum vector for the CMG array. The rate of change of this vector is extremely high due to the physically unrealizable gimbal rates.

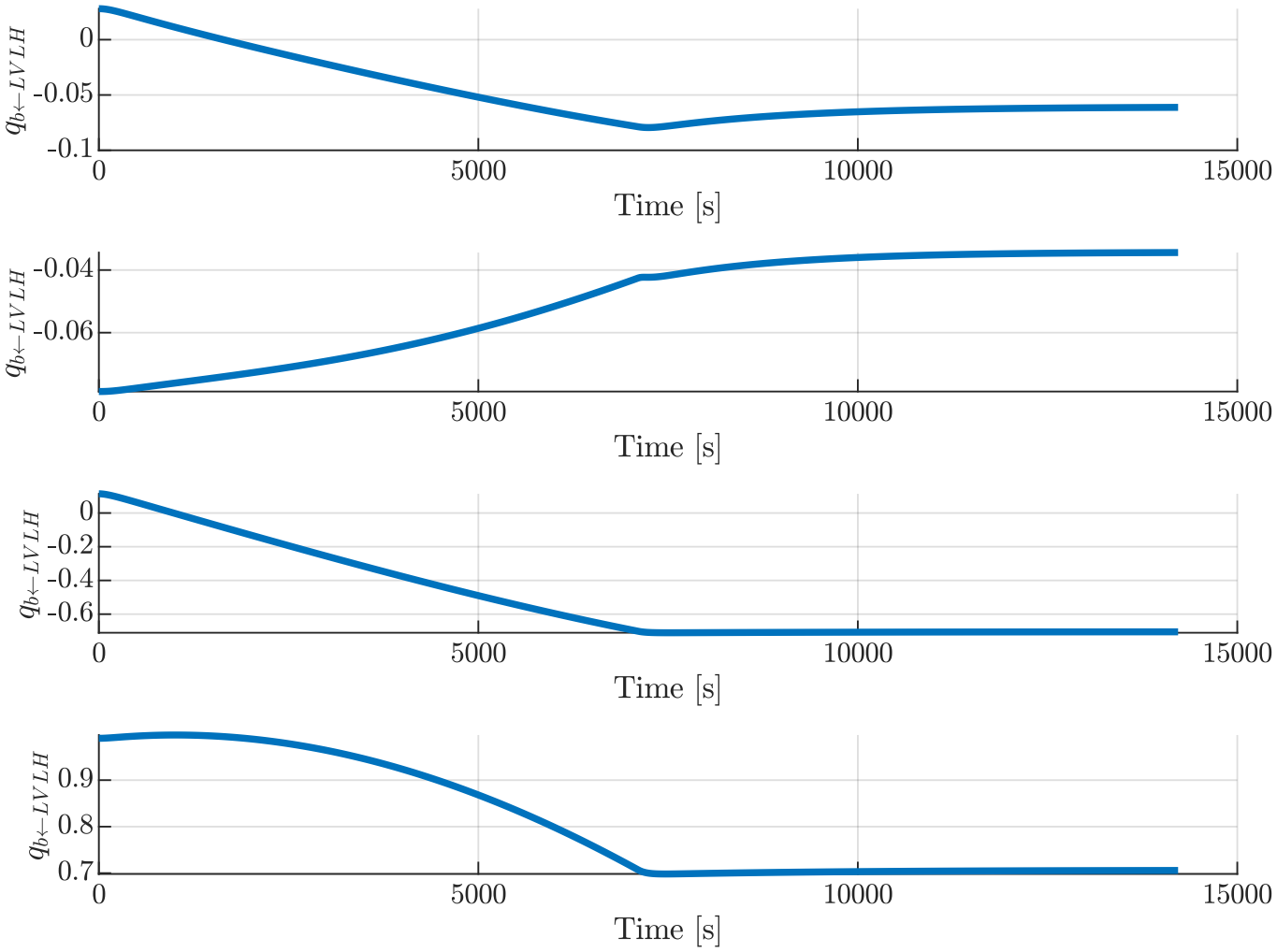


Figure 13: The true LVLH to body attitude quaternion.

4 Code

```
%% Setup
clear
close all
clc

addpath("..\util\")

%% Problem Initalization

% Gravity
mu = 398600; %km3/s2

% Orbital elements
a = 6371 + 400;
e = 0;
i = 0;
Ohm = 0;
w = 0;
theta = 0;
[r_inertial_0, v_inertial_0] = OE2State(a, e, i, Ohm, w, theta);

% Find orbit rate
n = sqrt(mu/a^3);

% Find LVLH rotation rate
w_LVLH_wrt_inertial_in_LVLH = [0, -n, 0]';

% Initial rotation between LVLH and inertial
x_LVLH_inertial_0 = v_inertial_0/norm(v_inertial_0);
z_LVLH_inertial_0 = -r_inertial_0/norm(r_inertial_0);
y_LVLH_inertial_0 = cross(z_LVLH_inertial_0,x_LVLH_inertial_0);
T_inertial2LVLH_0 = [x_LVLH_inertial_0'; y_LVLH_inertial_0'; z_LVLH_inertial_0'];
q_inertial2LVLH_0 = DCM2Quat(T_inertial2LVLH_0);

% Rotation formulations
J = [24181836 3783405 3898808
    3783405 37621803 -1171849
    3898808 -1171849 51576634];
w_b_LVLH_0 = [0 0 0]'; % Initial LVLH rotation rate, rad/sec
q_LVLH2body_0 = [0.028, -0.0788, 0.1141, 0.9899]'; % Initial attitude quaternion
%q_LVLH2body_f = q_LVLH2body_0;
q_LVLH2body_f = [-0.0607, -0.0343, -0.7045, 0.7062]'; % Attitude quaternion at end of the↙
manuever

% Initial pose and rate in inertial
q_inertial2body_0 = QuatProduct(q_LVLH2body_0,q_inertial2LVLH_0);
w_body_wrt_inertial_0 = QuatTransform(q_LVLH2body_0,w_LVLH_wrt_inertial_in_LVLH) +↙
w_b_LVLH_0;

% Final simulation time
Tf = 2*7110;
```

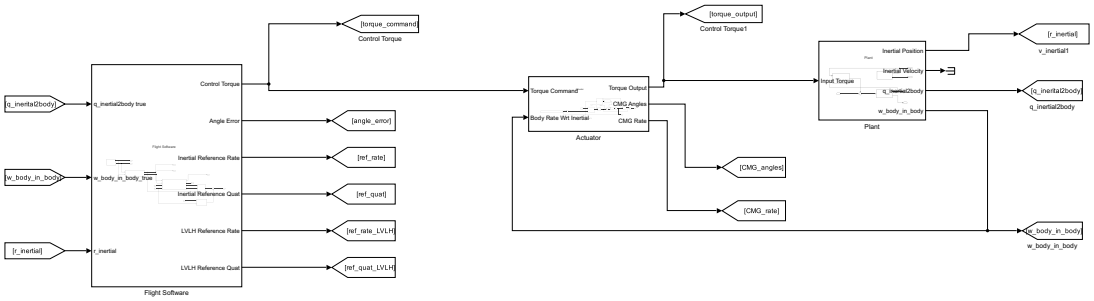
```
% Tf = 100;
% Tf = 10000;

% Final manuever time
Tf_man = 7110;

% CMG momentum
h0 = 4881;

% Maximum CMG rates
rate_max = Inf*(pi/180); % Rad/sec

%% Design the maneuver in the LVLH frame

% Change in quaternion
dq_LVLH = QuatProduct(q_LVLH2body_f,QuatInv(q_LVLH2body_0));

% Euler axis and angle change
[dtheta_LVLH, dn_LVLH] = Quat2AxisAngle(dq_LVLH);

% Find angular rate in rad/sec
w_b_LVLH_man = dtheta_LVLH/Tf_man*dn_LVLH;

%% Nonlinear controller design

kp_nonlin = 500;
kd_nonlin = 500000;

%% Main

use_CMG = false;
out_no_CMG = sim("simulink\midterm_sim.slx");

use_CMG = true;
out_w_CMG = sim("simulink\midterm_sim.slx");

%% Extract information for Part 1

tout_no_CMG = out_no_CMG.tout;
q_LVLH2body_ref = squeeze(out_no_CMG.ref_quat_LVLH);
w_body_LVLH_ref = squeeze(out_no_CMG.ref_rate_LVLH);
q_inertial2body_ref = squeeze(out_no_CMG.ref_quat);
w_body_inertial_ref_no_CMG = squeeze(out_no_CMG.ref_rate);


%% Extract Informatin for Part 2

err_quat_no_CMG = squeeze(out_no_CMG.error_quat);
q_inertial2body_no_CMG = squeeze(out_no_CMG.quat);
w_body_inertial_no_CMG = squeeze(out_no_CMG.w);
q_inertial2LVLH_no_CMG = squeeze(out_no_CMG.q_inertial2LVLH);
```

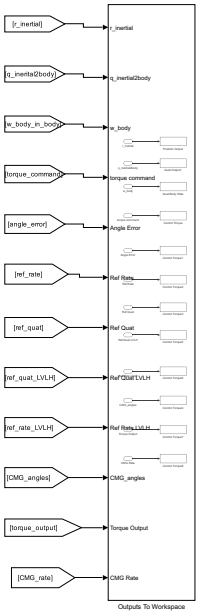
```
% Find quaternion from body to LVLH
q_LVLH2body_no_CMG = zeros(size(q_inertial2LVLH_no_CMG));
for ii = 1:length(tout_no_CMG)
    q_LVLH2body_no_CMG(:,ii) = QuatProduct(q_inertial2body_no_CMG(:,ii),QuatInv↙
(q_inertial2LVLH_no_CMG(:,ii)));
end

%% Extract Information for Part 3

tout_w_CMG = out_w_CMG.tout;
w_body_inertial_w_CMG = squeeze(out_w_CMG.w);
q_inertial2LVLH_w_CMG = squeeze(out_w_CMG.q_inertial2LVLH);
q_inertial2body_w_CMG = squeeze(out_w_CMG.quat);
CMG_rates = squeeze(out_w_CMG.CMG_rates);
err_quat_w_CMG = squeeze(out_w_CMG.error_quat);
w_body_inertial_ref_w_CMG = squeeze(out_w_CMG.ref_rate);
CMG_h = squeeze(out_w_CMG.CMG_h);

% Find quaternion from body to LVLH
q_LVLH2body_w_CMG = zeros(size(q_inertial2LVLH_no_CMG));
for ii = 1:length(tout_w_CMG)
    q_LVLH2body_w_CMG(:,ii) = QuatProduct(q_inertial2body_w_CMG(:,ii),QuatInv↙
(q_inertial2LVLH_w_CMG(:,ii)));
end


%% Plotting Part 1

figure
for ii = 1:3
    subplot(3,1,ii)
    hold on
    plot(tout_no_CMG, w_body_LVLH_ref(ii,:),'LineWidth',2)
    xlabel('Time [s]','Interpreter','latex')
    ylabel('$\omega_{LVLH}$ [rad/sec]','Interpreter','latex')
    grid on
end
saveas(gcf,"latex/figs/P1Q1.pdf")

figure
for ii = 1:4
    subplot(4,1,ii)
    hold on
    plot(tout_no_CMG, q_LVLH2body_ref(ii,:),'LineWidth',2)
    xlabel('Time [s]','Interpreter','latex')
    ylabel("$q_{b \leftarrow LVLH}$","Interpreter","latex")
    grid on
end
saveas(gcf,"latex/figs/P1Q2.pdf")
```

```
figure
for ii = 1:3
    subplot(3,1,ii)
    hold on
    plot(tout_no_CMG, w_body_inertial_ref_no_CMG(ii,:),'LineWidth',2)
    xlabel('Time [s]','Interpreter','latex')
    ylabel('$\omega_{inertial}$ [rad/sec]','Interpreter','latex')
    grid on
end
saveas(gcf,"latex/figs/P1Q3.pdf")

figure
for ii = 1:4
    subplot(4,1,ii)
    hold on
    plot(tout_no_CMG, q_inertial2body_ref(ii,:),'LineWidth',2)
    xlabel('Time [s]')
    ylabel("$q_{b \leftarrow inertial}$",'Interpreter','latex')
    grid on
end
saveas(gcf,"latex/figs/P1Q4.pdf")

%% Plotting Part 2

figure
for ii = 1:3
    subplot(3,1,ii)
    plot(tout_no_CMG, err_quat_no_CMG(ii,:),'LineWidth',2)
    xlabel('Time [s]','Interpreter','latex')
    ylabel("$q_{b \leftarrow \bar{b}}$","Interpreter","latex")
    grid on
end
saveas(gcf,"latex/figs/P2Q1.pdf")

figure
for ii = 1:3
    subplot(3,1,ii)
    hold on
    plot(tout_no_CMG, w_body_inertial_ref_no_CMG(ii,:) - w_body_inertial_no_CMG↙
(ii,:),'LineWidth',2)
    xlabel('Time [s]',"Interpreter","latex")
    ylabel('$\delta \omega$ [rad/sec]',"Interpreter","latex")
    grid on
end
saveas(gcf,"latex/figs/P2Q2.pdf")

figure
for ii = 1:4
    subplot(4,1,ii)
    hold on
    plot(tout_no_CMG, q_inertial2body_no_CMG(ii,:),'LineWidth',2)
```

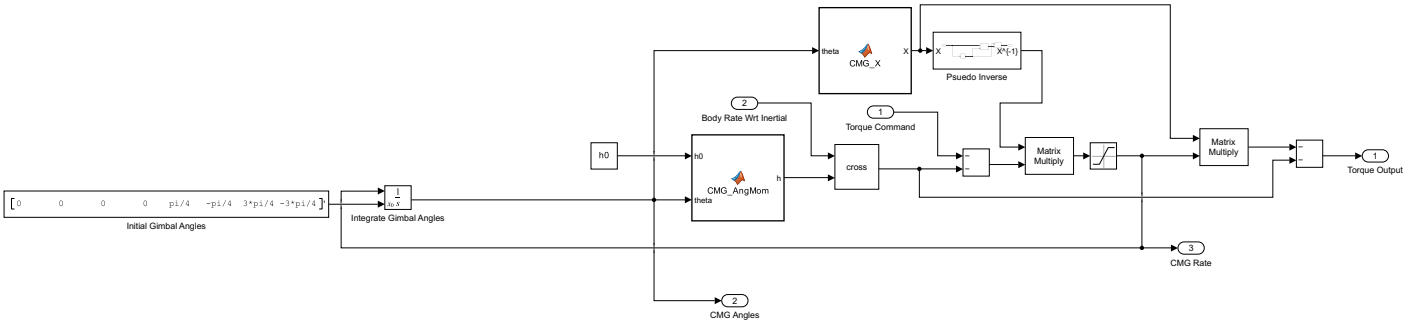
```
    xlabel('Time [s]','Interpreter','latex')
    ylabel("$q_{b \leftarrow inertial}$","Interpreter","latex")
    grid on
end
saveas(gcf,"latex/figs/P2Q3.pdf")

figure
for ii = 1:4
    subplot(4,1,ii)
    hold on
    plot(tout_no_CMG, q_LVLH2body_no_CMG(ii,:),"LineWidth",2)
    xlabel('Time [s]', "Interpreter","latex")
    ylabel("$q_{b \leftarrow LVLH}$","Interpreter","latex")
    grid on
end
saveas(gcf,"latex/figs/P2Q4.pdf")

%% Plotting Part 3

figure
for ii = 1:3
    subplot(3,1,ii)
    plot(tout_w_CMG, err_quat_w_CMG(ii,:),"LineWidth",2)
    xlabel('Time [s]',"Interpreter","latex")
    ylabel("$\delta q$","Interpreter","latex")
    grid on
end
saveas(gcf,"latex/figs/P3Q1.pdf")

figure
for ii = 1:3
    subplot(3,1,ii)
    hold on
    plot(tout_w_CMG, w_body_inertial_ref_w_CMG(ii,:) - w_body_inertial_w_CMG(ii,:),"↙
LineWidth",2)
    xlabel('Time [s]',"Interpreter","latex")
    ylabel('$\delta \omega$ [rad/sec]',"Interpreter","latex")
    grid on
end
saveas(gcf,"latex/figs/P3Q2.pdf")

figure
for ii = 1:4
    subplot(4,2,2*ii-1)
    plot(tout_w_CMG,CMG_rates(ii,:))
    xlabel('Time [sec]',"Interpreter","latex")
    ylabel(strcat("$\dot{\alpha}$ ",num2str(ii)," [rad/sec]"),"Interpreter",'latex')
    grid on

    subplot(4,2,2*ii)
    plot(tout_w_CMG,CMG_rates(ii+4,:))
```

```
    xlabel('Time [sec]',"Interpreter","latex")
    ylabel(strcat("$\dot{\beta}$ ",num2str(ii)," [rad/sec]"),"Interpreter",'latex')
    grid on

end
saveas(gcf,"latex/figs/P3Q3.pdf")

figure
for ii = 1:3
    subplot(3,1,ii)
    hold on
    plot(tout_w_CMG, abs(CMG_h(ii,:)),"LineWidth",2)
    xlabel('Time [s]',"Interpreter","latex")
    ylabel('$|h_{CMG}|$ [kg-m\textsuperscript{2}/sec]',"Interpreter","latex")
    grid on
end
saveas(gcf,"latex/figs/P3Q4.pdf")

figure
for ii = 1:4
    subplot(4,1,ii)
    hold on
    plot(tout_w_CMG, q_LVLH2body_w_CMG(ii,:),"LineWidth",2)
    xlabel('Time [s]',"Interpreter","latex")
    ylabel("$q_{b \leftarrow LVLH}$","Interpreter","latex")
    grid on
end
saveas(gcf,"latex/figs/P3Q5.pdf")
```


[torque_command]
Control Torque
[q_inertia2body]
q_inertia2body true
Control Torque
Angle Error
Inertial Reference Rate
Inertial Reference Quat
LVLH Reference Rate
LVLH Reference Quat
[w_body_in_body]
w_body_in_body_true
[r_inertial]
r_inertial
[angle_error]
[ref_rate]
[ref_quat]
[ref_rate_LVLH]
[ref_quat_LVLH]
Flight Software
[torque_output]
Control Torque1
Torque Command
Body Rate Wrt Inertial
Torque Output
CMG Angles
CMG Rate
Actuator
[CMG_angles]
[CMG_rate]
Inertial Position
Inertial Velocity
Input Torque
q_inertia2body
w_body_in_body
Plant
[r_inertial]
v_inertial1
[q_inertia2body]
q_inertia2body
[w_body_in_body]
w_body_in_body

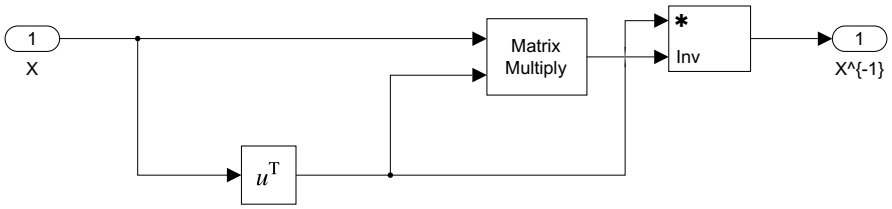

[r_inertial]
r_inertial
[q_inertia2body]
q_inertia2body
[w_body_in_body]
w_body
[torque_command]
torque command
[angle_error]
Angle Error
[ref_rate]
Ref Rate
[ref_quat]
Ref Quat
[ref_quat_LVLH]
Ref Quat LVLH
[ref_rate_LVLH]
Ref Rate LVLH
[CMG_angles]
CMG_angles
[torque_output]
Torque Output
[CMG_rate]
CMG Rate
Outputs To Workspace

Actuator

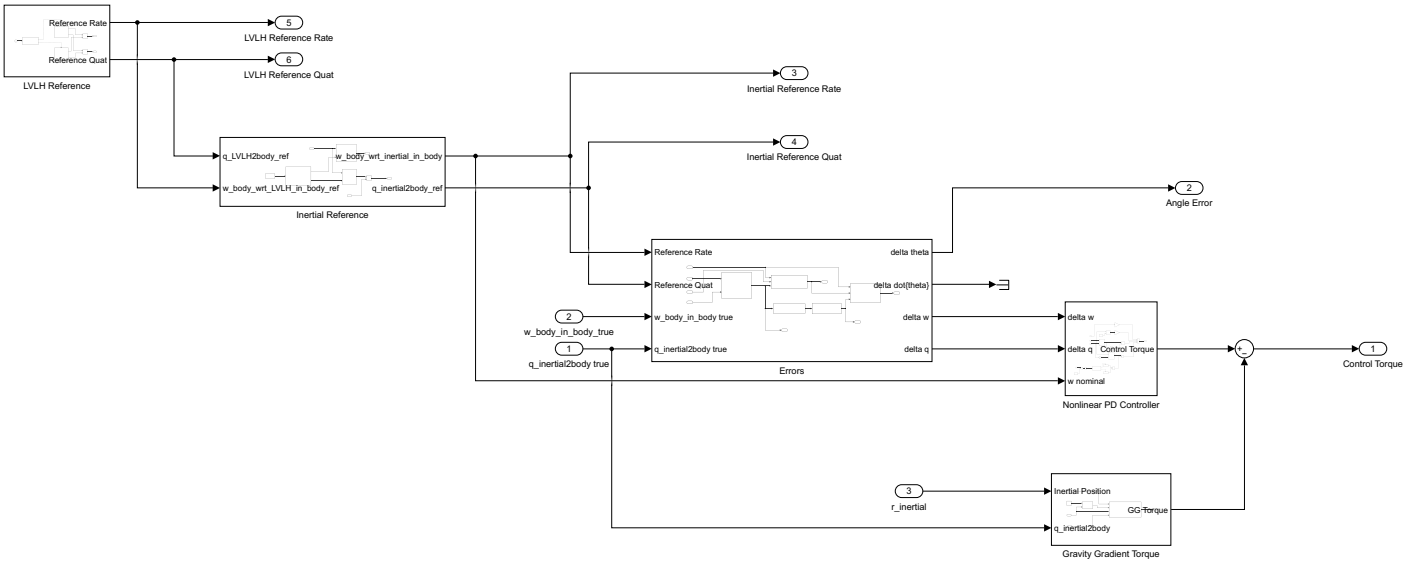

[0 0 0 0 pi/4 -pi/4 3*pi/4 -3*pi/4]
Initial Gimbal Angles
Integrate Gimbal Angles
h0
theta
CMG_X
X
X^(-1)
Psuedo Inverse
2
Body Rate Wrt Inertial
1
Torque Command
h0
CMG_AngMom
theta
h
cross
Matrix Multiply
Matrix Multiply
1
Torque Output
3
CMG Rate
2
CMG Angles

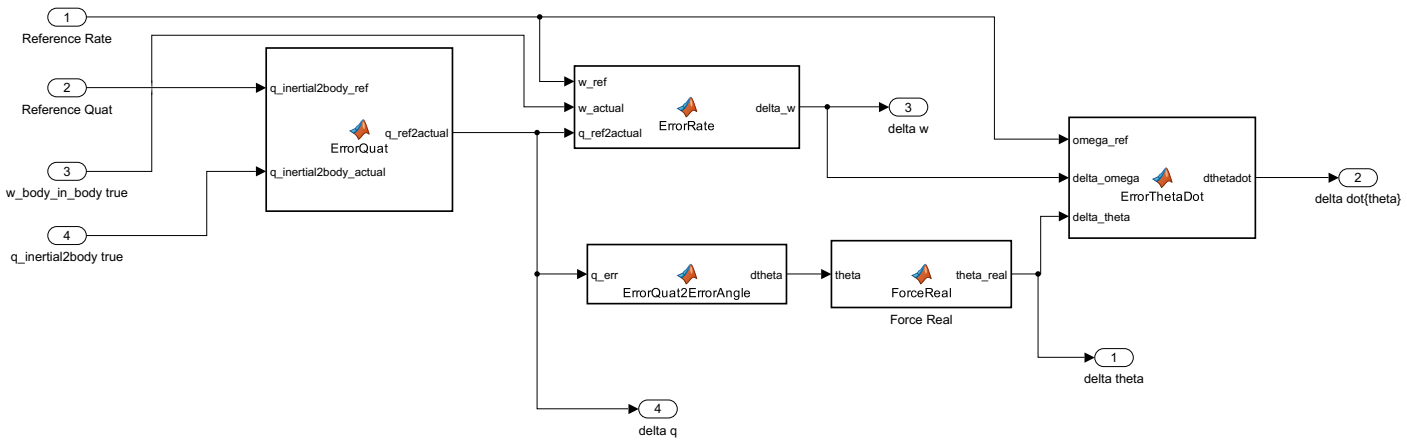
```
function h = CMG_AngMom(h0, theta)
% Produces the angular momentum vector for the CMGs

% Extract alpha and betas
alphas = theta(1:4);
betas = theta(5:8);

h = zeros(3,1);
for ii = 1:4
    h = h + h0*[sin(alphas(ii)); cos(alphas(ii))*cos(betas(ii)); cos(alphas(ii))*sin(betas(ii))];
end
```

```
function X = CMG_X(theta)
% Produces the matrix of CMG spin axes

% Extract alpha and betas
alphas = theta(1:4);
betas = theta(5:8);

X = zeros(3,8);
for ii = 1:4
    X(:,2*ii-1:2*ii) = [cos(alphas(ii)), 0;
        -sin(alphas(ii))*cos(betas(ii)), -cos(alphas(ii))*sin(betas(ii));
        -sin(alphas(ii))*sin(betas(ii)), cos(alphas(ii))*cos(betas(ii))];
end
```


1
X
Matrix
Multiply
*
Inv
1
X^{-1}
u^{T}

Flight Software




1
Reference Rate
2
Reference Quat
3
w_body_in_body true
4
q_inertial2body true
q_inertial2body_ref
ErrorQuat
q_ref2actual
q_inertial2body_actual
w_ref
w_actual
q_ref2actual
ErrorRate
delta_w
3
delta w
omega_ref
delta_omega
ErrorThetaDot
dthetadot
delta_theta
2
delta dot{theta}
q_err
ErrorQuat2ErrorAngle
dtheta
theta
ForceReal
theta_real
Force Real
1
delta theta
4
delta q

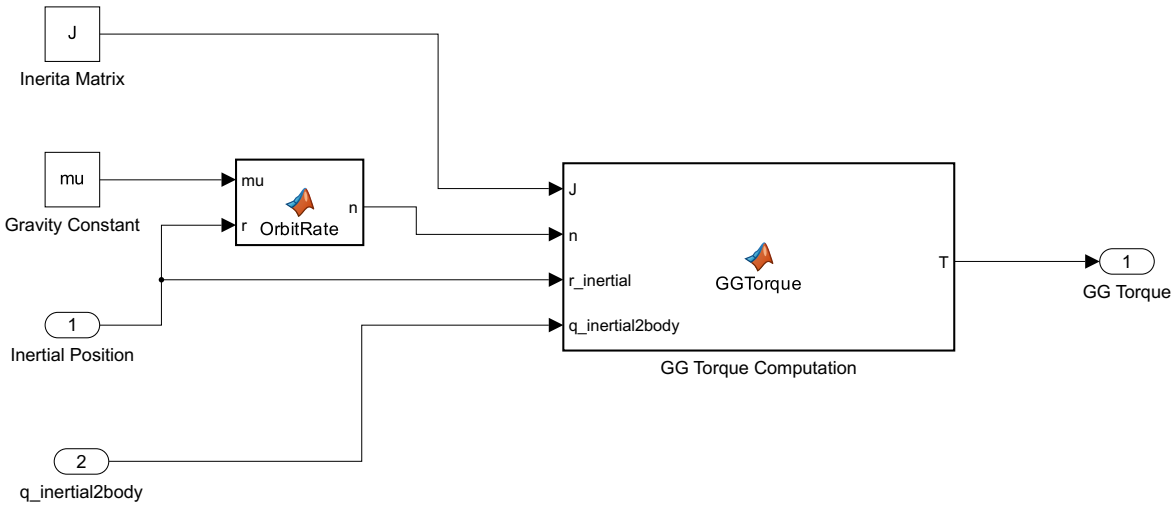
```
function theta_real = ForceReal(theta)
theta_real = real(theta);
```

```
function q_ref2actual = ErrorQuat(q_inertial2body_ref, q_inertial2body_actual)

q_ref2actual = QuatProduct(q_inertial2body_actual,QuatInv(q_inertial2body_ref));

% Force normalization
q_ref2actual = q_ref2actual/norm(q_ref2actual);
```

```
function delta_w = ErrorRate(w_ref, w_actual, q_ref2actual)
delta_w = w_actual - QuatTransform(q_ref2actual,w_ref);
```

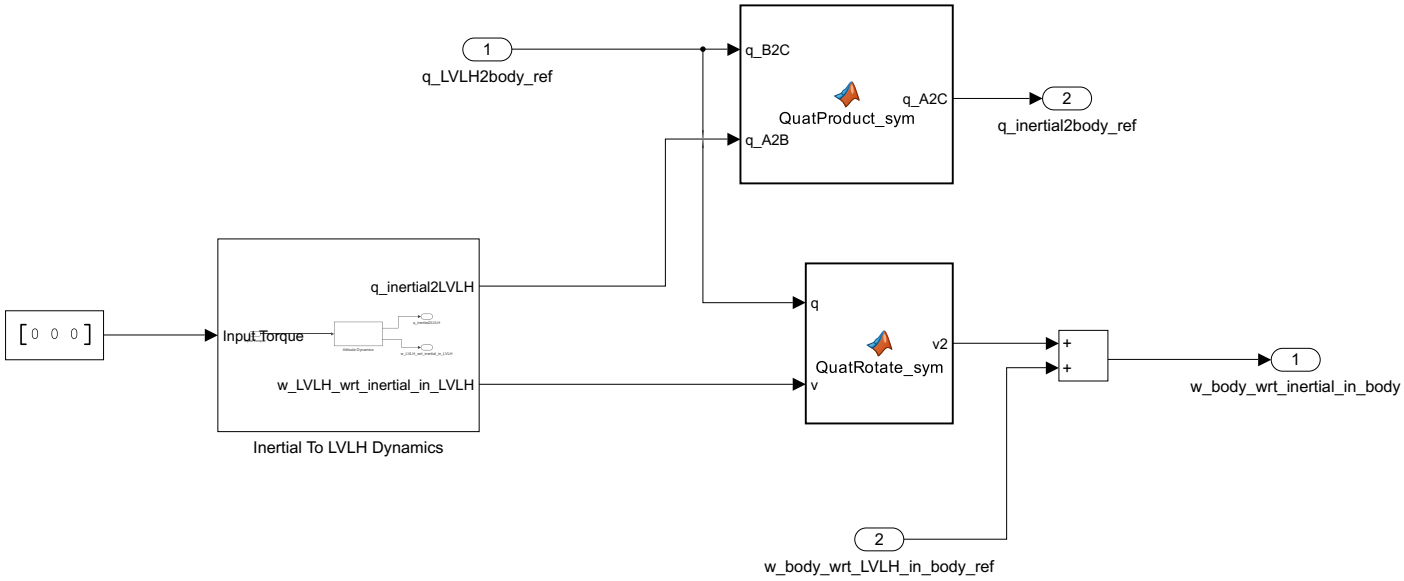
```
function dtheta = ErrorQuat2ErrorAngle(q_err)

% Quaternion components
q_v = real(q_err(1:3));
q_s = real(q_err(4));

dtheta = [atan2(q_v(1),q_s);
    atan2(q_v(2),q_s);
    atan2(q_v(3),q_s)];
```

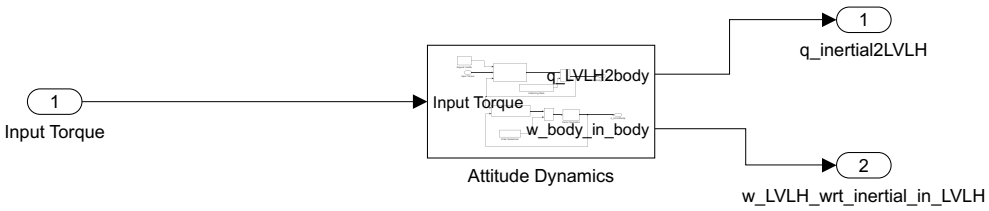
```
function dthetadot = ErrorThetaDot(omega_ref, delta_omega, delta_theta)
dthetadot = delta_omega - cross(omega_ref,delta_theta);
```

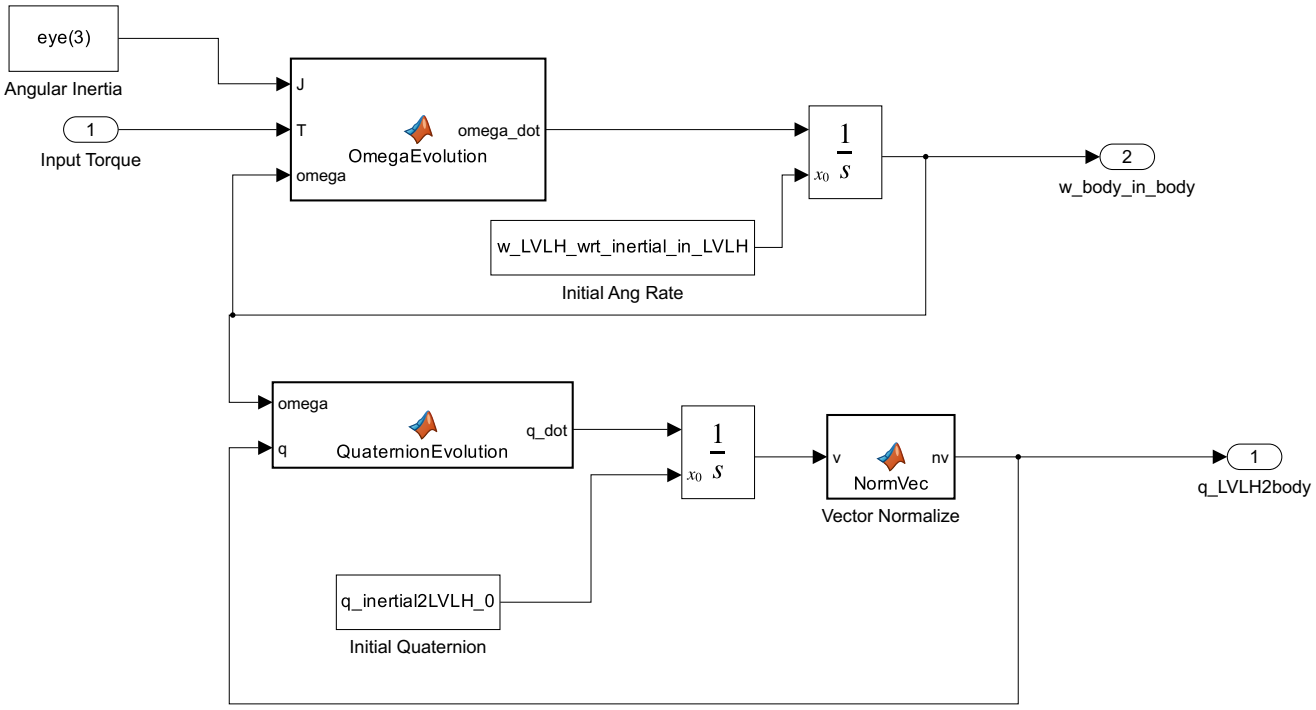

J
Inerita Matrix
mu
Gravity Constant
mu
r
OrbitRate
n
1
Inertial Position
2
q_inertial2body
J
n
r_inertial
q_inertial2body
GGTorque
T
GG Torque Computation
1
GG Torque

```
function T = GGTorque(J, n, r_inertial, q_inertial2body)

% Down in body frame
down_inertial = -r_inertial/norm(r_inertial);
down_body = QuatTransform(q_inertial2body,down_inertial);

% Gravity gradient torque
T = 3*n^2*cross(down_body,J*down_body);
```

```
function n = OrbitRate(mu,r)
n = sqrt(mu/norm(r)^3);
```


1
q_LVLH2body_ref
q_B2C
q_A2B
QuatProduct_sym
q_A2C
2
q_inertial2body_ref
[0 0 0]
Input Torque
q_inertial2LVLH
w_LVLH_wrt_inertial_in_LVLH
Inertial To LVLH Dynamics
q
v
QuatRotate_sym
v2
+
+
1
w_body_wrt_inertial_in_body
2
w_body_wrt_LVLH_in_body_ref


1
Input Torque
Input Torque
q_LVLH2body
w_body_in_body
Attitude Dynamics
1
q_inertial2LVLH
2
w_LVLH_wrt_inertial_in_LVLH


eye(3)
Angular Inertia
1
Input Torque
J
T
omega
omega_dot
OmegaEvolution
1
s
x0
2
w_body_in_body
w_LVLH_wrt_inertial_in_LVLH
Initial Ang Rate
omega
q
q_dot
QuaternionEvolution
1
s
x0
v
nv
NormVec
Vector Normalize
1
q_LVLH2body
q_inertial2LVLH_0
Initial Quaternion

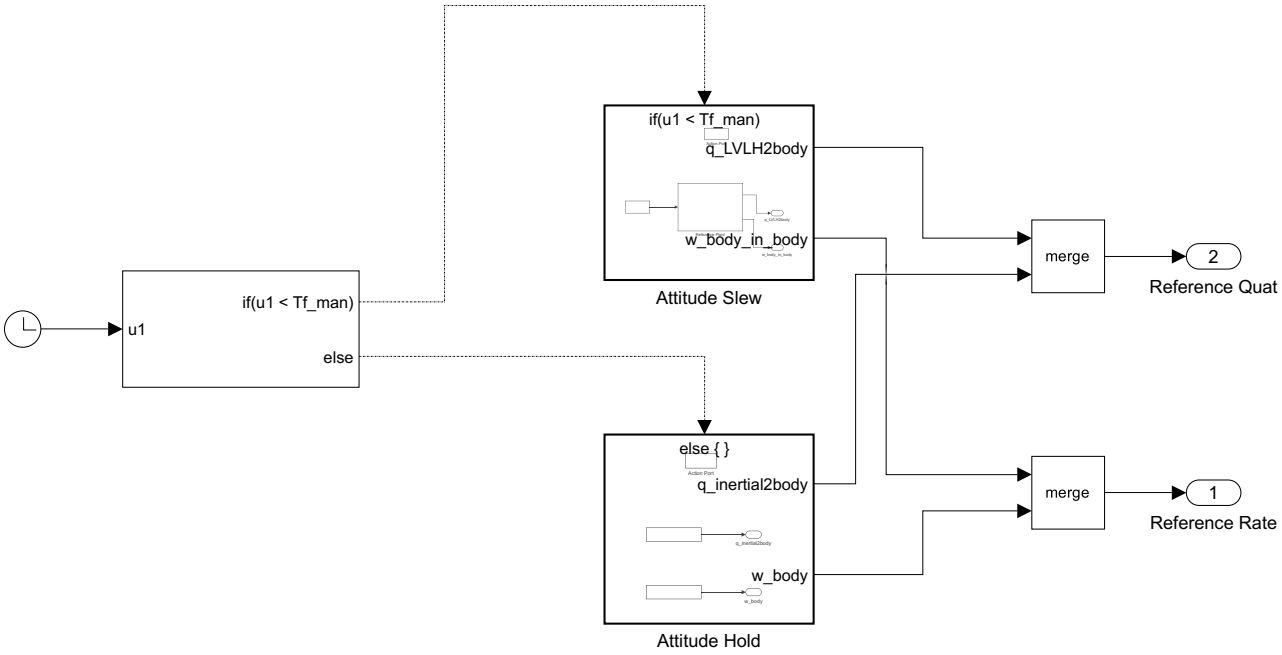
```
function q_dot = QuaternionEvolution(omega, q)
intermed = [omega; 0];
q_dot = 0.5*QuatProduct(intermed,q);
```

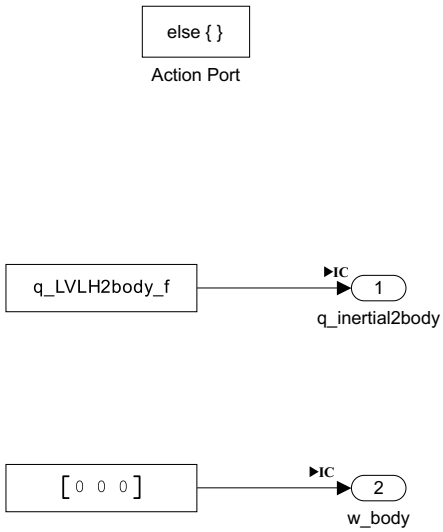
```
function omega_dot = OmegaEvolution(J, T, omega)
omega_dot = J\(T - cross(omega, J*omega));
```

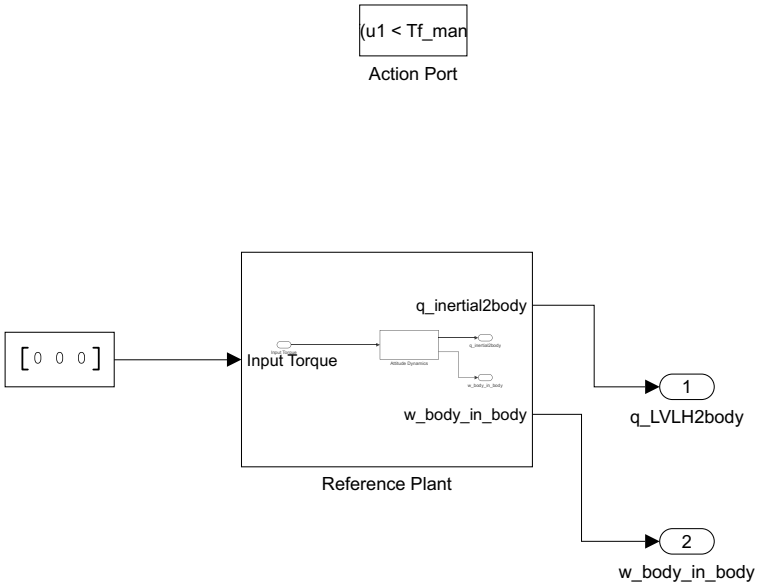
```
function nv = NormVec(v)
nv = v/norm(v);
```

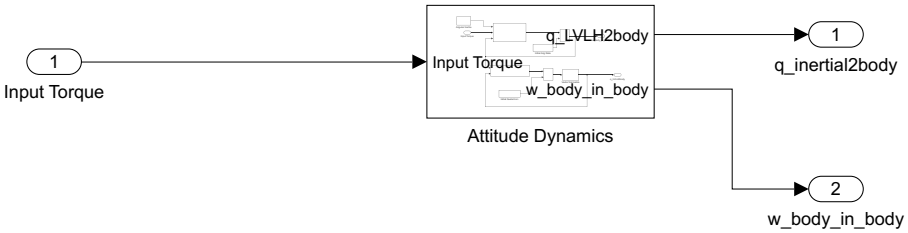
```
function q_A2C = QuatProduct_sym(q_B2C,q_A2B)
q_A2C = QuatProduct(q_B2C,q_A2B);
```

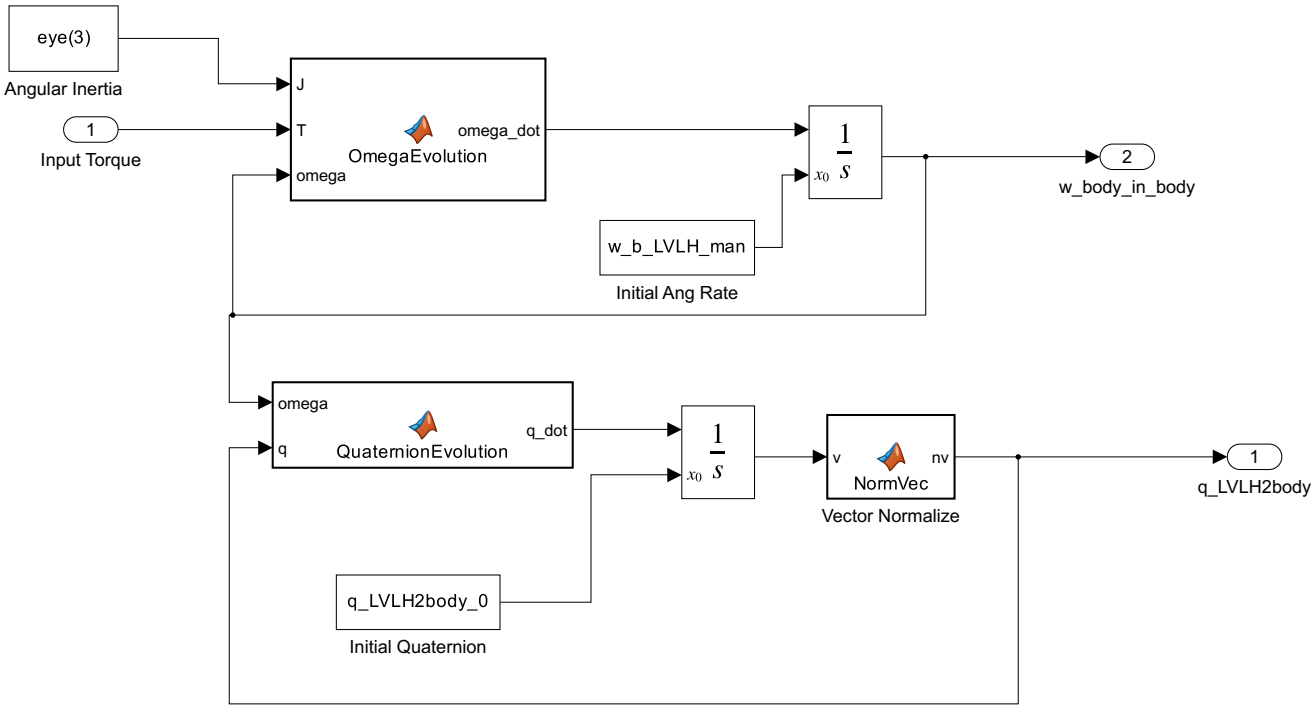
```
function v2 = QuatRotate_sym(q,v)
v2 = QuatTransform(q,v);
```


if(u1 < Tf_man)
q_LVLH2body
w_body_in_body
Attitude Slew
merge
2
Reference Quat
u1
if(u1 < Tf_man)
else
else { }
q_inertial2body
w_body
Attitude Hold
merge
1
Reference Rate


else { }
Action Port
q_LVLH2body_f
IC
1
q_inertial2body
[0 0 0]
IC
2
w_body


(u1 < Tf_man
Action Port
[0 0 0]
Input Torque
q_inertial2body
w_body_in_body
Reference Plant
1
q_LVLH2body
2
w_body_in_body

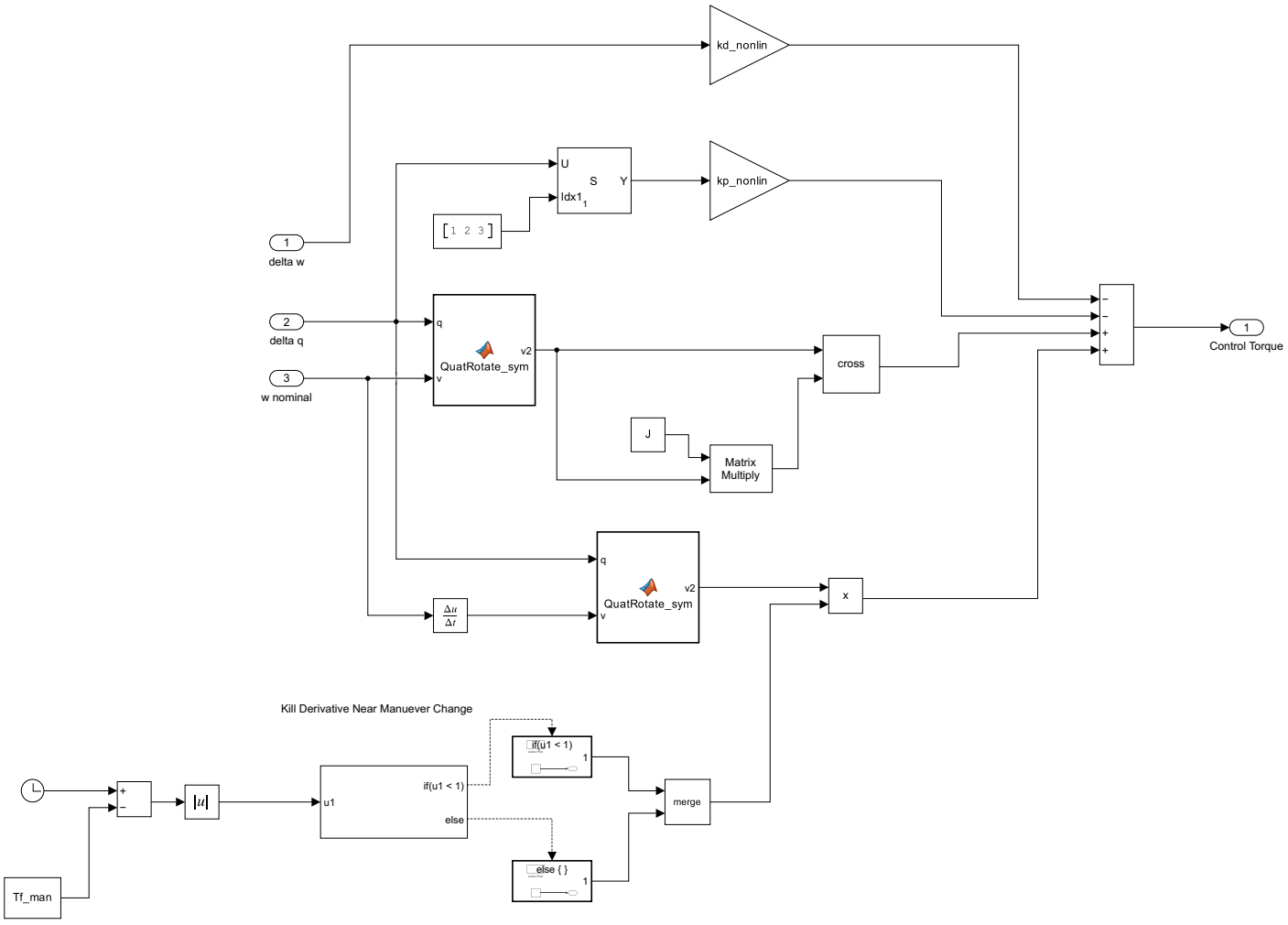

1
Input Torque
q_LVLH2body
Input Torque
w_body_in_body
Attitude Dynamics
1
q_inertial2body
2
w_body_in_body

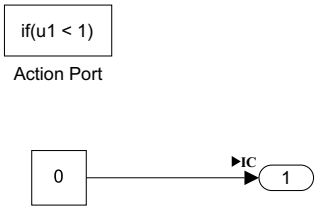

eye(3)
Angular Inertia
1
Input Torque
J
T
omega
omega_dot
OmegaEvolution
1
s
x0
2
w_body_in_body
w_b_LVLH_man
Initial Ang Rate
omega
q
q_dot
QuaternionEvolution
1
s
x0
v
nv
NormVec
Vector Normalize
1
q_LVLH2body
q_LVLH2body_0
Initial Quaternion

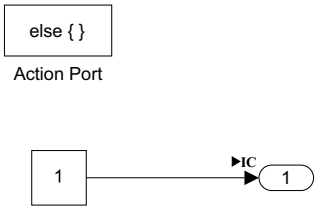
```
function q_dot = QuaternionEvolution(omega, q)
intermed = [omega; 0];
q_dot = 0.5*QuatProduct(intermed,q);
```

```
function omega_dot = OmegaEvolution(J, T, omega)
omega_dot = J\(T - cross(omega, J*omega));
```

```
function nv = NormVec(v)
nv = v/norm(v);
```

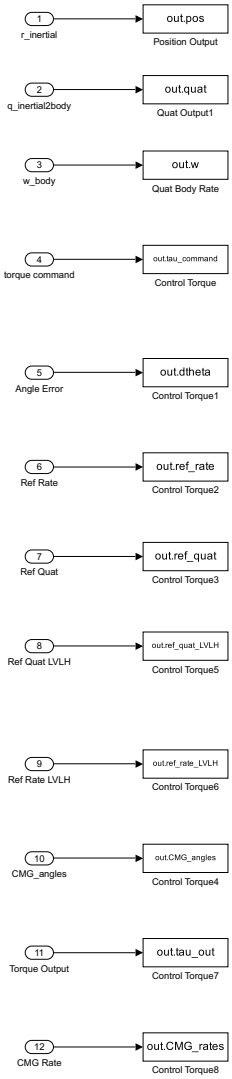

kd_nonlin
U
S
Y
Idx1
kp_nonlin
[1 2 3]
1
delta w
2
delta q
3
w nominal
q
v2
QuatRotate_sym
v
cross
J
Matrix Multiply
1
Control Torque
Δu/Δt
x
Kill Derivative Near Manuever Change
if(u1 < 1)
1
merge
u1
else
else { }
Tf_man


if(u1 < 1)
Action Port
0
IC
1

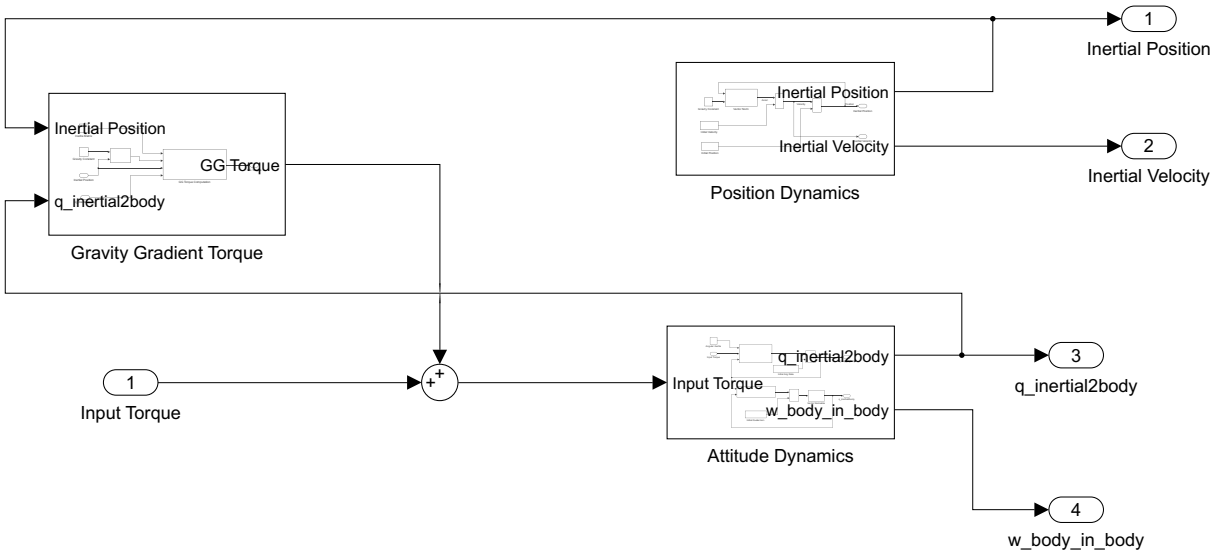

else { }
Action Port
1
IC
1

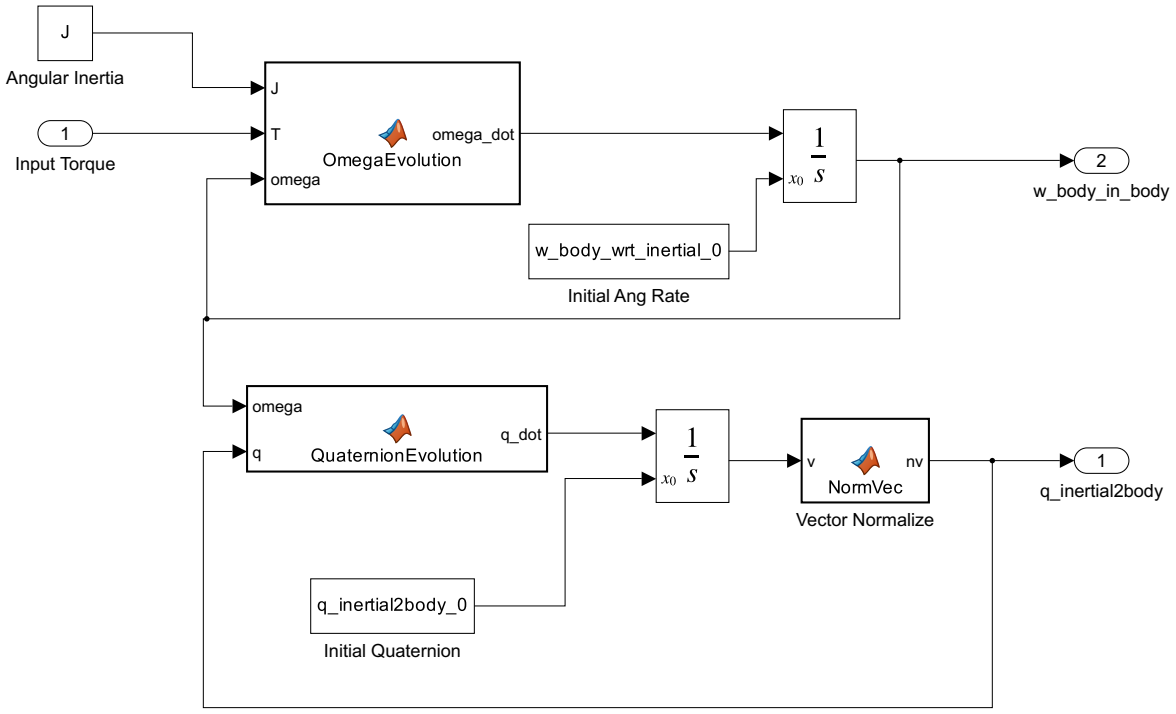
```
function v2 = QuatRotate_sym(q,v)
v2 = QuatTransform(q,v);
```

```
function v2 = QuatRotate_sym(q,v)
v2 = QuatTransform(q,v);
```


1
r_inertial
out.pos
Position Output
2
q_inertial2body
out.quat
Quat Output1
3
w_body
out.w
Quat Body Rate
4
torque command
out.tau_command
Control Torque
5
Angle Error
out.dtheta
Control Torque1
6
Ref Rate
out.ref_rate
Control Torque2
7
Ref Quat
out.ref_quat
Control Torque3
8
Ref Quat LVLH
out.ref_quat_LVLH
Control Torque5
9
Ref Rate LVLH
out.ref_rate_LVLH
Control Torque6
10
CMG_angles
out.CMG_angles
Control Torque4
11
Torque Output
out.tau_out
Control Torque7
12
CMG Rate
out.CMG_rates
Control Torque8

Plant

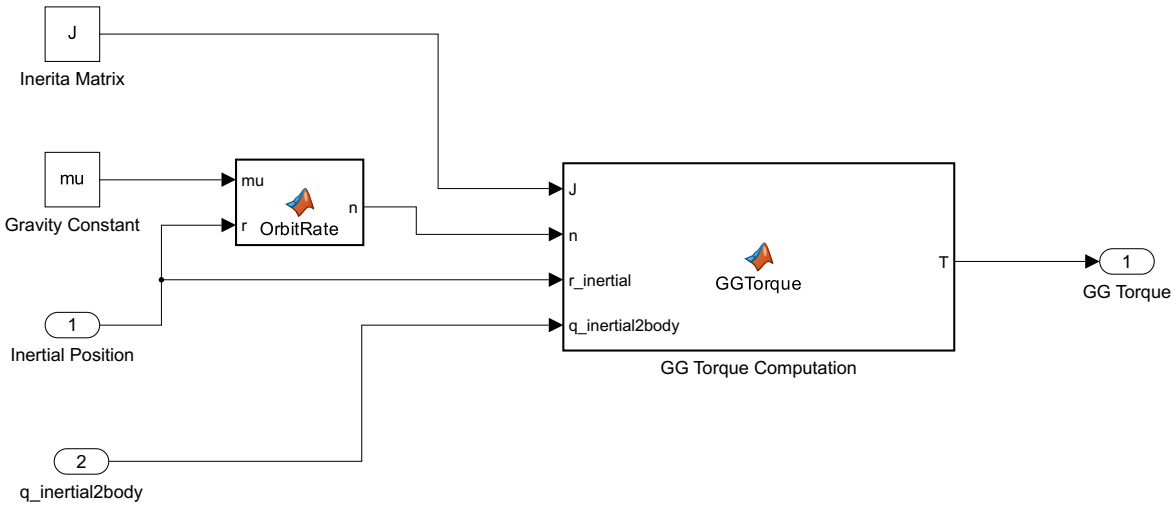

1
Inertial Position
Inertial Position
Inertial Velocity
Position Dynamics
2
Inertial Velocity
Inertial Position
GG Torque
q_inertial2body
Gravity Gradient Torque
1
Input Torque
+
+
q_inertial2body
Input Torque
w_body_in_body
Attitude Dynamics
3
q_inertial2body
4
w_body_in_body


J
Angular Inertia
1
Input Torque
J
T
omega
omega_dot
OmegaEvolution
1
s
x0
2
w_body_in_body
w_body_wrt_inertial_0
Initial Ang Rate
omega
q
q_dot
QuaternionEvolution
1
s
x0
v
nv
NormVec
Vector Normalize
1
q_inertial2body
q_inertial2body_0
Initial Quaternion

```
function q_dot = QuaternionEvolution(omega, q)
intermed = [omega; 0];
q_dot = 0.5*QuatProduct(intermed,q);
```

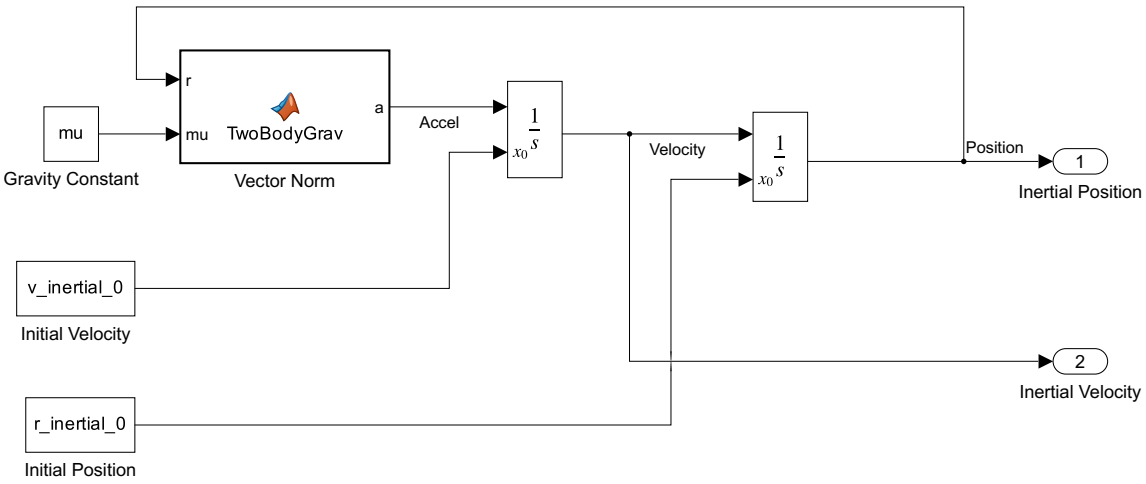
```
function omega_dot = OmegaEvolution(J, T, omega)
omega_dot = J\(T - cross(omega, J*omega));
```

```
function nv = NormVec(v)
nv = v/norm(v);
```


J
Inerita Matrix
mu
Gravity Constant
mu
r
OrbitRate
n
1
Inertial Position
2
q_inertial2body
J
n
r_inertial
q_inertial2body
GGTorque
T
GG Torque Computation
1
GG Torque

```
function T = GGTorque(J, n, r_inertial, q_inertial2body)

% Down in body frame
down_inertial = -r_inertial/norm(r_inertial);
down_body = QuatTransform(q_inertial2body,down_inertial);

% Gravity gradient torque
T = 3*n^2*cross(down_body,J*down_body);
```

```
function n = OrbitRate(mu,r)
n = sqrt(mu/norm(r)^3);
```


r
mu
TwoBodyGrav
a
Vector Norm
Gravity Constant
Accel
1
s
x0
Velocity
Position
1
Inertial Position
v_inertial_0
Initial Velocity
2
Inertial Velocity
r_inertial_0
Initial Position

```
function a = TwoBodyGrav(r,mu)

nr = norm(r);
a = -mu*r/(nr^3);
```