# ASEN 6080 HW 7

- Ian Faber, 108577813

1. a. Implement the sequential consider covariance analysis algorithm, where

$$P_{xx,0} = 10^4 \cdot I_{6\times6}$$

$$c^* = 0, \quad c = c_{true} - c^* = J_3, \text{ and}$$

$$P_{cc,0} = J_3^2$$

See PDF for code!

b. Plot $P_{x,i}$ and $P_{xx,i}$ with $2\sigma$ covariance envelopes and include RMS errors

See PDF for plots!

Component-wise RMS, no consider:

$[3.429, 9.436\times10^{-1}, 7.763, 5.208\times10^{-2}, 1.613\times10^{-2}, 8.46\times10^{-2}]$

3D RMS, no consider: 8.539

Component RMS, with consider:

$[3.548, 9.754\times10^{-1}, 7.953, 5.158\times10^{-2}, 1.601\times10^{-2}, 8.397\times10^{-2}]$

3D RMS, with consider: 8.784

## Table of Contents



```
function filterOut = CFA(Xstar0, stations, pConst, P0, Pcc0, x0, S0, numMeas)

% Function that implements the Consider Filter Algorithm for stat OD
% problems
%   Inputs:
%       - Xstar0: Initial value of the reference trajectory, organized as
%                 follows:
%                 [X0; Y0; Z0; Xdot0; Ydot0; Zdot0]
%       - stations: Stations struct as defined by makeStations.m. Must have
%                   propagated station states! To propagate states, see
%                   generateTruthData.m.
%       - pConst: Planetary constant structure as formatted by
%                 getPlanetConst.m
%       - P0: Initial state covariance estimate
%       - Pcc0: Initial consider parameter covariance estimate
%       - x0: Initial state deviation estimate
%       - S0: Initial sensitivity analysis
%       - numMeas: Number of measurements to process (optional). If not
%                  specified, defaults to all measurements
%   Outputs:
%       - filterOut: Filter output structure with the following fields:
%           - xEst: Estimated state deviation at each time processed from
%                   the station measurements in "stations" (t), organized
%                   as follows:
%                   [xEst_1, xEst_2, ..., x_t], where
%                   xEst = [x; y; z; xDot; yDot; zDot]
%           - xcEst: Estimated state deviation at each time processed from
%                    the station measurements in "stations" (t), including
%                    consider parameter effects, organized as follows:
%                    [xcEst_1, xcEst_2, ..., xc_t], where
%                    xcEst = [xc; yc; zc; xcDot; ycDot; zcDot]
%           - PEst: Estimated state covariance at each time in t, organized
%                   as follows:
%                   [{P_1}, {P_2}, ..., {P_t}]
%           - PcEst: Estimated state covariance at each time in t,
%                    including consider parameter effects, organized as
%                    follows:
%                    [{Pc_1}, {Pc_2}, ..., {Pc_t}]
%           - PxcEst: Estimated state to consider parameter covariance at
%                     each time in t, organized as follows:
%                     [{Pxc_1}, {Pxc_2}, ..., {Pxc_t}]
%           - prefit_res: Pre-fit residuals (y_i) at each time in t:
%                         [y_1, y_2, ..., y_t]
```

```
%           - postfit_res: Postfit residuals (epsilon = y_i - Htilde_i*x_i)
%                          at each time in t:
%                          [epsilon_1, epsilon_2, ..., epsilon_t]
%           - postfit_res_c: Postfit residuals (epsilon_c = y_i -
Htilde_i*x_ci)
%                            at each time in t:
%                            [epsilon_c1, epsilon_c2, ..., epsilon_ct]
%           - t: Measurement time vector for the LKF filter
%           - statVis: Station visibility vector
%           - XEst: Estimated full state at each time in t:
%                   [XEst_1, XEst_2, ..., XEst_t], where
%                   XEst = [X; Y; Z; XDot; YDot; ZDot]
%           - XcEst: Estimated full state at each time in t, including
%                    consider parameter effects:
%                    [XcEst_1, XcEst_2, ..., XcEst_t], where
%                    XcEst = [Xc; Yc; Zc; XcDot; YcDot; ZcDot]
%           - Phi_total: Integrated STM from t0 to t_i for each time in t:
%                        [{Phi_1}, {Phi_2}, ..., {Phi_t}], where
%                        Phi_t = Phi(t_i, t_0)
%           - Psi: Integrated consider parameter and state mapping matrix
%                  from t_0 to t_i for each time in t:
%                  [{Psi_1}, {Psi_2}, ..., {Psi_t}], where
%                  Psi_t = Psi(t_i, t_0)
%
%   By: Ian Faber, 04/05/2025
%
```

# Initialize settings

```
    % Format ode45 and sizes
opt = odeset('RelTol',1e-12,'AbsTol',1e-12);
n = length(Xstar0);
xEst = [];
xcEst = [];
PEst = [];
PcEst = [];
PxcEst = [];
prefit_res = [];
postfit_res = [];
postfit_res_c = [];
XEst = [];
XcEst = [];
Phi_total = [];
Theta_total = [];
Psi = [];
```

# Process station data into a usable form

```
[t, Y, R, Xs, vis] = processStations(stations);

    % Specify number of measurements to process
if ~exist("numMeas", 'var')
```

```
    numMeas = length(Y);
end
```

# Loop through each observation

```
t_im1 = t(1);
Xstar_im1 = Xstar0;
x_im1 = x0;
P_im1 = P0;
S_im1 = S0;
Phi_full = eye(n);
Theta_full = zeros(6,1);

[~, XPhi_test] =
ode45(@(t,XPhi)STMEOM_J2(t,XPhi,pConst.mu,pConst.J2,pConst.Ri), [t(1),
t(end)], [Xstar0; reshape(eye(n),n^2,1)], opt);
Phi_test = reshape(XPhi_test(end,n+1:end), n, n);

[~, XTheta_test] = ode45(@(t,XTheta)STMEOM_CP_J3(t,XTheta,pConst), [t(1),
t(end)], [Xstar0; zeros(n,1)], opt);
Theta_test = reshape(XTheta_test(end,n+1:end), n, 1);

for k = 2:numMeas

        % Read next time, measurement, and measurement covariance
    t_i = t(k);
    Y_i = Y{k};
    R_i = R{k};

        % Continue to integrate Phi(t, t0) and Theta(t, t0) for iteration
purposes
    XPhi_full = [Xstar_im1; reshape(Phi_full,n^2,1)];
    [~, XPhi_full] = ode45(@(t,XPhi)STMEOM_J2(t,XPhi,pConst.mu, pConst.J2,
pConst.Ri), [t_im1 t_i], XPhi_full, opt);
    Phi_full = reshape(XPhi_full(end,n+1:end), n, n);

    XTheta_full = [Xstar_im1; reshape(Theta_full,n,1)];
    [~, XTheta_full] = ode45(@(t,XTheta)STMEOM_CP_J3(t,XTheta,pConst),
[t_im1 t_i], XTheta_full, opt);
    Theta_full = reshape(XTheta_full(end,n+1:end), n, 1);

        % Integrate Xstar, Phi, and Theta from t_im1 to t_i
    Phi_im1 = eye(n);
    XPhi_im1 = [Xstar_im1; reshape(Phi_im1,n^2,1)];
    [~, XPhi_i] = ode45(@(t,XPhi)STMEOM_J2(t,XPhi,pConst.mu, pConst.J2,
pConst.Ri), [t_im1 t_i], XPhi_im1, opt);
    Xstar_i = XPhi_i(end,1:n)';
    Phi_i = reshape(XPhi_i(end,n+1:end),size(Phi_im1));

    Theta_im1 = zeros(n,1);
    XTheta_im1 = [Xstar_im1; reshape(Theta_im1,n,1)];
    [~, XTheta_i] = ode45(@(t,XTheta)STMEOM_CP_J3(t,XTheta,pConst), [t_im1
t_i], XTheta_im1, opt);
```

```
    Theta_i = reshape(XTheta_i(end,n+1:end),size(Theta_im1));

        % Build Psi(t, t_0)
    Psi_i = [
                Phi_full, Theta_full;
                zeros(1,n), eye(1)
            ];

        % Time update
    x_i = Phi_i*x_im1;
    P_i = Phi_i*P_im1*Phi_i';
    S_i = Phi_i*S_im1 + Theta_i;
    P_ci = P_i + S_i*Pcc0*S_i'; % aka Pxx^-
    P_xci = S_i*Pcc0;

        % Get number of measurements in Y, station states, and station
        % visibility at this time
    meas = length(Y_i)/2; % Assuming 2 data points per measurement: range
and range-rate
    Xstat = Xs{k}'; % Extract station state(s) at the time of measurement
    statVis = vis{k}; % Extract the stations that were visible at the time
of measurement

        % Build y_i
    yExp = [];
    for kk = 1:meas
        genMeas = generateRngRngRate(Xstar_i, Xstat(:,meas),
stations(statVis(kk)).elMask, true); % Ignore elevation mask
        yExp = [yExp; genMeas(1:2)];
    end

    y_i = Y_i - yExp;

        % Build Htilde_i
    Htilde_x_i = [];
    Htilde_c_i = [];
    for kk = 1:meas
        [Htilde_x, Htilde_c] = MeasurementPartials_CP_J3_sc(Xstar_i,
Xstat(:,meas));
        Htilde_x_i = [Htilde_x_i; Htilde_x];
        Htilde_c_i = [Htilde_c_i; Htilde_c];
    end

        % Build K_i
    K_i = P_i*Htilde_x_i'*(Htilde_x_i*P_i*Htilde_x_i' + R_i)^-1;
    K_ci = P_ci*Htilde_x_i'*(Htilde_x_i*P_ci*Htilde_x_i' + R_i)^-1;

        % Measurement update
    xBar_i = x_i;
    x_i = x_i + K_i*(y_i - Htilde_x_i*x_i);
    S_i = (eye(n) - K_i*Htilde_x_i)*S_i - K_i*Htilde_c_i;
    x_ci = xBar_i + K_ci*(y_i - Htilde_x_i*x_i);

    mat = K_i*Htilde_x_i; % Intermediate matrix for sizing
```

```
    P_i = (eye(size(mat)) - mat)*P_i*(eye(size(mat)) - mat)' + K_i*R_i*K_i';
    P_ci = P_i + S_i*Pcc0'*S_i'; %% AKA Pxx^+
    P_xci = S_i*Pcc0;

        % Accumulate data to save
    xEst = [xEst, x_i];
    xcEst = [xcEst, x_ci];
    PEst = [PEst, {P_i}];
    PcEst = [PcEst, {P_ci}];
    PxcEst = [PxcEst, {P_xci}];
    prefit_res = [prefit_res, y_i];
    postfit_res = [postfit_res, y_i - Htilde_x_i*x_i];
    postfit_res_c = [postfit_res_c, y_i - Htilde_x_i*x_ci];
    XEst = [XEst, Xstar_i + x_i];
    XcEst = [XcEst, Xstar_i + x_ci];
    Phi_total = [Phi_total, {Phi_full}];
    Theta_total = [Theta_total, {Theta_full}];
    Psi = [Psi, {Psi_i}];

        % Update for next run
    t_im1 = t_i;
    Xstar_im1 = Xstar_i;
    P_im1 = P_i;
    x_im1 = x_i;
    S_im1 = S_i;

end
```

# Assign outputs

```
filterOut.xEst = xEst;
filterOut.xcEst = xcEst;
filterOut.PEst = PEst;
filterOut.PcEst = PcEst;
filterOut.PxcEst = PxcEst;
filterOut.prefit_res = prefit_res;
filterOut.postfit_res = postfit_res;
filterOut.postfit_res_c = postfit_res_c;
filterOut.t = t(2:end); % t_0 not included in estimate
filterOut.statVis = vis;
filterOut.XEst = XEst;
filterOut.XcEst = XcEst;
filterOut.Phi_total = Phi_total;
filterOut.Phi_test = Phi_test;
filterOut.Theta_total = Theta_total;
filterOut.Theta_test = Theta_test;
filterOut.Psi = Psi;

end
```

*Published with MATLAB® R2023b*

c. Comment on the results, which covariance is more realistic? What percentage of errors lie outside the 2σ envelopes?

The state error RMS is almost identical between including and ignoring consider parameter effects. However, as expected, the real difference appears in the respective 2σ covariance envelopes.

When ignoring consider effects, almost all of the state errors lie outside the 2σ bounds, which isn't very realistic and suggests that the filter is smug.

On the other hand, when including consider effects, nearly all of the state errors lie inside the 2σ bounds, which is more realistic and suggests that the filter is still taking measurements into account. Further, what was once seen as a biased state estimate is now comfortably inside our covariance bounds.

# ASEN 6080 HW 7 Problem 1 Main Script

## Table of Contents

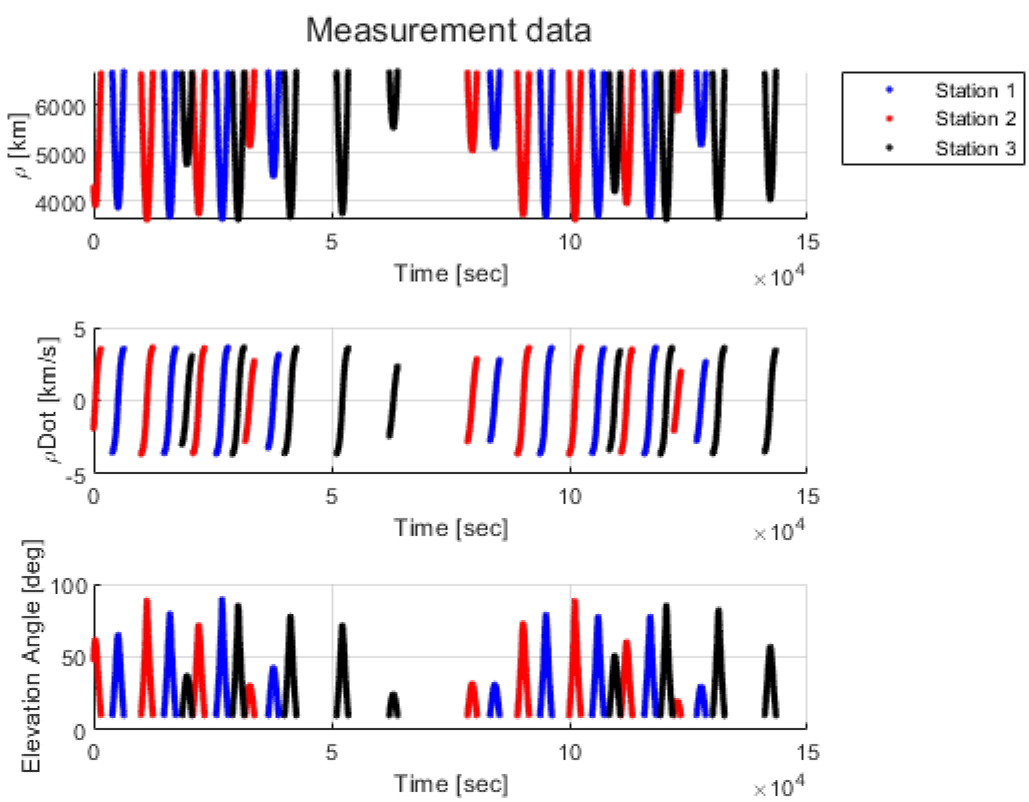


By: Ian Faber

# Housekeeping

# Setup

# Make truth data



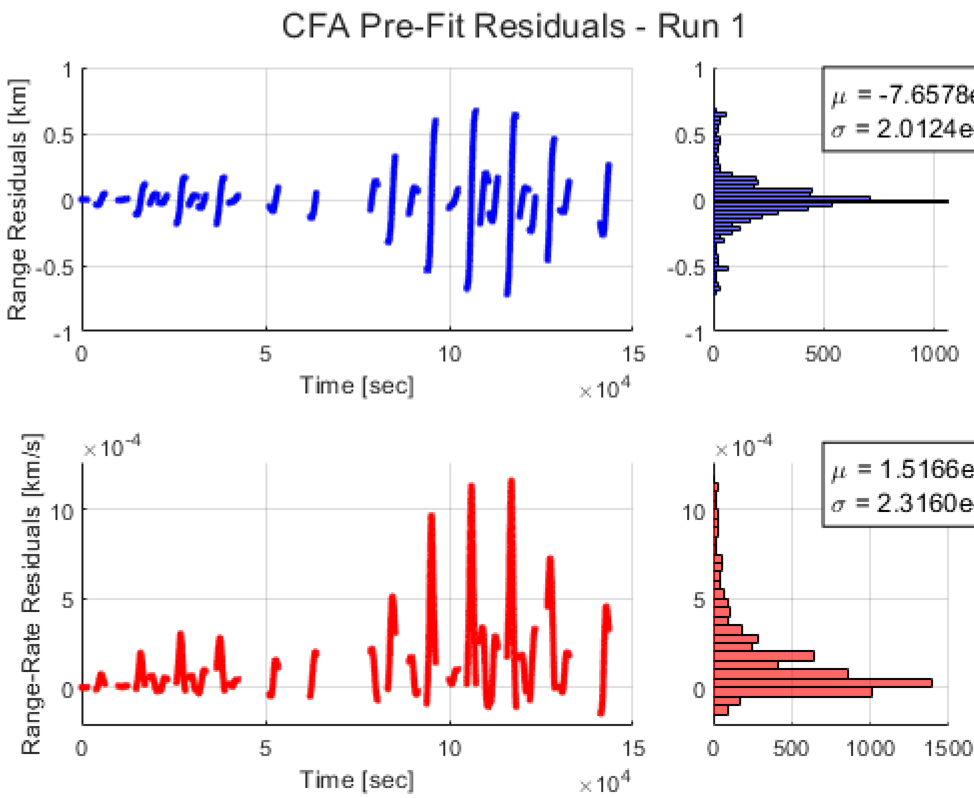
# Problem 1a. Filter setup

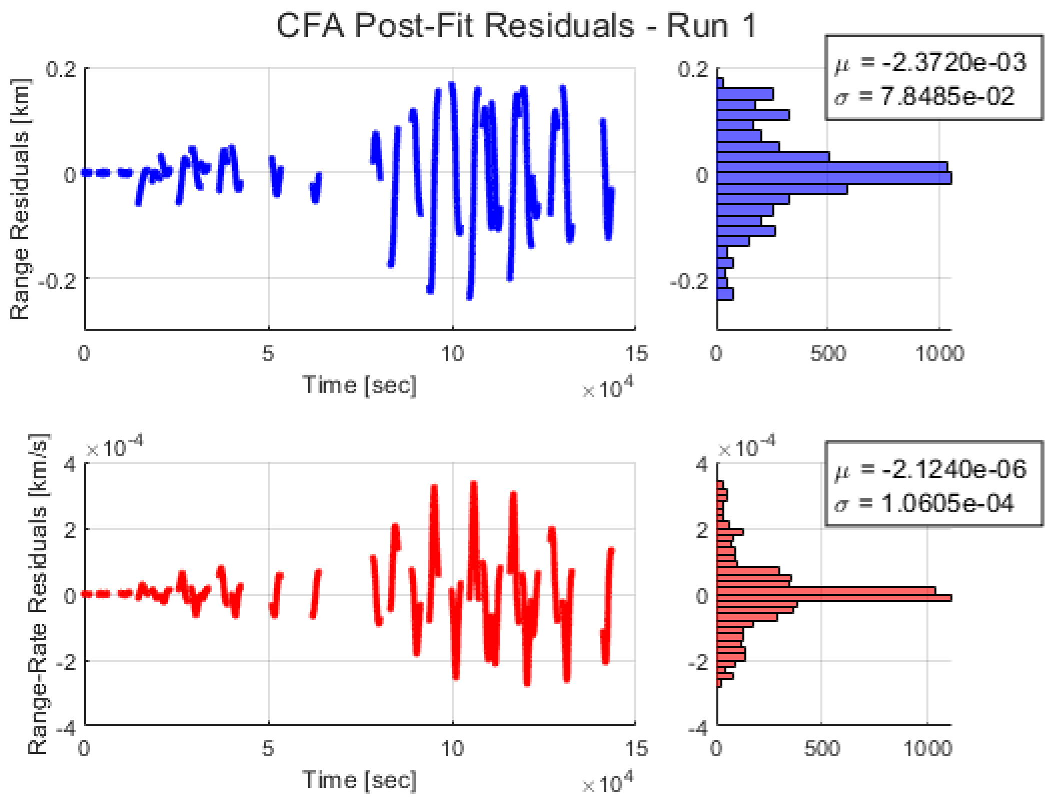
Covariance matrices

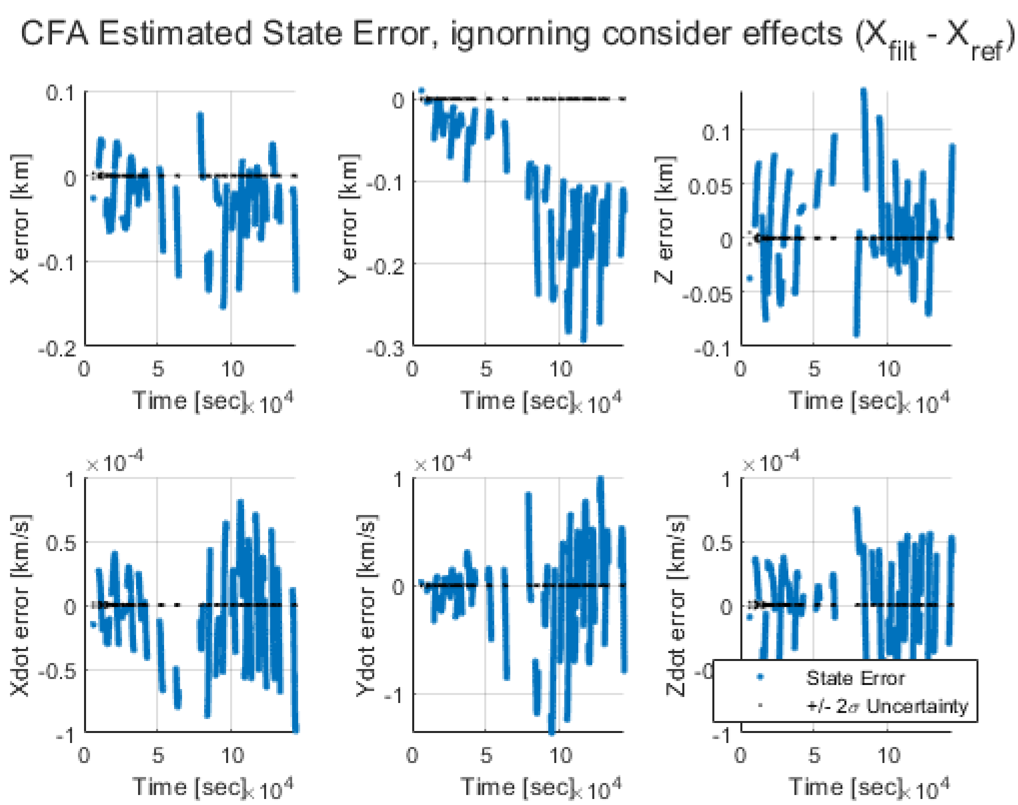
# Problem 1b/c. Run Consider Covariance filter

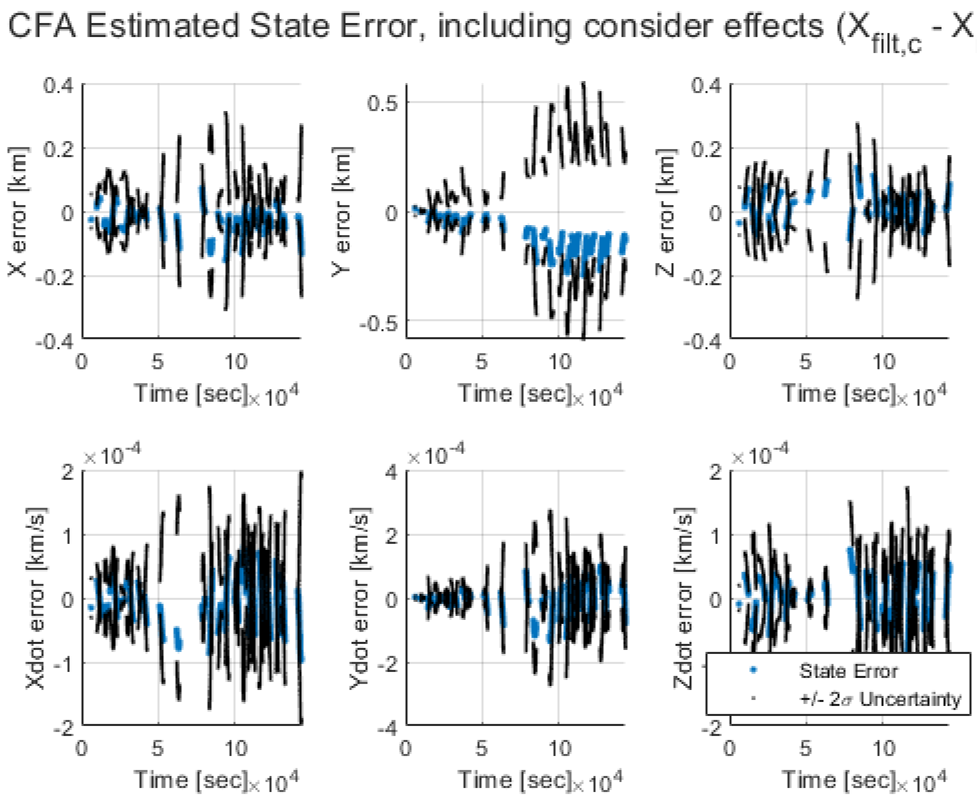
```
Running CFA on data

    Running CFA:
Prefit RMS: 242.0526, Postfit RMS: 93.3103. Hit max CFA iterations. Runs so
far: 1
Final prefit RMS: 242.0526
Final postfit RMS: 93.3103
State Error Component-wise RMS, no consider effects: [5.743e+00, 1.681e+00,
1.079e+01, 2.163e-01, 6.702e-02, 3.521e-01]
State Error 3D RMS, no consider effects: 1.234e+01
State Error Component-wise RMS, including consider effects: [7.548e+00,
2.253e+00, 1.341e+01, 2.182e-01, 6.766e-02, 3.553e-01]
State Error 3D RMS, including consider effects: 1.556e+01
```




CFA Post-Fit Residuals - Run 1
$\mu$ = -2.3720e-03
$\sigma$ = 7.8485e-02
Range Residuals [km]
Time [sec]
$\mu$ = -2.1240e-06
$\sigma$ = 1.0605e-04
Range-Rate Residuals [km/s]
Time [sec]


CFA Estimated State Error, ignorning consider effects ($X_{filt}$ - $X_{ref}$)
X error [km]
Y error [km]
Z error [km]
Xdot error [km/s]
Ydot error [km/s]
Zdot error [km/s]
Time [sec]
State Error
+/- 2$\sigma$ Uncertainty




# Problem 1d/e. Map xHat_f and Pc,f to t0 and propagate again

*Backpropagating CFA solution*

*State Error Component-wise RMS, backpropagated: [5.121e-02, 1.364e-01, 3.918e-02, 3.882e-05, 4.705e-05, 3.237e-05]*
*State Error 3D RMS, backpropagated: 1.509e-01*

d. Map $\Delta\hat{x}_{k,f}$ and $P_{k,f}$ from $t_f$ to $t_0$, and then integrate the results forward to resemble a batch solution.

See PDF for plots!

e. Plot the state errors with $2\sigma$ covariance envelopes, and include RMS errors.

See PDF for plots!

Component-wise RMS:

$[5.115\times10^{-2}, 1.364\times10^{-1}, 3.912\times10^{-2}, 3.879\times10^{-5}, 4.704\times10^{-5}, 3.234\times10^{-5}]$

3D RMS: $1.504\times10^{-1}$

f. Compare the state errors to the covariance envelopes, how do they match?

The state errors are nearly all within the covariance envelopes, and the shapes match nearly identically!

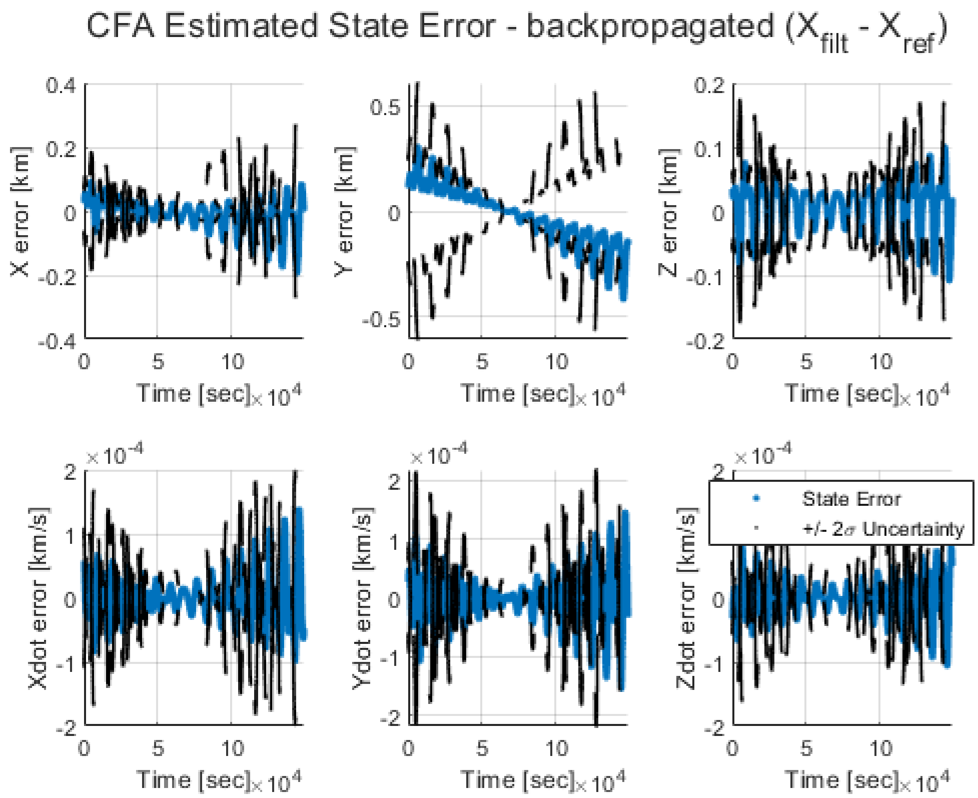
There are intervals when the envelopes get very narrow where the state errors sometimes exit the envelope, but these are rare.

9. Based on these results, what conclusions can be made on deciding whether to include $J_3$ in the filter dynamics?

Even when including consider parameter effects, we can see that our uncertainty steadily grows with time. Thus, eventually the uncertainty grows so large that we can no longer confidently tell where the spacecraft is, which may violate requirements. Thus, to avoid this we should include $J_3$ in our filter dynamics.

If this scenario were a flyby of Earth, then the influence of $J_3$ won't have as much time to manifest in our state estimate, and hence the uncertainty won't be as inflated. Thus, we could ignore $J_3$ feasibly.

As always, it depends on the scenario, and what happens to our covariance.




*Published with MATLAB® R2023b*