```
%% Setup
clear
close all
clc

addpath("..\util\","..\..\matlabScripts")

modelname = 'project_sim';

%% Symbolic math for some functions
perform_symbolic = true;
if(perform_symbolic)
    w_hat = sym('w_hat',[3 1],'real');
    syms dt tau 'real'

    wx = CrossProductMat(w_hat);
    A = ExpmSkewSym(-wx*dt);
    B = int(ExpmSkewSym(-wx*(dt - tau)),tau,0,dt);

    matlabFunction(A,'File','PaPaFunc.m');
    matlabFunction(B,'File','PaPdomegaFunc.m');
end

%% Problem Initalization

% Number of monte carlos
N_MC = 100;

% Noise parameters
Sigma_a = 1E-3*eye(3); % Measurement noise for q_inertial2body
Sigma_w = 1E-3*eye(3); % Measurement noise for body angular rate
underweight = 0.5;

% Initial state dispersion parameters
Disp_a = 1E-2*eye(3);
Disp_w = 1E-3*eye(3);

% Vehicle inertia matrix
J = 1E-4*[24181836 3783405 3898808
    3783405 37621803 -1171849
    3898808 -1171849 51576634];

% Actuator noise
Sigma_act = 1E-6*eye(3); % Noise on torque commands to acctuator
w_noise_scale = max(J\diag(Sigma_act));

% Process noise for filter
Q_filter = blkdiag(1E-8*eye(3),0.1*w_noise_scale*eye(3));

% Initial uncertainty for filter
Phat0 = blkdiag(1E-3*eye(3),1E-4*eye(3));
```

```
% Flight software frequency
FSW_freq = 1/10;

% 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
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;
R_change = angle2dcm(3*pi/180,3*pi/180,3*pi/180);
q_change = DCM2Quat(R_change);
% q_LVLH2body_f = [-0.0607, -0.0343, -0.7045, 0.7062]'; % Attitude quaternion at end of
the manuever
q_LVLH2body_f = QuatProduct(q_change,q_LVLH2body_0);

% 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 = 400;
% Tf = 100;
% Tf = 10000;

% Final manuever time
Tf_man = 200;
```

```
% CMG momentum
h0 = 4881;

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


%% Nonlinear controller design

kp_nonlin = 100;
kd_nonlin = 1000;

%% Storage
out_data = cell(N_MC,1);

%% Run MC

for ii = 97:N_MC

    fprintf("MC Iteration: %d / %d \n",ii,N_MC)
    rng(ii+18)

    % Sample randomness
    tsample = 0:FSW_freq:Tf;
    Nsample = length(tsample);
    q_meas_noise = mvnrnd(zeros(3,1),Sigma_a,Nsample);
    w_meas_noise = mvnrnd(zeros(3,1),Sigma_w,Nsample);
    act_noise = mvnrnd(zeros(3,1),Sigma_act,Nsample);
    simin = [];
    simin.act_noise = timeseries(act_noise,tsample);
    simin.q_meas_noise = timeseries(q_meas_noise,tsample);
    simin.w_meas_noise = timeseries(w_meas_noise,tsample);
    init_err = mvnrnd(zeros(6,1),Phat0)';
    dq0 = [0.5*init_err(1:3); 1];
    dq0 = dq0/norm(dq0);

    init_disp = mvnrnd(zeros(6,1),blkdiag(Disp_a,Disp_w))';
    dq0_disp = [0.5*init_disp(1:3); 1];
    dq0_disp = dq0_disp/norm(dq0_disp);

    % Disperse true initial state
    q_inertial2body_0_est = QuatProduct(dq0_disp,q_inertial2body_0);
    w_body_wrt_inertial_0_est = w_body_wrt_inertial_0 + init_disp(4:6);
    q_LVLH2body_0_est = QuatProduct(q_inertial2body_0_est,QuatInv(q_inertial2LVLH_0));

    % Corrupt initial state certainty
    q_inertial2body_0_true = QuatProduct(dq0,q_inertial2body_0_est);
    w_body_wrt_inertial_0_true = w_body_wrt_inertial_0_est + init_err(4:6);

    % Design the maneuver in the LVLH frame
```

```
    % Change in quaternion
    dq_LVLH = QuatProduct(q_LVLH2body_f,QuatInv(q_inertial2body_0_est));

    % 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;

    % Run sim
    out_data{ii} = sim(modelname);
    save("data2.mat","out_data","-v7.3")

end
%% Extract Information From First Run

w_body_inertial = out_data{1}.w;
w_body_inertial_est = out_data{1}.w_body_est;
w_body_inertial_meas = out_data{1}.w_body_meas;
q_inertial2LVLH = out_data{1}.q_inertial2LVLH;
q_inertial2body = out_data{1}.quat;
q_inertial2body_est = out_data{1}.q_inertial2body_est;
q_inertial2body_meas = out_data{1}.q_inertial2body_meas;
CMG_rates = out_data{1}.CMG_rates;
% err_quat = squeeze(out_data{1}.error_quat)';
w_body_inertial_ref = out_data{1}.ref_rate';
CMG_h = squeeze(out_data{1}.CMG_h)';

% Squeeze for some reason
q_inertial2body_est.Data = squeeze(q_inertial2body_est.Data)';
w_body_inertial_est.Data = squeeze(w_body_inertial_est.Data)';
q_inertial2body_meas.Data = squeeze(q_inertial2body_meas.Data)';
% w_body_inertial_meas.Data = squeeze(w_body_inertial_meas.Data)';

% % Find quaternion from body to LVLH
% q_LVLH2body = zeros(size(q_inertial2LVLH));
% for ii = 1:length(tout)
%     q_LVLH2body(:,ii) = QuatProduct(q_inertial2body(:,ii),QuatInv(q_inertial2LVLH(:,↙
ii)));
% end

%% Extract MC Statistics

% Initialize data storage
Ntime = length(q_inertial2body_est.Time);
est_err_quats = zeros(3,Ntime,N_MC);
command_err_quats = zeros(3,Ntime,N_MC);
est_err_w = zeros(3,Ntime,N_MC);
command_err_w = zeros(3,Ntime,N_MC);
mean_est_err_quat = zeros(3,Ntime);
mean_command_err_quat = zeros(3,Ntime);
```

```
var_est_err_quat = zeros(3,Ntime);
disp_command_err_quat = zeros(3,Ntime);
mean_est_var_quat = zeros(3,Ntime);
mean_est_err_w = zeros(3,Ntime);
mean_command_err_w = zeros(3,Ntime);
var_est_err_w = zeros(3,Ntime);
disp_command_err_w = zeros(3,Ntime);
mean_est_var_w = zeros(3,Ntime);
est_cov = zeros(6,6,Ntime,N_MC);
for jj = 1:Ntime
    for ii = 1:N_MC
        q_est = squeeze(out_data{ii}.q_inertial2body_est.Data(:,:,jj));
        q_true = out_data{ii}.quat.Data(jj,:)';
        q_ref = out_data{ii}.ref_quat.Data(jj,:)';
        q_est_err = QuatProduct(QuatInv(q_est),q_true);
        q_com_err = QuatProduct(QuatInv(q_true),q_ref);
        est_err_quats(:,jj,ii) = 2*q_est_err(1:3);
        command_err_quats(:,jj,ii) = 2*q_com_err(1:3);

        w_est = squeeze(out_data{ii}.w_body_est.Data(:,:,jj));
        w_true = out_data{ii}.w.Data(jj,:)';
        w_ref = out_data{ii}.ref_rate.Data(jj,:)';
        est_err_w(:,jj,ii) = w_est - w_true;
        command_err_w(:,jj,ii) = w_true - w_ref;
        est_cov(:,:,jj,ii) = out_data{ii}.est_cov.Data(:,:,jj);
    end

    mean_est_err_quat(:,jj) = mean(squeeze(est_err_quats(:,jj,:)),2);
    mean_est_err_w(:,jj) = mean(squeeze(est_err_w(:,jj,:)),2);
    mean_command_err_quat(:,jj) = mean(squeeze(command_err_quats(:,jj,:)),2);
    mean_command_err_w(:,jj) = mean(squeeze(command_err_w(:,jj,:)),2);

    var_est_err_quat(:,jj) = var(squeeze(est_err_quats(:,jj,:)),0,2)';
    var_est_err_w(:,jj) = var(squeeze(est_err_w(:,jj,:)),0,2)';
    disp_command_err_quat(:,jj) = var(squeeze(command_err_quats(:,jj,:)),0,2)';
    disp_command_err_w(:,jj) = var(squeeze(command_err_w(:,jj,:)),0,2)';

    mean_est_var_quat(:,jj) = diag(mean(est_cov(1:3,1:3,jj,:),4));
    mean_est_var_w(:,jj) = diag(mean(est_cov(4:6,4:6,jj,:),4));


end

save("data2.mat","-v7.3")


%% Plotting Part 3

figure
for ii = 1:3
    subplot(3,1,ii)
```

```
    plot(out_data{1}.error_quat.Time,out_data{1}.error_quat.Data(:,ii),"LineWidth",2)
    xlabel('Time [s]',"Interpreter","latex")
    ylabel("$\delta q$","Interpreter","latex")
    grid on
    title("Estimated Pointing Error")
end

figure
for ii = 1:4
    subplot(4,1,ii)
    hold on
    plot(q_inertial2body.Time, q_inertial2body.Data(:,ii),"LineWidth",2)
    plot(q_inertial2body_est.Time, q_inertial2body_est.Data(:,ii),"LineWidth",2)
%     plot(q_inertial2body_meas.Time, q_inertial2body_meas.Data(:,ii),"LineWidth",2)
    xlabel('Time [s]',"Interpreter","latex")
    ylabel('Quat Element',"Interpreter","latex")
%     legend("Actual","Estimate","Meas")
legend("Actual",'Estimate')
    grid on
end

figure
for ii = 1:3
    subplot(3,1,ii)
    hold on
    plot(w_body_inertial.Time, w_body_inertial.Data(:,ii),"LineWidth",2)
    plot(w_body_inertial_est.Time, w_body_inertial_est.Data(:,ii),"LineWidth",2)
%     plot(w_body_inertial_meas.Time, w_body_inertial_meas.Data(:,ii),"LineWidth",2)
    xlabel('Time [s]',"Interpreter","latex")
    ylabel('Angular Rate',"Interpreter","latex")
%     legend("Actual","Estimate","Meas")
    legend("Actual","Estimate")
    grid on
end

figure
for ii = 1:3
    subplot(3,1,ii)
    hold on
    plot(w_body_inertial.Time, w_body_inertial_ref.Data(:,ii) - w_body_inertial.Data(:,↙
ii),"LineWidth",2)
    xlabel('Time [s]',"Interpreter","latex")
    ylabel('$\delta \omega$ [rad/sec]',"Interpreter","latex")
%     legend("Actual","Estimate")
    grid on
end

figure
for ii = 1:4
    subplot(4,2,2*ii-1)
    plot(CMG_rates.Time,CMG_rates.Data(:,ii))
```

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

    subplot(4,2,2*ii)
    plot(CMG_rates.Time,CMG_rates.Data(:,ii+4))
    xlabel('Time [sec]',"Interpreter","latex")
    ylabel(strcat("$\dot{\beta}$ ",num2str(ii)," [rad/sec]"),"Interpreter",'latex')
    grid on

end

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

figure
for ii = 1:3
    subplot(3,1,ii)
    hold on
    plot(CMG_rates.Time,mean_est_err_quat(ii,:))
    plot(CMG_rates.Time,sqrt(var_est_err_quat(ii,:)))
    plot(CMG_rates.Time,sqrt(mean_est_var_quat(ii,:)))
    ylabel('$a(\delta q)_i$','Interpreter','latex')
    xlabel('Time [sec]')
    grid on
    %title("MC Quat Estimation Performance")
    legend("Mean Error","Est. Err. $1\sigma$","Mean Est. Cov. ↙
$1\sigma$",'Interpreter','latex')

end
saveas(gcf,"MC_quat_est_perf.pdf")

figure
for ii = 1:3
    subplot(3,1,ii)
    hold on
    plot(CMG_rates.Time,mean_est_err_w(ii,:))
    plot(CMG_rates.Time,sqrt(var_est_err_w(ii,:)))
    plot(CMG_rates.Time,sqrt(mean_est_var_w(ii,:)))
    grid on
%     title("MC Body Rate Estimation Performance")
    ylabel('$\omega_i - \hat{\omega}_i$ [rad/sec]','Interpreter','latex')
    xlabel('Time [sec]')
    legend("Mean Error","Est. Err. $1\sigma$","Mean Est. Cov. ↙
```

```
$1\sigma$",'Interpreter','latex')
end
saveas(gcf,"MC_rate_est_perf.pdf")

figure
for ii = 1:3
    subplot(3,1,ii)
    hold on
    plot(CMG_rates.Time,mean_command_err_quat(ii,:))
    plot(CMG_rates.Time,disp_command_err_quat(ii,:))
    title("MC Quat Control Performance")
    legend("Mean Error","Err Disp")


end

figure
for ii = 1:3
    subplot(3,1,ii)
    hold on
    plot(CMG_rates.Time,mean_command_err_w(ii,:))
    plot(CMG_rates.Time,disp_command_err_w(ii,:))
    title("MC Ang Rate Control Performance")
    legend("Mean Error","Err Disp")
end

figure
for ii = 1:3
    subplot(3,1,ii)
    hold on
    for jj = 1:N_MC
        plot(est_err_quats(ii,:,jj))
    end
    title("Quat Est Err Traces")
end

figure
for ii = 1:3
    subplot(3,1,ii)
    hold on
    for jj = 1:N_MC
        plot(command_err_quats(ii,:,jj))

    end
    ylabel('$a^c(\delta q)_i$','Interpreter','latex')
    xlabel('Time [sec]')
    grid on
end
saveas(gcf,"MC_quat_command_perf.pdf")

figure
```

```
for ii = 1:3
    subplot(3,1,ii)
    hold on
    for jj = 1:N_MC
        plot(command_err_w(ii,:,jj))

    end
    ylabel('$\omega^c_i - \hat{\omega}_i [rad/sec]$','Interpreter','latex')
    xlabel('Time [sec]')
    grid on
end
saveas(gcf,"MC_rate_command_perf.pdf")
```