HW 6: Setting Up the Term Project

Table of Contents

Initialize 1
Compute information matrix, normal matrix, x_hat 1

Initialize

```
clearvars -except function_list pub_opt CKF_joseph_init_state CKF_joseph_final_sta
global function_list;
function_list = {};
% close all

stat_od_proj_init
ObsData = load('ObsData.txt');
```

Compute information matrix, normal matrix, x_hat

```
consts.Re = Re;
consts.area = drag.A;
consts.rho = compute_density(ri);
consts.theta_dot = theta_dot;
consts.m = drag.m;
consts.state_len = 18;

P0 = eye(consts.state_len)*1e6;
P0(7,7) = 1e20;

% Things to change for further analysis
sig_meas_up = 0;
sig_meas_dn = 0;

fix_ship = 1;
fix_turk = 0;
fix_gnld = 0;

range_only = 0;
rangerate_only = 0;

cov_down = 1;
cov_up = 0;

if fix_ship
    P0(10:12,10:12) = eye(3)*1e-10;
end
```

```
if fix_turk
    P0(13:15,13:15) = eye(3)*1e-10;
end
if fix_gnld
    P0(16:18,16:18) = eye(3)*1e-10;
end

if cov_down
    P0(1:3,1:3) = eye(3)*10; %sigma = 50 m
    P0(4:6,4:6) = eye(3)*25; %sigma = 50 m
%     P0(7,7) = 1e15;
end
if cov_up
    P0(1:3,1:3) = eye(3)*1e12; %sigma = 50 m
    P0(4:6,4:6) = eye(3)*1e12; %sigma = 50 m
%     P0(7,7) = 1e15;
end

x0_ap = zeros(consts.state_len,1);

sig_range = 0.01; % m
sig_rangerate = 0.001; %m/s
if sig_meas_up
    sig_range = sig_range * 10;
    sig_rangerate = sig_rangerate * 10
elseif sig_meas_dn
    sig_range = sig_range / 10;
    sig_rangerate = sig_rangerate / 10;
end
W = [1/(sig_range*sig_range) 0; 0 1/(sig_rangerate*sig_rangerate)];


dt = 0.1;
times = 0:dt:18340;
ode_opts = odeset('RelTol', 1e-12, 'AbsTol', 1e-20);

pfr_plots = figure;
if range_only
    W = 1/(sig_range*sig_range);
elseif rangerate_only
    W = 1/(sig_rangerate*sig_rangerate);
end

[num_obs, ~] = size(ObsData);
y_store = zeros(2,num_obs);
for iter = 1:3
[T,X] = ode45(@two_body_state_dot, times, state, ode_opts, propagator_opts);

% Store off every 20 seconds of data
X_store = X(mod(times,20) == 0,:);
T_store = T(mod(times,20) == 0);

% Accumulate the information and normal matrices
% info_mat = zeros(consts.state_len);
```

```
chol_P0 = chol(P0,'lower');
P0_inv = eye(18)/(chol_P0')/(chol_P0);
info_mat = P0_inv;
norm_mat = P0_inv*x0_ap;
% H_tilda_given = load('BatchHtilda.mat');
cntr =1 ;
% Obs. deviation
y1 = zeros(num_obs,1);
y2 = zeros(num_obs,1);
for ii = 1:num_obs
    site_num = 0;
    for jj = 1:3
        if ObsData(ii, 2) == site(jj).id
            site_num = jj;
            break
        end
    end
    t_obs = ObsData(ii,1);
    ostate = X(T(:,1)==t_obs,1:6);

    r_comp = compute_range_ECFsite(ostate(1:3),...
        site(site_num).r,theta_dot*t_obs);
    rr_comp = compute_range_rate_ECFsite(ostate(1:6),...
        site(site_num).r,theta_dot*t_obs, theta_dot);

    y1(ii) = (ObsData(ii,3)-r_comp);
    y2(ii) = (ObsData(ii,4)-rr_comp);

end

RMS_accum = 0;
num_RMS_meas = 0;
H_store = zeros(2,consts.state_len,num_obs);
pfr_store = zeros(2,num_obs);
if range_only || rangerate_only
    H_store = zeros(1,consts.state_len,num_obs);
    pfr_store = zeros(1,num_obs);
end
obs_time_store = zeros(num_obs,1);


% Begin batch
for ii = 1:num_obs
    obs_time = ObsData(ii,1);
    obs_site = ObsData(ii,2);
    obs_time_store(ii) = obs_time;
    % STM
    STM = eye(consts.state_len);
    STM(1:important_block(1),1:important_block(2)) = ...
        reshape(X_store(T_store == obs_time,consts.state_len+1:end), ...
        important_block(1), important_block(2));

    % H~
    consts.t = obs_time;
```

```
        for xx = 1:3
            if site(xx).id == obs_site
                consts.site = xx;
                break
            end
        end
        state_at_obs = X_store(T_store == obs_time,1:consts.state_len);
        H_tilda = stat_od_proj_H_tilda(state_at_obs, consts);
        if range_only
            H_tilda(2,:) = [];
        elseif rangerate_only
            H_tilda(1,:) = [];
        end

        %H
        H = H_tilda*STM;
        H_store(:,:,ii) = H;

        % Accumulate information matrix
        info_mat = info_mat + H'*W*H;

        % Accumulate normal matrix
        y = [y1(ii); y2(ii)];
        if range_only
            y = y1(ii);
        elseif rangerate_only
            y = y2(ii); %#ok<*UNRCH>
        end
        norm_mat = norm_mat + H'*W*y;

        if iter == 1
            y_store(:,ii) = y;
        end
    end
    x_est = cholesky_linear_solver(info_mat,norm_mat)

    % RMS
    for ii = 1:num_obs
        y = [y1(ii); y2(ii)];
        if range_only
            y = y1(ii);
        elseif rangerate_only
            y = y2(ii); %#ok<*UNRCH>
        end
        H = H_store(:,:,ii);
        postfit_res = y - H*x_est;
        pfr_store(:,ii) = postfit_res;
        RMS_accum = RMS_accum + postfit_res'*W*postfit_res;
    end
    RMS = sqrt(RMS_accum/num_obs)
    if ~range_only
        rangerate_RMS = sqrt(sum(pfr_store(1,:).*pfr_store(1,:))/num_obs)
    end
    if ~rangerate_only
        range_RMS = sqrt(sum(pfr_store(1,:).*pfr_store(1,:))/num_obs)
    end
    if ~range_only && ~rangerate_only
        rangerate_RMS = sqrt(sum(pfr_store(2,:).*pfr_store(2,:))/num_obs)
        range_RMS = sqrt(sum(pfr_store(1,:).*pfr_store(1,:))/num_obs)
    end

    x0_ap = x0_ap-x_est;
    state(1:18) = state(1:18) + x_est;

    if ~range_only && ~rangerate_only
        figure(pfr_plots);
        subplot(3,2,iter*2-1)
        plot(1:num_obs,pfr_store(1,:))
        hold on
        plot(1:num_obs,3*sig_range*ones(1,num_obs),'r--')
        plot(1:num_obs,-3*sig_range*ones(1,num_obs),'r--')
        ylabel('m')
        title({sprintf('Pass %d',iter),...
            sprintf('Range RMS = %.4e (m)',range_RMS)})
        subplot(3,2,iter*2)
        plot(1:num_obs,pfr_store(2,:))
        hold on
        plot(1:num_obs,3*sig_rangerate*ones(1,num_obs),'r--')
        plot(1:num_obs,-3*sig_rangerate*ones(1,num_obs),'r--')
        ylabel('m/s')
        title({sprintf('Pass %d',iter),...
            sprintf('RR RMS = %.4e (m/s)',rangerate_RMS)})
    else

        if rangerate_only
            units = 'm/s';
            my_title = 'Range-Rate';
            this_RMS = rangerate_RMS;
            this_sig = sig_rangerate;
        else
            units = 'm';
            my_title = 'Range';
            this_RMS = range_RMS;
            this_sig = sig_range;
        end
        figure(pfr_plots);
        subplot(3,1,iter)
        plot(1:num_obs,pfr_store(1,:))
        hold on
        plot(1:num_obs,3*this_sig*ones(1,num_obs),'r--')
        plot(1:num_obs,-3*this_sig*ones(1,num_obs),'r--')
        ylabel(units)
        title(sprintf('Pass %d: %s RMS = %.4e %s',...
            iter, my_title, this_RMS, units))
    end

    end
```

```
if ~range_only && ~rangerate_only
    figure(pfr_plots);
    subplot(3,2,1)
%     title('Range Postfit Residuals')
    subplot(3,2,2)
%     title('Range-Rate Postfit Residuals')
    subplot(3,2,5)
    xlabel('Observation')
    subplot(3,2,6)
    xlabel('Observation')
else
    if rangerate_only
    end
    figure(pfr_plots);
    subplot(3,1,1)
    subplot(3,1,3)
    xlabel('Observation')
end

figure
subplot(2,1,1)
plot(1:num_obs,y_store(1,:),'.')
title('Prefit Residuals')
ylabel('Range (m)')
subplot(2,1,2)
plot(1:num_obs,y_store(2,:),'.')
xlabel('Observation'),ylabel('Range Rate (m/s)')
%
if ~range_only && ~rangerate_only && fix_ship && ~fix_gnld && ~fix_turk
X_est_batch = x_est + state(1:consts.state_len);
X_est_final_batch = STM*x_est + X_store(end,1:consts.state_len)';
elseif range_only
X_est_batch_r = x_est + state(1:consts.state_len);
X_est_final_batch_r = STM*x_est + X_store(end,1:consts.state_len)';
elseif rangerate_only
X_est_batch_rr = x_est + state(1:consts.state_len);
X_est_final_batch_rr = STM*x_est + X_store(end,1:consts.state_len)';
elseif fix_turk
X_est_batch_tfix = x_est + state(1:consts.state_len);
X_est_final_batch_tfix = STM*x_est + X_store(end,1:consts.state_len)';
elseif fix_gnld
X_est_batch_gfix = x_est + state(1:consts.state_len);
X_est_final_batch_gfix = STM*x_est + X_store(end,1:consts.state_len)';
elseif ~fix_ship
X_est_batch_nfix = x_est + state(1:consts.state_len);
X_est_final_batch_nfix = STM*x_est + X_store(end,1:consts.state_len)';
end



chol_info_mat = chol(info_mat,'lower');
P0_est_batch = eye(18)/(chol_info_mat')/(chol_info_mat);
% The last STM in the loop is the 0,t_end one.
P_final_batch = STM*P0_est_batch*STM';
```

```
fprintf('Final Covariance:\n')
disp(P_final_batch(1:6,1:6))

% Ellipsoid
[U,Pprime] = eigs(P_final_batch(1:3,1:3));
Semi = [sqrt(9*Pprime(1,1)) sqrt(9*Pprime(2,2)) sqrt(9*Pprime(3,3))];
figure
subplot(1,2,1)
plotEllipsoid(U,Semi);
% title('3-sigma Position Error Probability Ellipsoid - Batch')
xlabel('x (m)'),ylabel('y (m)'),zlabel('z (m)')
subplot(1,2,2)
plotEllipsoid(U,Semi);
% title('3-sigma Position Error Probability Ellipsoid - Batch')
xlabel('x (m)'),ylabel('y (m)'),zlabel('z (m)')
```

Published with MATLAB® R2013b