ASEN 6080 HW 6

-Ian Faber, 108577813

1. a. Implement a UKF, including only μ + J_2 in the dynamic model.

See PDF for code!

b/c. Use the UKF to process the same data from HW 3 under the following cases, and compare to the EKF with SNC from HW 3!

To compare, I'm using an EKF initialized with 50 LKF measurements.

i. $\alpha = 1.0$, $\beta = 2$, no process noise

See PDF for plots

In this case, the UKF and EKF with SNC perform nearly identically. In particular, both exhibit a bias in state errors near the end of the time span, and their pre and postfit residuals have very similar mean and standard deviation.

UKF pre/postfit RMS: 93.1557

EKF pre/postfit RMS: 93.8570

Table of Contents


```
function filterOut = UKF(stations, pConst, X0, P0, Q0, alpha, beta,
includeJ3)

% Function that implements a UKF for stat OD problems
%   Inputs:
%       - 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
%       - X0: Initial full state estimate
%       - P0: Initial state covariance estimate
%       - Q0: Initial process noise covariance matrix
%       - alpha: UKF sigma point spacing variable from [1e-4, 1]
%       - beta: UKF probability distribution variable, generally 2 for
%               Gaussian probability distributions
%       - includeJ3: Boolean indicating whether the filter dynamics should
%                    include J3 in addition to mu and J2
%   Outputs:
%       - filterOut: Output filter structure with the following fields:
%           - XEst: Estimated full state vector at each time in t:
%                   [XEst_1, XEst_2, ..., XEst_t], where
%                   XEst = [X; Y; Z; XDot; YDot; ZDot]
%           - PEst: Estimated state covariance at each time in t, organized
%                   as follows:
%                   [{P_1}, {P_2}, ..., {P_t}]
%           - prefit_res: Pre-fit residuals (y_i - yBar_i) at each time in t:
%                         [prefit_1, prefit_2, ..., prefit_t]
%           - postfit_res: Post-fit residuals (y_i - yBar_i after XEst has
%                          been computed) at each time in t:
%                          [postfit_1, postfit_2, ..., postfit_t]
%           - t: Measurement time vector for the EKF filter
%           - statVis: Station visibility vector
%
%   By: Ian Faber, 03/15/2025
%
```

Initialize settings

Format ode45 and sizes

```
opt = odeset('RelTol',1e-12,'AbsTol',1e-12);
L = length(X0);
XEst = [];
PEst = [];
prefit_res = [];
postfit_res = [];
```

Define helper functions

```
GammaFunc = @(dt) [(dt/2)*eye(3); eye(3)];
J2Func = @(t,X)orbitEOM_MuJ2(t,X,pConst.mu,pConst.J2,pConst.Ri);
J3Func = @(t,X)orbitEOM_MuJ2J3(t,X,pConst.mu,pConst.J2,pConst.J3,pConst.Ri);
```

Process station data into a usable form

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

Precompute UKF weights

```
kappa = 3 - L;
lambda = (alpha^2)*(L + kappa) - L;
gamma = sqrt(L + lambda);

W_0m = lambda/(L + lambda);
W_0c = lambda/(L + lambda) + (1 - alpha^2 + beta);
W_im = [W_0m, (1/(2*(L + lambda)))*ones(1,2*L)];
W_ic = [W_0c, (1/(2*(L + lambda)))*ones(1,2*L)];

if alpha == 1e-4
    alpha;
end
```

Loop through all observations

Initialize UKF variables

```
X_im1 = X0;
P_im1 = P0;
t_im1 = t(1);

    % Loop through each observation
for k = 2:length(Y)
        % Read next time, measurement, and measurement covariance
    t_i = t(k);
    Y_i = Y{k};
    R_i = R{k};

        % Create Q if necessary for process noise
    dT = t_i - t_im1;
    if any(any(Q0 > 0) & (dT <= 10)) % Process noise exists and the time gap
isn't too big for assumptions to break
```

```
        Gamma_i = GammaFunc(dT);
        Q = Gamma_i*Q0*Gamma_i';
    else
        Q = zeros(L);
    end

        % Calculate previous sigma points
    sqrtP_im1 = sqrtm(P_im1); % Used to be chol()
    Chi_im1 = [X_im1, X_im1 + gamma*sqrtP_im1, X_im1 - gamma*sqrtP_im1]; % L
x (2L + 1) matrix

        % Propagate previous sigma points through dynamics
    ChiVec_im1 = reshape(Chi_im1, L*(2*L+1), 1);
    tspan = [t_im1, t_i];
    if ~includeJ3 % Only include mu and J2
        [~,ChiVec] = ode45(@(t,ChiVec)sigPointEOM(t,ChiVec,J2Func), tspan,
ChiVec_im1, opt);
    else % Include mu, J2, and J3
        [~,ChiVec] = ode45(@(t,ChiVec)sigPointEOM(t,ChiVec,J3Func), tspan,
ChiVec_im1, opt);
    end
    Chi_i = reshape(ChiVec(end,:), L, 2*L + 1);

        % Time update
    X_i = 0;
    for kk = 1:2*L+1
        X_i = X_i + W_im(kk)*Chi_i(:,kk);
    end

    P_i = Q;
    for kk = 1:2*L+1
        P_i = P_i + W_ic(kk)*(Chi_i(:,kk) - X_i)*(Chi_i(:,kk) - X_i)';
    end

        % Recompute sigma points to account for propagation and process
        % noise
    sqrtP_i = sqrtm(P_i); % Used to be chol()
    Chi_i = [X_i, X_i + gamma*sqrtP_i, X_i - gamma*sqrtP_i];

        % 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

        % Construct yBar_i
    yBar_i = 0;
    YExp = [];
    for kk = 1:2*L + 1
        yExp = [];
        state = Chi_i(:,kk);
        for idx = 1:meas % Account for multiple stations visible at the same
```

```
time
            genMeas = generateRngRngRate(state, Xstat(:,idx),
stations(statVis(idx)).elMask, true); % Ignore elevation mask
            yExp = genMeas(1:2);
            % YExp = [YExp, genMeas];
            YExp = [YExp, yExp];
            yBar_i = yBar_i + W_im(kk)*yExp;
        end
    end

        % Compute innovation and cross covariances
    Pyy = R_i;
    Pxy = zeros(L,2);
    for kk = 1:2*L + 1
        Pyy = Pyy + W_ic(kk)*(YExp(:,kk) - yBar_i)*(YExp(:,kk) - yBar_i)';
        Pxy = Pxy + W_ic(kk)*(Chi_i(:,kk) - X_i)*(YExp(:,kk) - yBar_i)';
    end

        % Compute Kalman Gain
    K_i = Pxy*(Pyy^-1);

        % Measurement update
    X_i = X_i + K_i*(Y_i - yBar_i);
    P_i = P_i - K_i*Pyy*K_i';

        % Calculate expected measurement after measurement update for
        % postfits
    genMeas_post = generateRngRngRate(X_i, Xstat(:,1),
stations(statVis(1)).elMask, true);

        % Accumulate data to save
    XEst = [XEst, X_i];
    PEst = [PEst, {P_i}];
    prefit_res = [prefit_res, Y_i - yBar_i];
    postfit_res = [postfit_res, Y_i - genMeas_post(1:2)];

        % Update for next run
    X_im1 = X_i;
    P_im1 = P_i;
    t_im1 = t_i;

end
```

Assign outputs

```
filterOut.XEst = XEst;
filterOut.PEst = PEst;
filterOut.prefit_res = prefit_res;
filterOut.postfit_res = postfit_res;
filterOut.t = t(2:end); % t_0 not included in estimate
filterOut.statVis = vis;

end
```

ii. add process noise

To add process noise, I am using $\sigma = 1 \times 10^{-8}$ km/s^2 for both UKF and EKF. The optimal value may differ from 1×10^{-8} for UKF.

see PDF for plots

after adding process noise to the UKF/EKF, I found that both filters performed similarly with the EKF slightly outperforming the UKF in terms of residual RMS. This is likely because the UKF has a slightly different optimal σ than the EKF's 1×10^{-8}. as expected, adding process noise eliminated the state error bias and resulted in the following pre/postfit RMS:

UKF pre/postfit RMS: 2.4621
EKF pre/postfit RMS: 0.9998

To improve the UKF, an optimal σ of 1×10^{-7} was chosen, resulting in a pre/postfit RMS of 0.9938, hence matching the EKF in terms of performance.

iii. $\alpha = 1\times10^{-4}$, all else the same

See PDF for plots

Changing α to 1×10^{-4} caused the UKF to break for this problem, resulting in state errors on the order of 2×10^{3} km in position and 20 km/s in velocity, which is clearly incorrect. This is expected, as setting α to be small results in very large weights. We know that

$$W_0^m = \frac{\lambda}{L+\lambda}, \quad W_0^c = \frac{\lambda}{L+\lambda} + (1-\alpha^2+\beta),$$

$$W_i^m = W_i^c = \frac{1}{2(L+\lambda)} \quad i = 1, \ldots, 2L$$

where $\lambda = \alpha^2(L+\kappa) - L$, $\kappa = 3-L$

If $\alpha \ll 1$, then $\lambda \to -L$ and

$(L+\lambda) \to 0$, i.e. the weights approach infinity. This causes tiny deviations in the sigma points to pull $\bar{x}$ in oscillatory directions, resulting in numerical instability and poorer performance than EKF, which effectively just uses 1 "sigma point"!

ASEN 6080 HW 6 Problem 1 Main Script

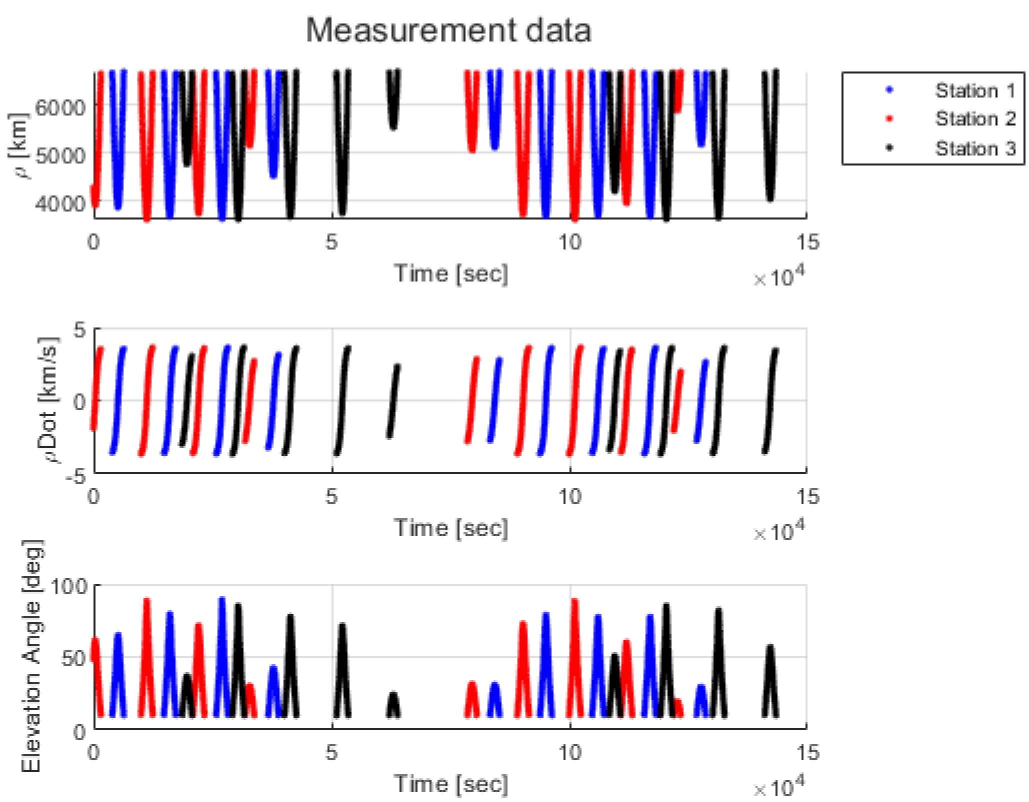
Table of Contents

By: Ian Faber

Housekeeping

Setup

Make truth data



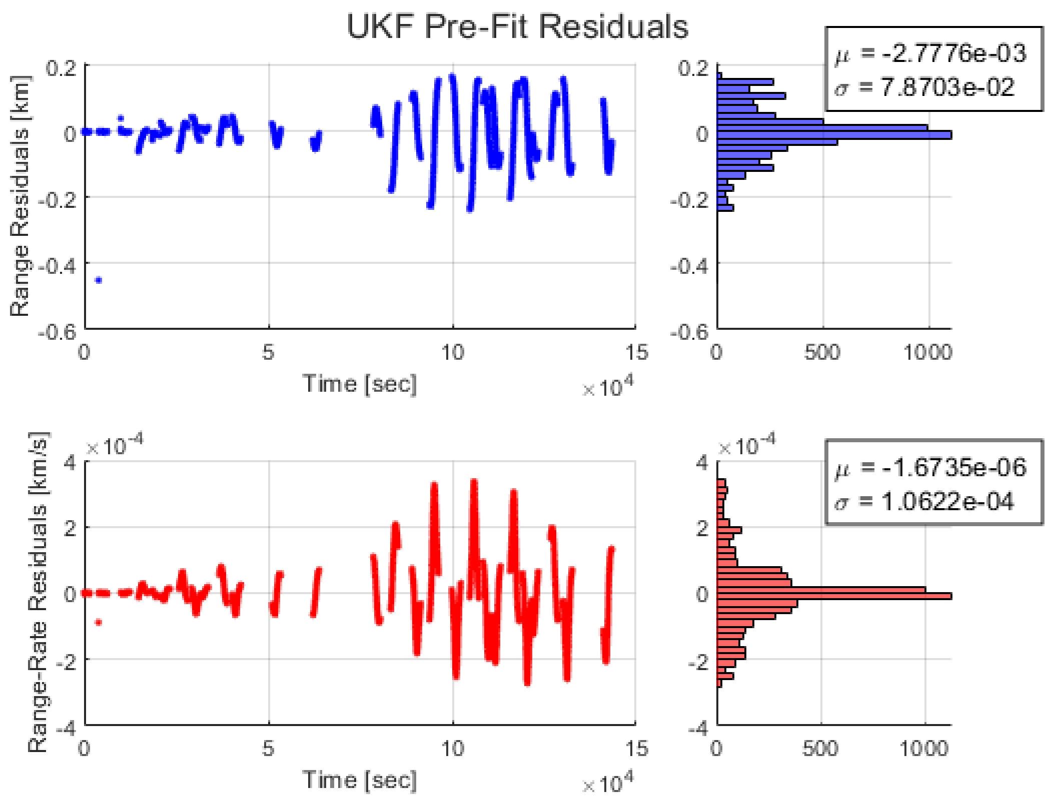
Problem 1a. Filter setup

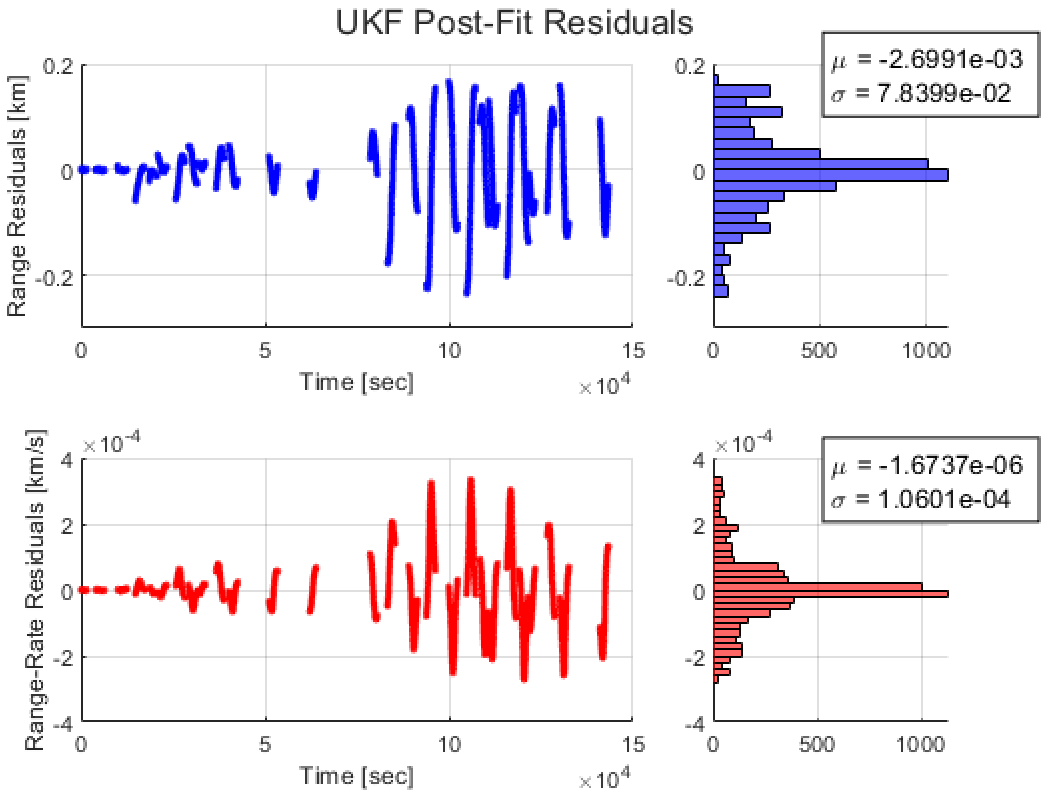
Problem 1b/c. UKF test cases

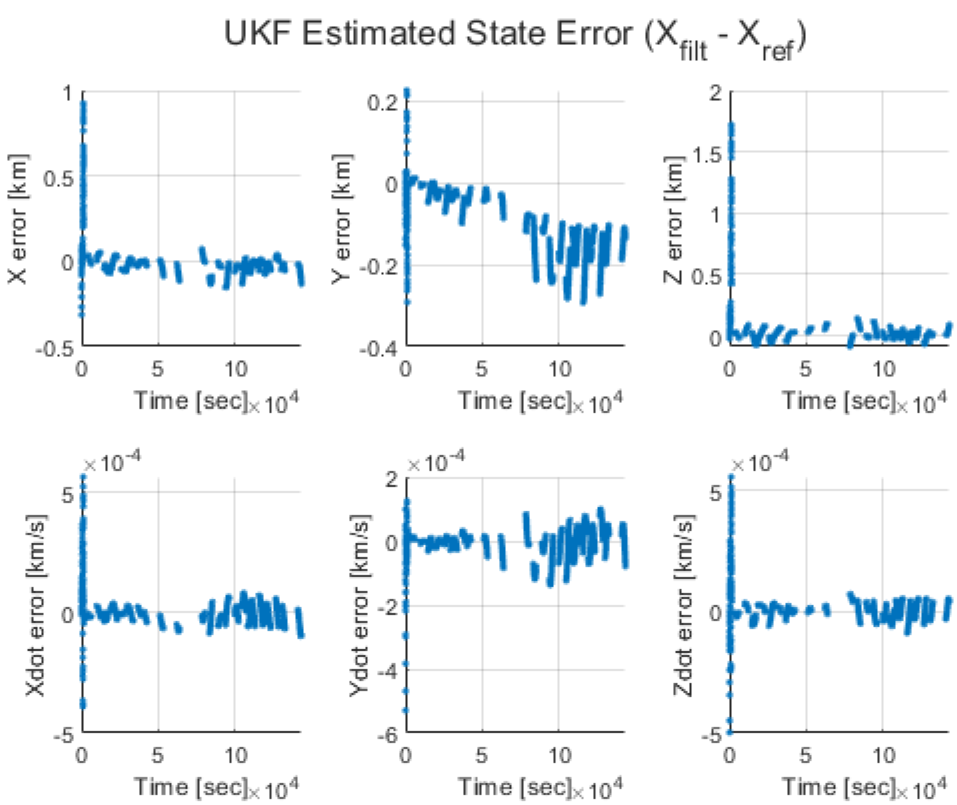
alpha = 1, beta = 2, no process noise

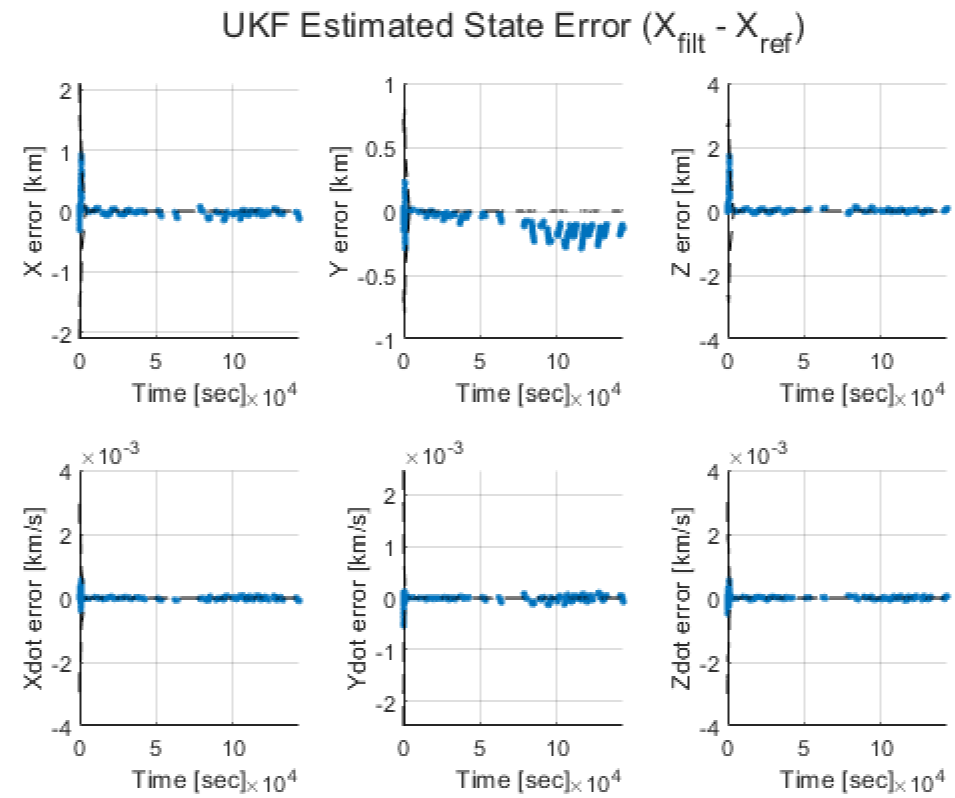
```
Problem 1b. UKF with alpha = 1.0000, beta = 2, no process noise

    Running UKF:
Prefit RMS: 93.2520
Postfit RMS: 93.2520

    Running LKF:
Prefit RMS: 242.0660, Postfit RMS: 93.4928. Hit max LKF iterations. Runs so
far: 1
Final prefit RMS: 242.0660. Hit maximum number of 1 runs
Final postfit RMS: 93.4928. Hit maximum number of 1 runs

    Running EKF:
Prefit RMS: 93.8586
Postfit RMS: 93.8586

Problem 1b. UKF with alpha = 1.0000, beta = 2, with process noise

    Running UKF:
Prefit RMS: 0.9967
Postfit RMS: 0.9967

    Running LKF:
Prefit RMS: 242.0660, Postfit RMS: 0.9967. Hit max LKF iterations. Runs so
far: 1
Final prefit RMS: 242.0660. Hit maximum number of 1 runs
Final postfit RMS: 0.9967. Hit maximum number of 1 runs

    Running EKF:
Prefit RMS: 0.9972
Postfit RMS: 0.9972

Problem 1b. UKF with alpha = 0.0001, beta = 2, with process noise

    Running UKF:
Prefit RMS: 12496922.9662
Postfit RMS: 12496922.9662
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
```

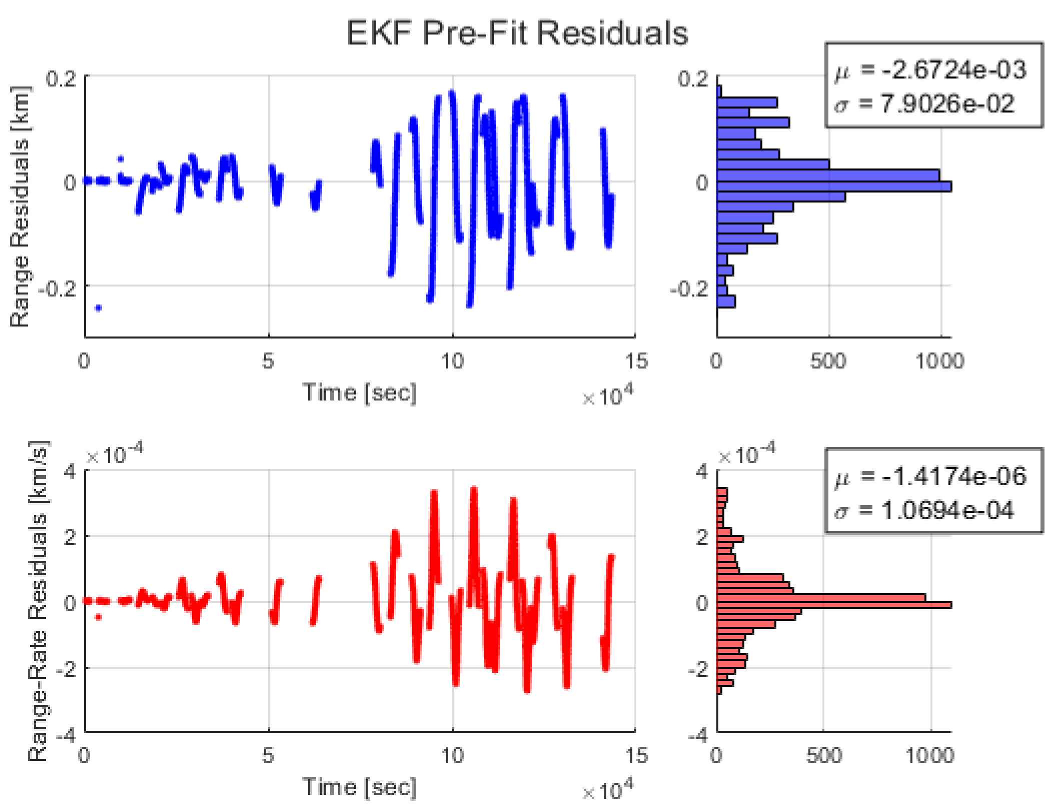
```
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
```

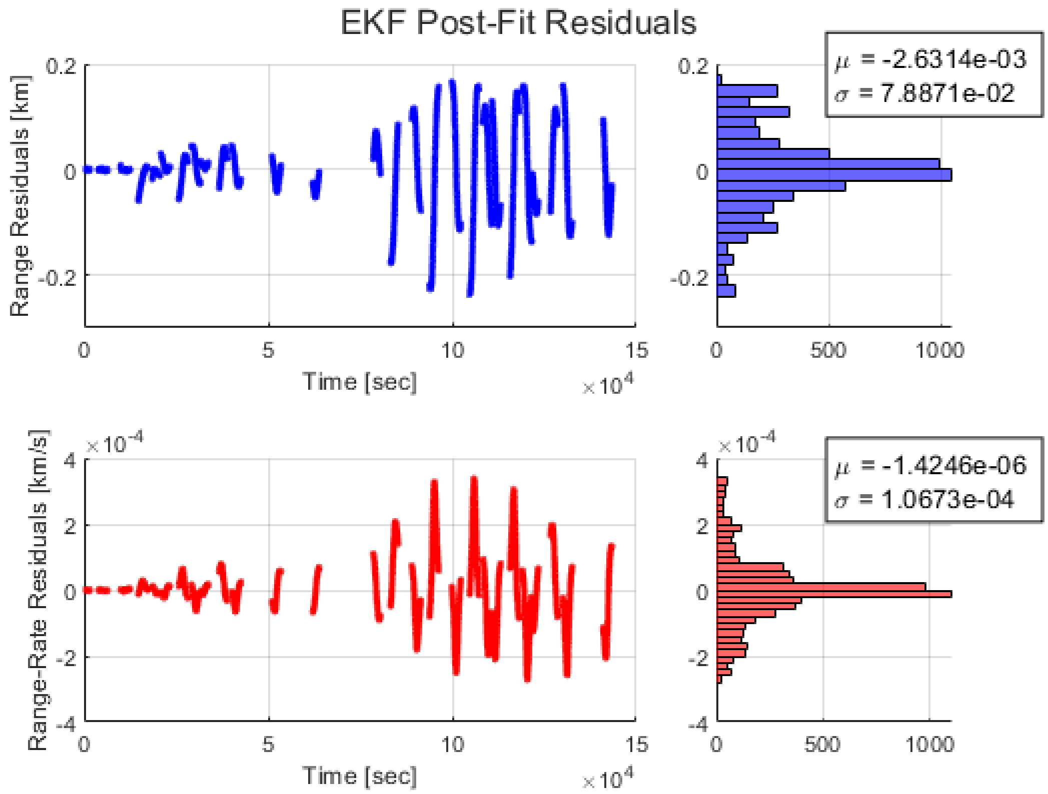


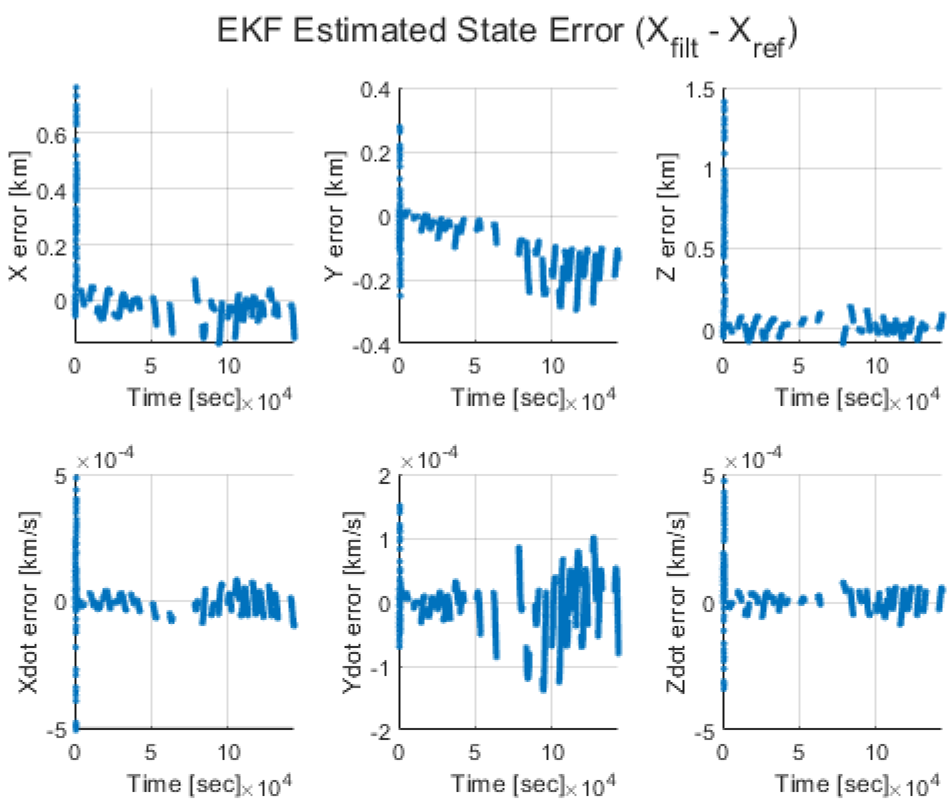

UKF Post-Fit Residuals
μ = -2.6991e-03
σ = 7.8399e-02
Range Residuals [km]
Time [sec]
×10^4
μ = -1.6737e-06
σ = 1.0601e-04
Range-Rate Residuals [km/s]
×10^-4

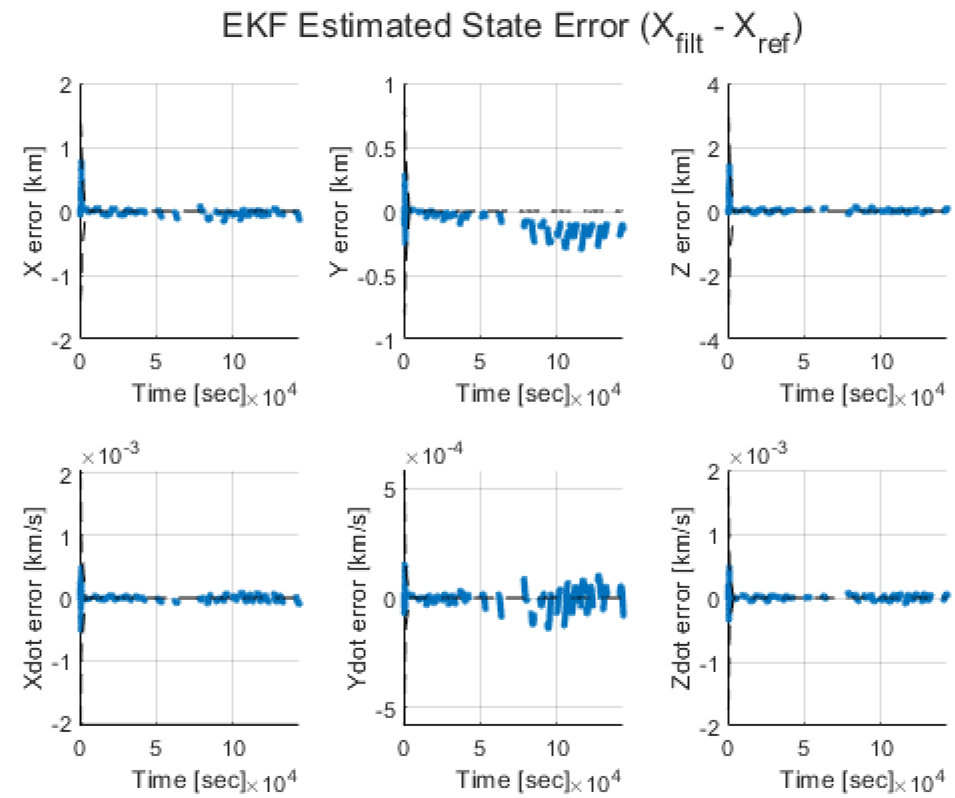

UKF Estimated State Error (X_filt - X_ref)
X error [km]
Y error [km]
Z error [km]
Xdot error [km/s]
Ydot error [km/s]
Zdot error [km/s]
Time [sec]
×10^4
×10^-4

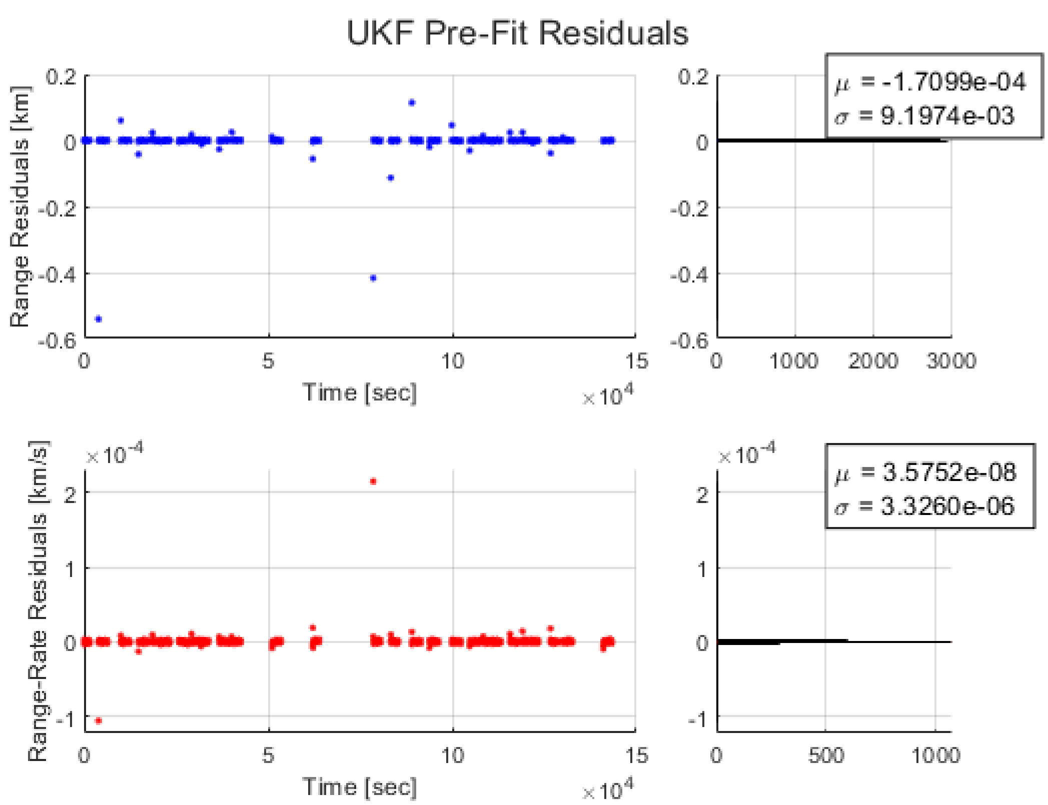

UKF Estimated State Error (X_{filt} - X_{ref})
X error [km]
Y error [km]
Z error [km]
Xdot error [km/s]
Ydot error [km/s]
Zdot error [km/s]
Time [sec]
×10^4
×10^-3

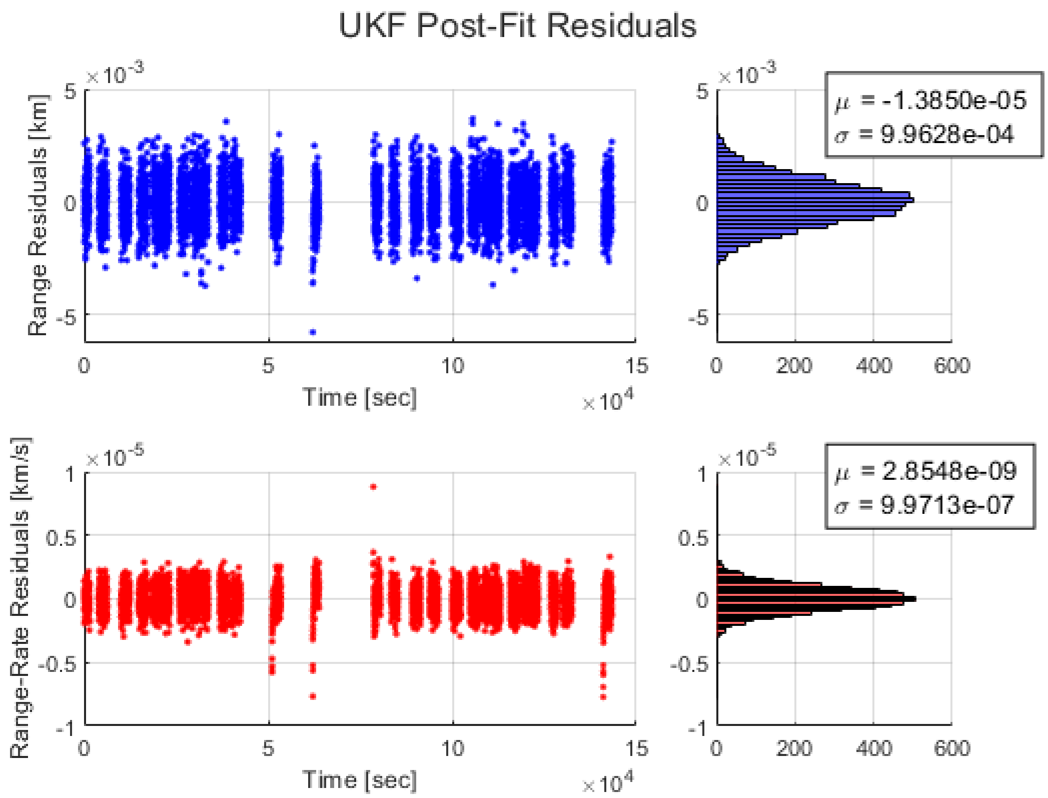

EKF Pre-Fit Residuals
Range Residuals [km]
Range-Rate Residuals [km/s]
Time [sec]
×10^4
×10^-4
μ = -2.6724e-03
σ = 7.9026e-02
μ = -1.4174e-06
σ = 1.0694e-04

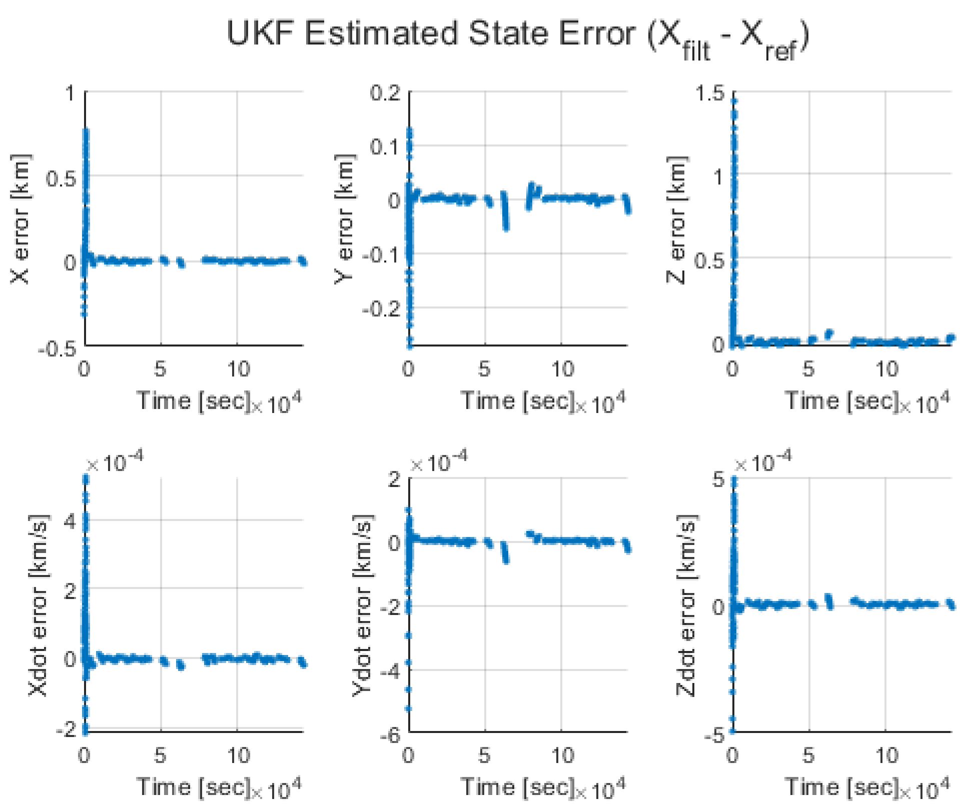

EKF Post-Fit Residuals
Range Residuals [km]
Time [sec]
μ = -2.6314e-03
σ = 7.8871e-02
Range-Rate Residuals [km/s]
Time [sec]
μ = -1.4246e-06
σ = 1.0673e-04

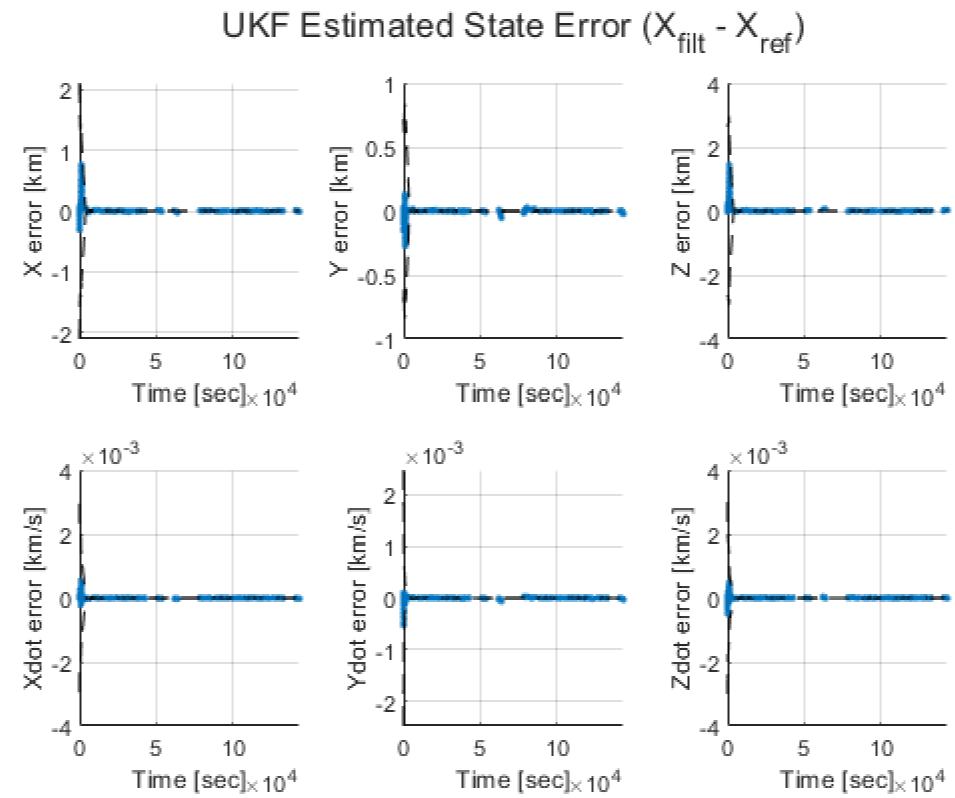

EKF Estimated State Error (X_filt - X_ref)
X error [km]
Y error [km]
Z error [km]
Xdot error [km/s]
Ydot error [km/s]
Zdot error [km/s]
Time [sec]

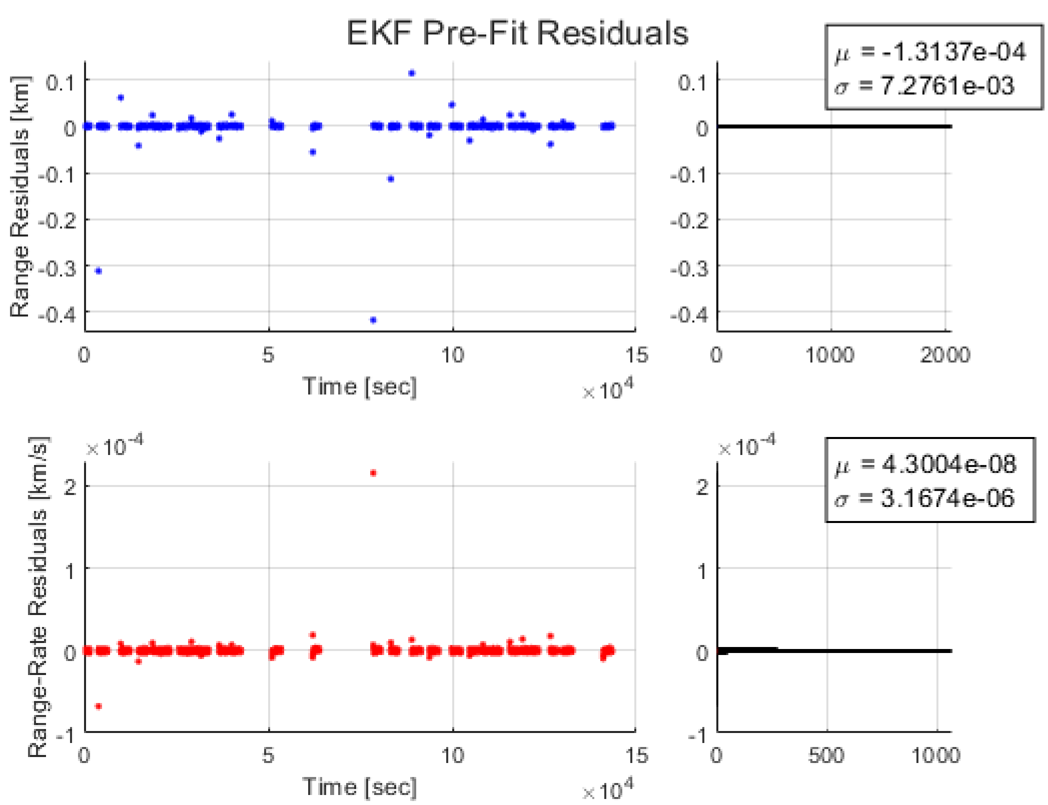

EKF Estimated State Error (X_filt - X_ref)
X error [km]
Y error [km]
Z error [km]
Xdot error [km/s]
Ydot error [km/s]
Zdot error [km/s]
Time [sec]
×10^4
×10^-3
×10^-4

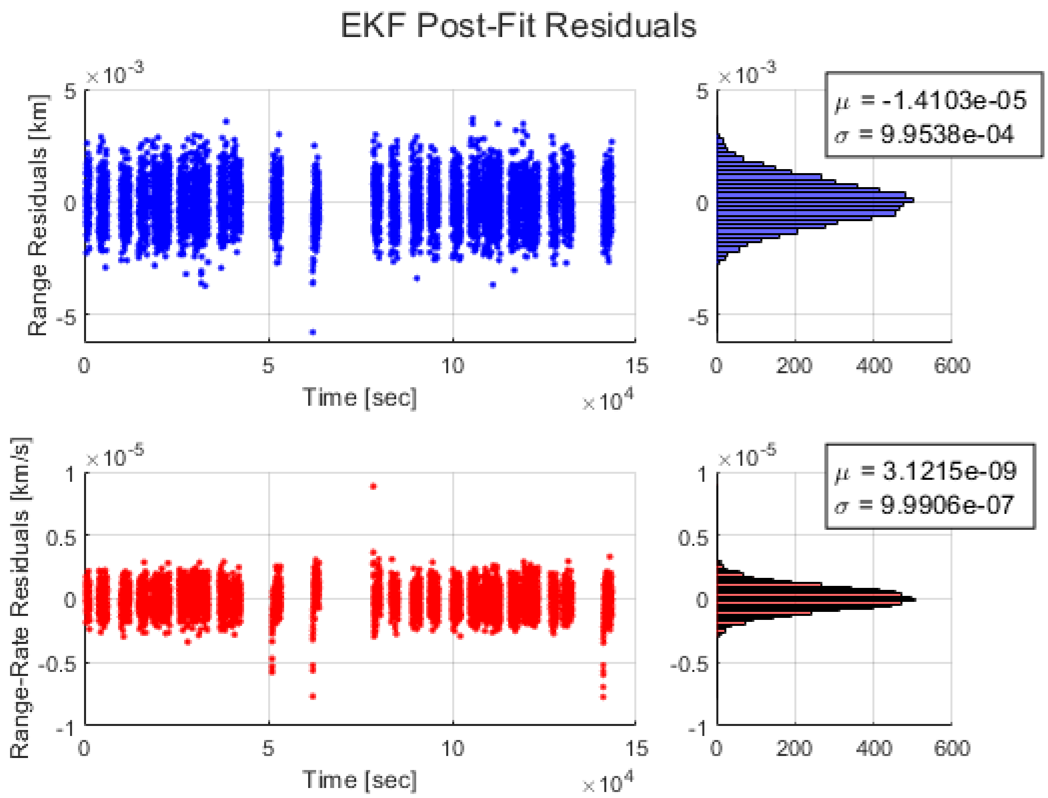

UKF Pre-Fit Residuals
Range Residuals [km]
Range-Rate Residuals [km/s]
Time [sec]
×10^4
×10^-4
μ = -1.7099e-04
σ = 9.1974e-03
μ = 3.5752e-08
σ = 3.3260e-06

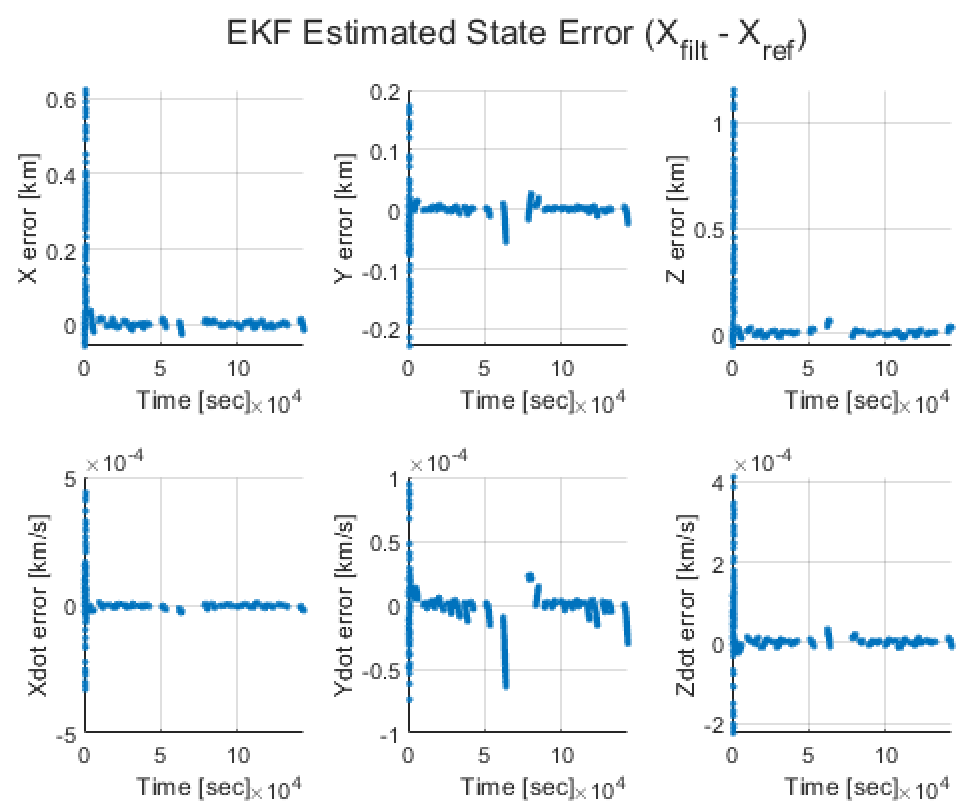

UKF Post-Fit Residuals
Range Residuals [km]
Time [sec]
μ = -1.3850e-05
σ = 9.9628e-04
Range-Rate Residuals [km/s]
μ = 2.8548e-09
σ = 9.9713e-07

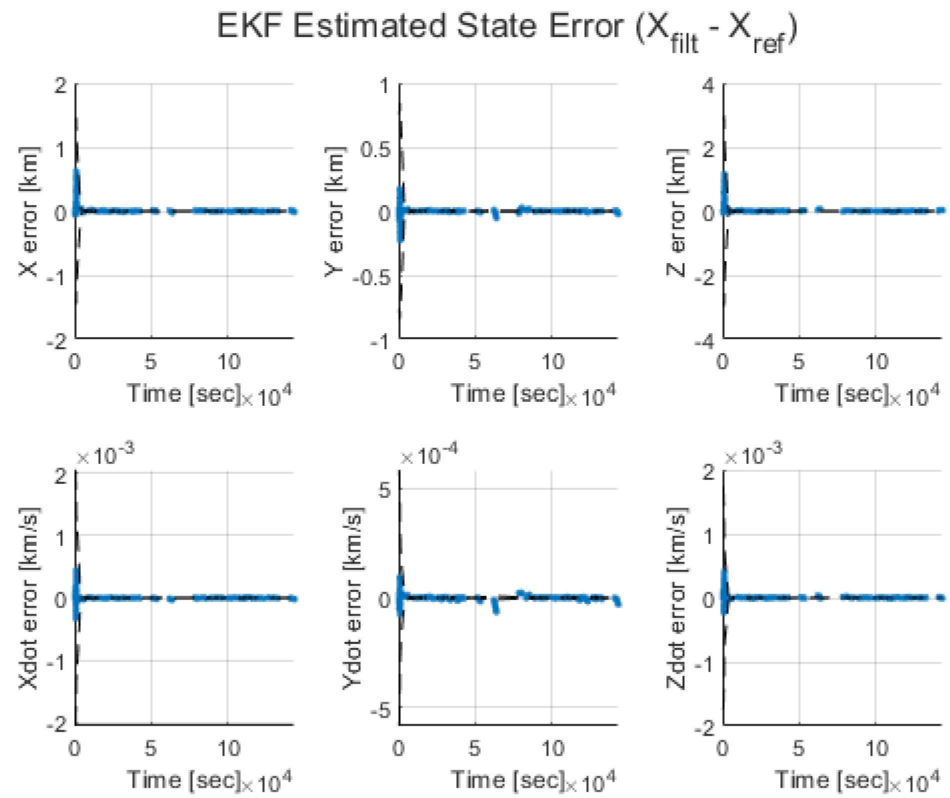

UKF Estimated State Error (X_filt - X_ref)
X error [km]
Y error [km]
Z error [km]
Xdot error [km/s]
Ydot error [km/s]
Zdot error [km/s]
Time [sec]

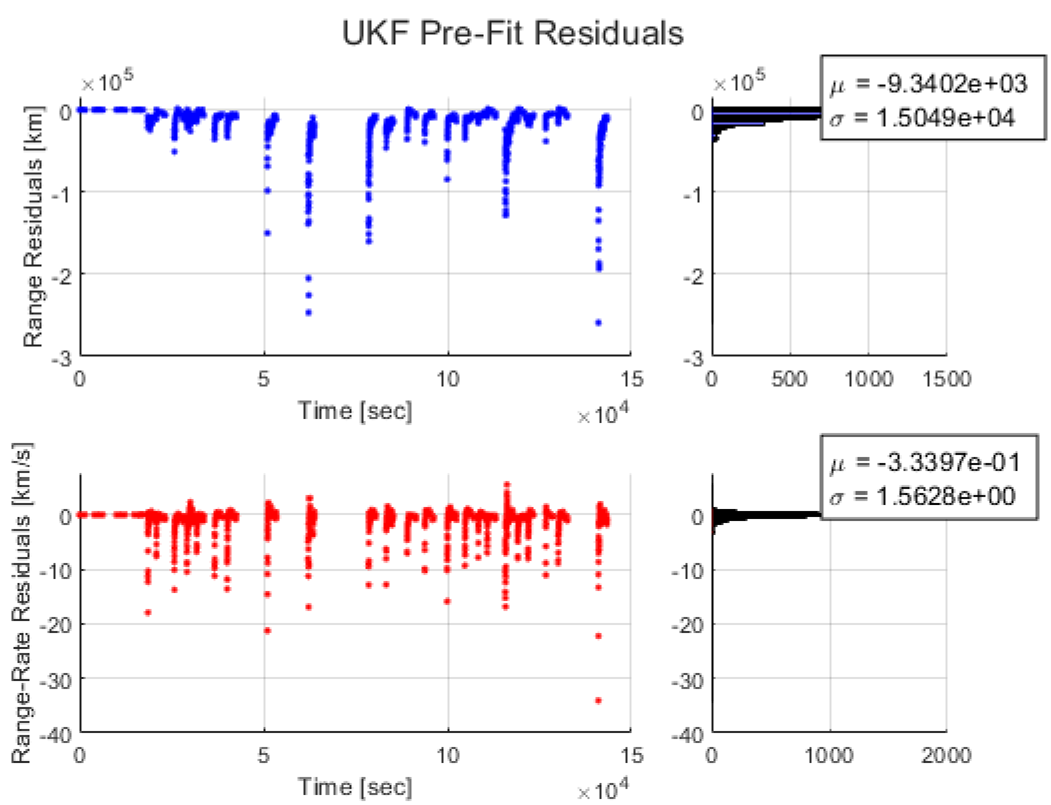

UKF Estimated State Error (X_{filt} - X_{ref})
X error [km]
Y error [km]
Z error [km]
Xdot error [km/s]
Ydot error [km/s]
Zdot error [km/s]
Time [sec]
×10^4
×10^-3

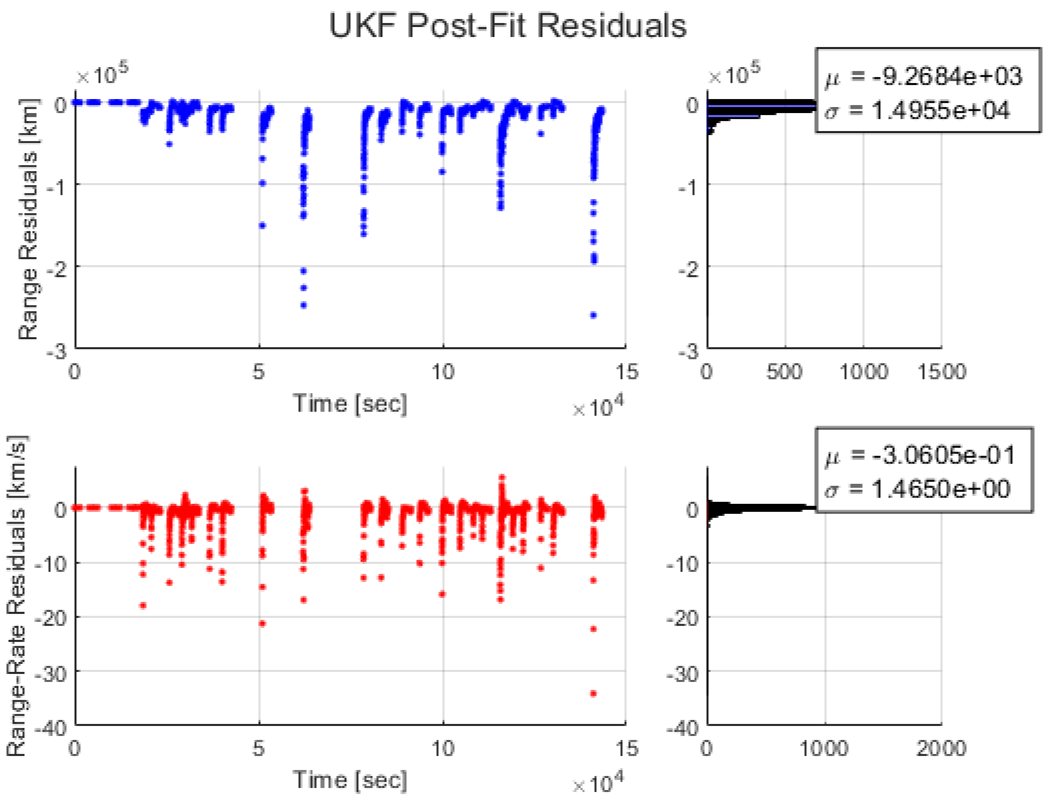

EKF Pre-Fit Residuals
Range Residuals [km]
Range-Rate Residuals [km/s]
Time [sec]
×10^4
×10^-4
μ = -1.3137e-04
σ = 7.2761e-03
μ = 4.3004e-08
σ = 3.1674e-06

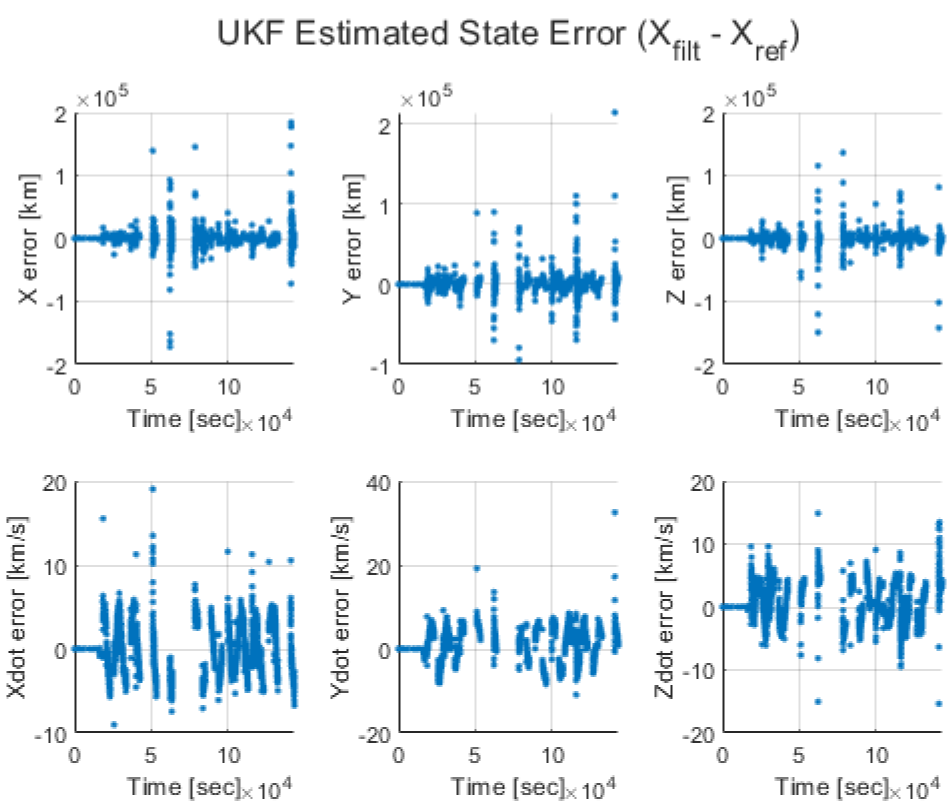

EKF Post-Fit Residuals
Range Residuals [km]
Time [sec]
μ = -1.4103e-05
σ = 9.9538e-04
Range-Rate Residuals [km/s]
μ = 3.1215e-09
σ = 9.9906e-07


EKF Estimated State Error (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]


EKF Estimated State Error (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]
×10^4
×10^-3
×10^-4


UKF Pre-Fit Residuals
Range Residuals [km]
Range-Rate Residuals [km/s]
Time [sec]
×10^4
×10^5
μ = -9.3402e+03
σ = 1.5049e+04
μ = -3.3397e-01
σ = 1.5628e+00


UKF Post-Fit Residuals
Range Residuals [km]
Time [sec]
Range-Rate Residuals [km/s]
Time [sec]
μ = -9.2684e+03
σ = 1.4955e+04
μ = -3.0605e-01
σ = 1.4650e+00


UKF Estimated State Error (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]

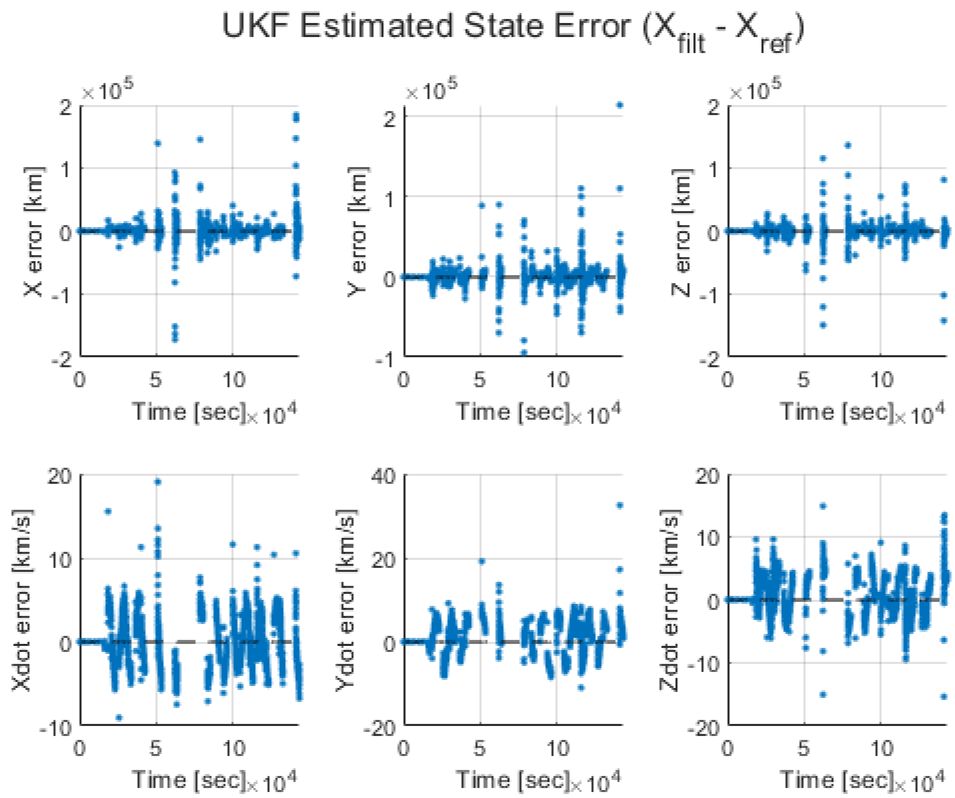
d. Investigate the claim that the UKF is more robust to large errors in initial conditions than the EKF.

See PDF for plots. $\alpha = 1$ again!

This claim appears to be true. When initializing the estimated state at 2 times the original value, the resulting pre/postfit RMS is smaller for UKF by a bit over a factor of 10. Further, the UKF state error appears to converge to around 0 as more measurements are processed, while the EKF state error oscillates wildly and without recovery.

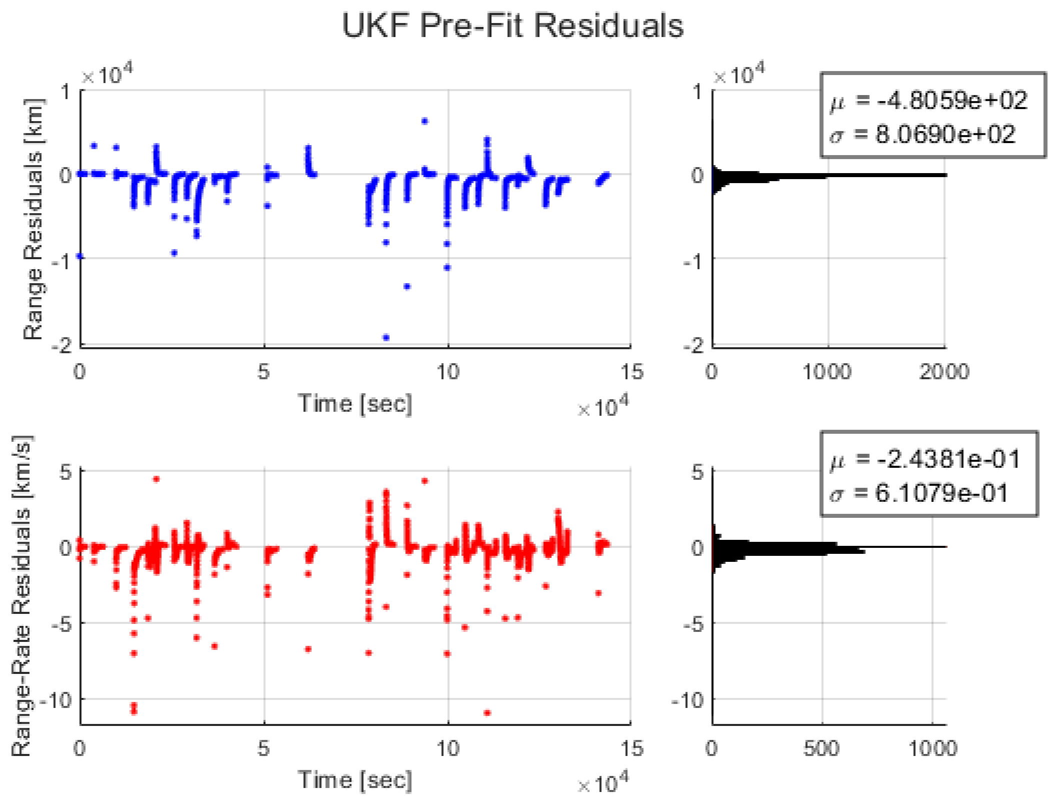
UKF pre/postfit RMS at $\hat{x}_0 = 2 \cdot \hat{x}_{0,init}$: 46,990.7934

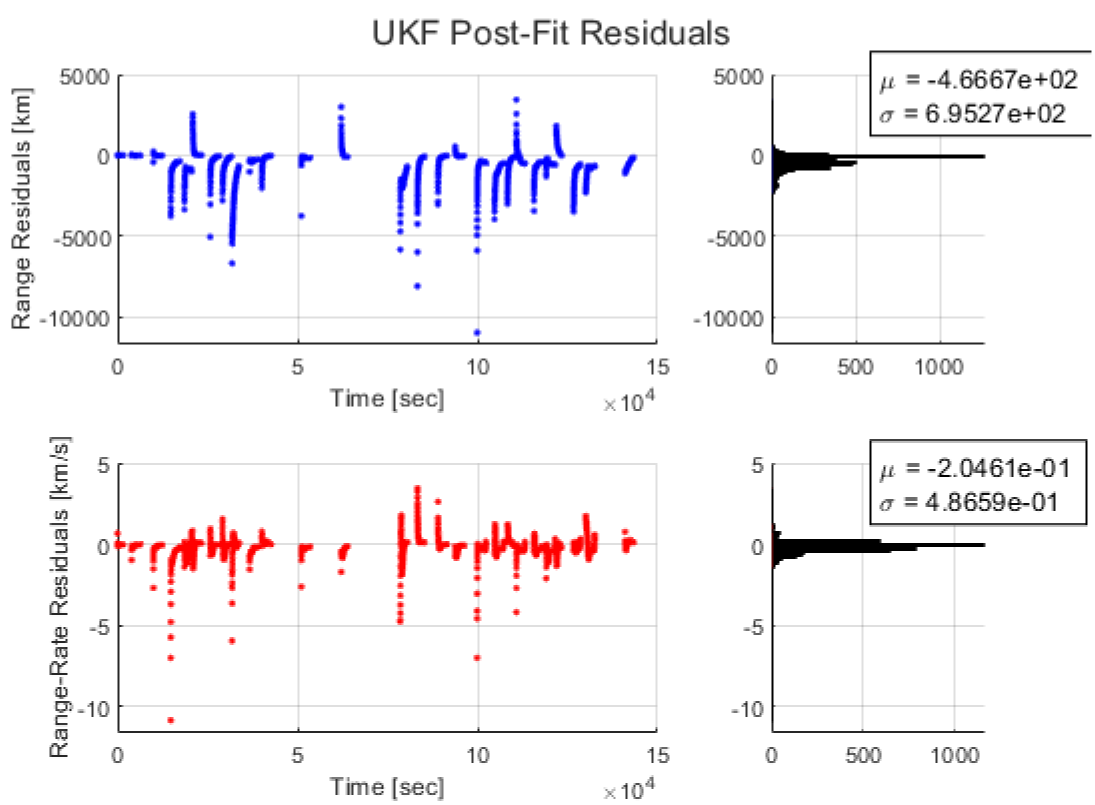
EKF pre/postfit RMS at $\hat{x}_0 = 2 \cdot \hat{x}_{0,init}$: 577,191.8930

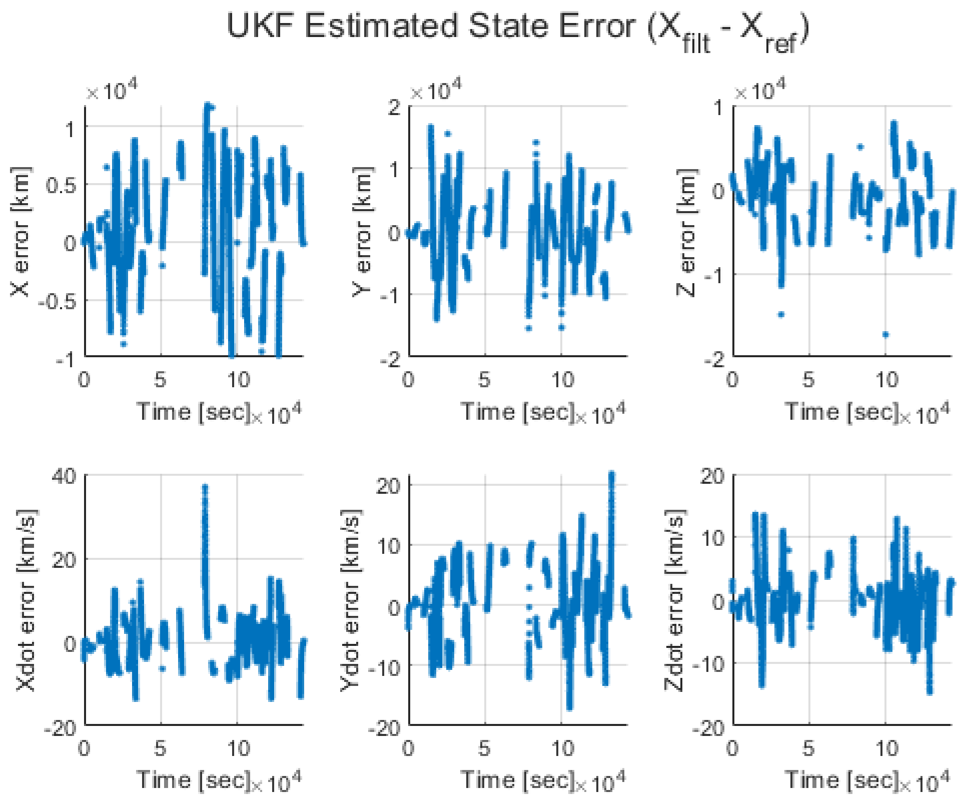


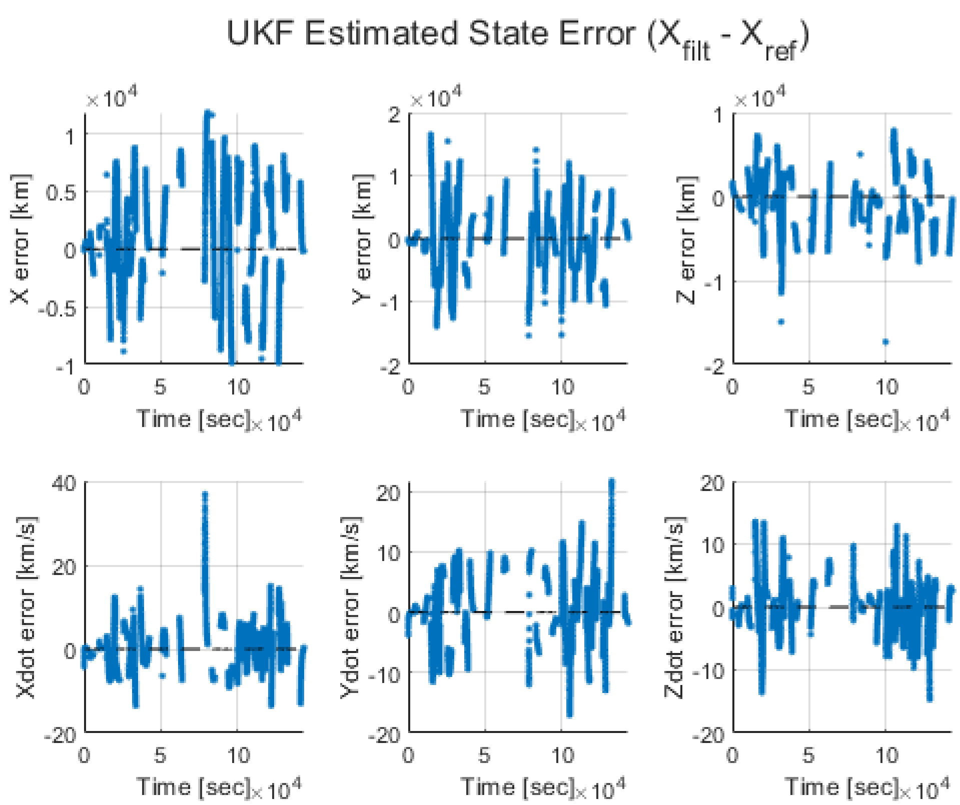
Problem 1d. Investigate how robust UKF is to large state errors compared to EKF

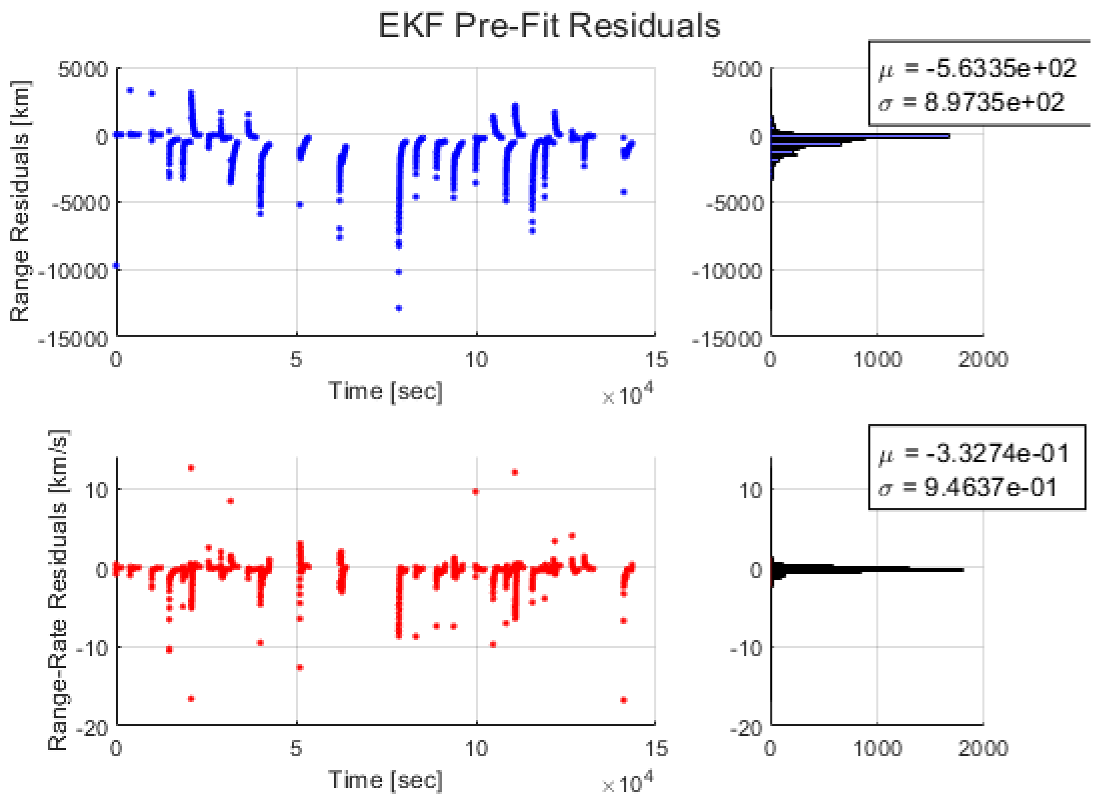
```
Problem 1d. Investigate performance of UKF vs. EKF for large initial state
errors

    Running UKF:
Prefit RMS: 699895.3461
Postfit RMS: 699895.3461

    Running EKF:
Prefit RMS: 870361.8262
Postfit RMS: 870361.8262
```

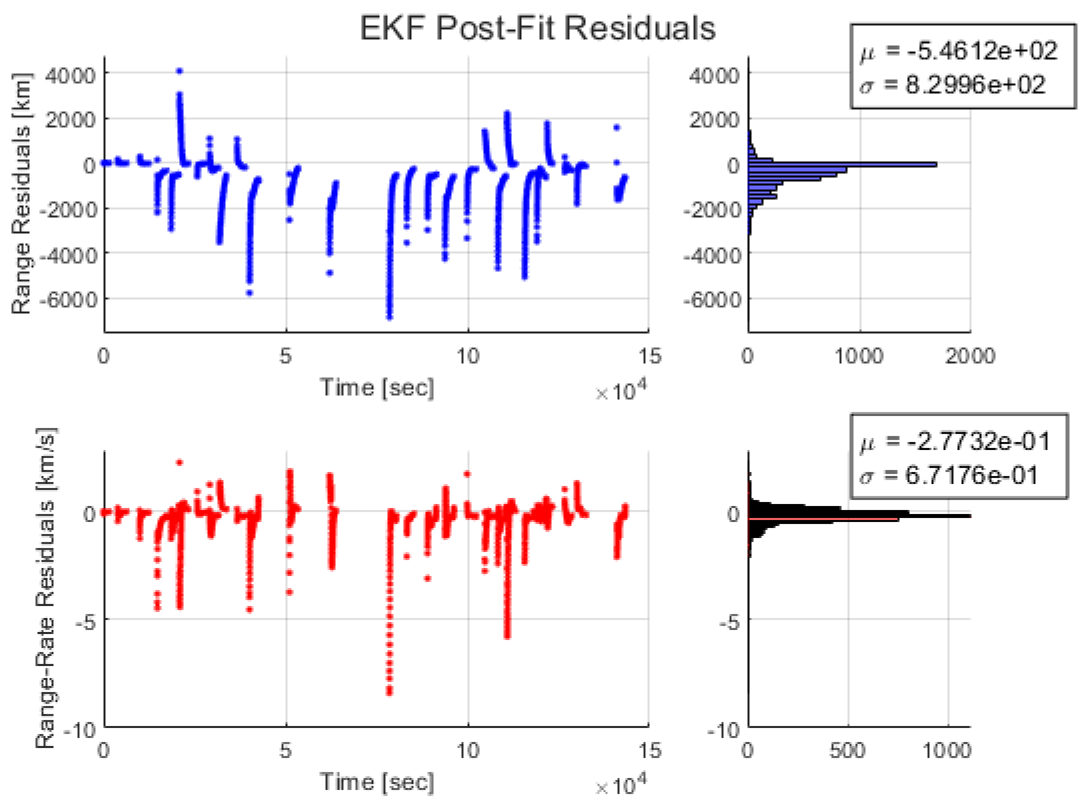

UKF Pre-Fit Residuals
Range Residuals [km]
Time [sec]
Range-Rate Residuals [km/s]
μ = -4.8059e+02
σ = 8.0690e+02
μ = -2.4381e-01
σ = 6.1079e-01

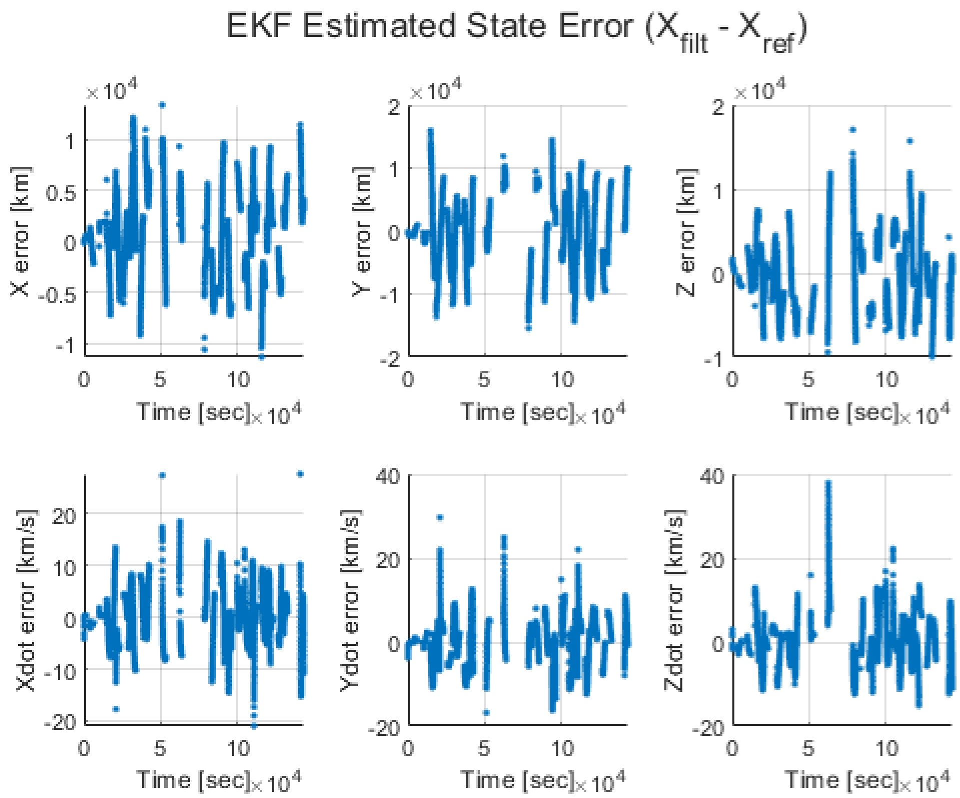

UKF Post-Fit Residuals
Range Residuals [km]
Time [sec]
Range-Rate Residuals [km/s]
μ = -4.6667e+02
σ = 6.9527e+02
μ = -2.0461e-01
σ = 4.8659e-01

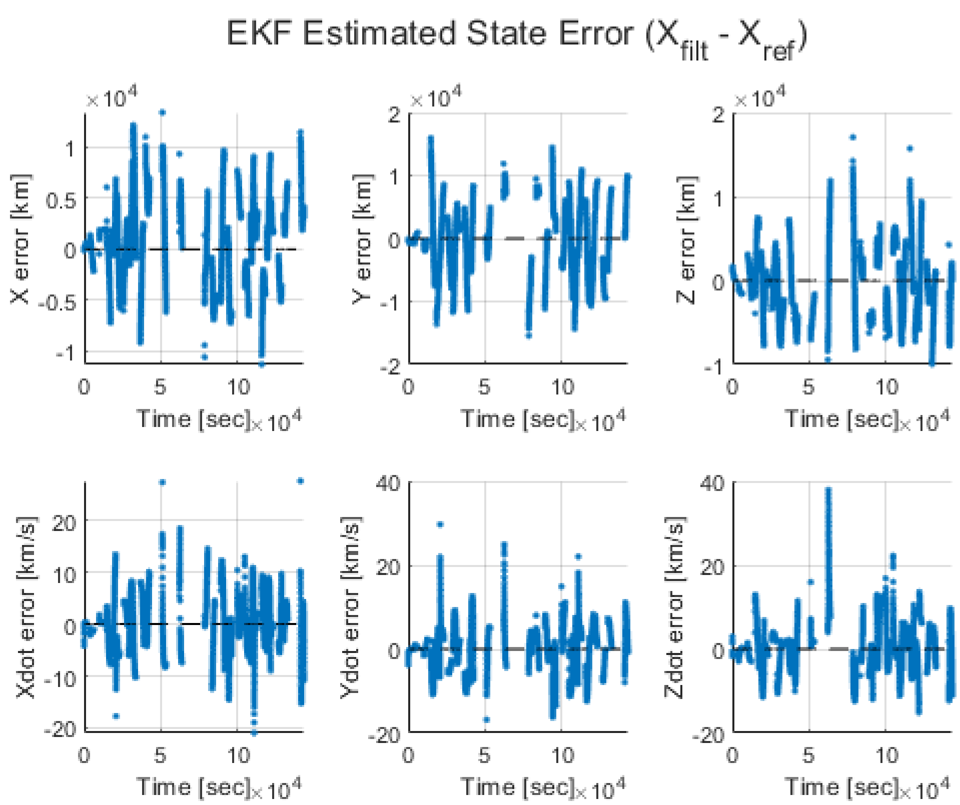

UKF Estimated State Error (X_{filt} - X_{ref})
X error [km]
Y error [km]
Z error [km]
Xdot error [km/s]
Ydot error [km/s]
Zdot error [km/s]
Time [sec]
$\times 10^4$


UKF Estimated State Error (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]
$\times 10^4$


EKF Pre-Fit Residuals
Range Residuals [km]
Time [sec]
×10^4
μ = -5.6335e+02
σ = 8.9735e+02
Range-Rate Residuals [km/s]
Time [sec]
×10^4
μ = -3.3274e-01
σ = 9.4637e-01


EKF Post-Fit Residuals
Range Residuals [km]
Time [sec]
×10^4
μ = -5.4612e+02
σ = 8.2996e+02
Range-Rate Residuals [km/s]
Time [sec]
×10^4
μ = -2.7732e-01
σ = 6.7176e-01


EKF Estimated State Error (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]


EKF Estimated State Error (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]

e. add J_3 to the UKF dynamical model. How long did this take, and how are the results?

see PDF for plots

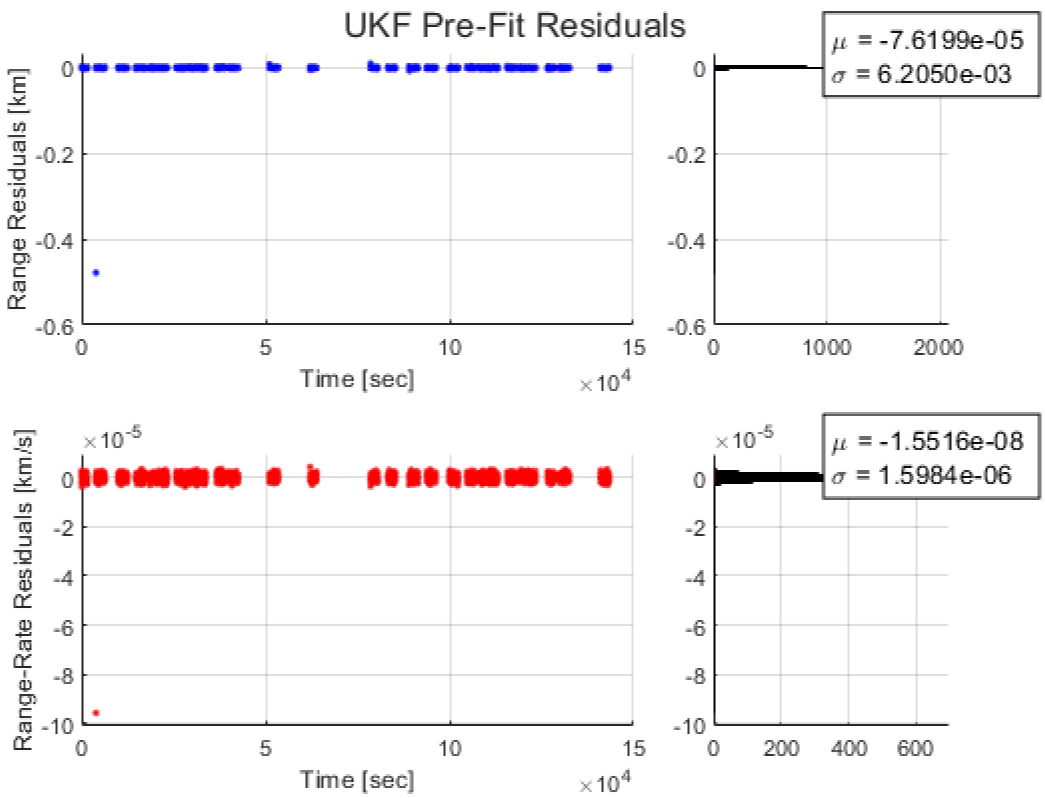
adding J_3 to the UKF dynamics model was as easy as changing out the nonlinear EOM to include J_3. In fact, I accomplish this with a boolean I can flip to add or ignore J_3.

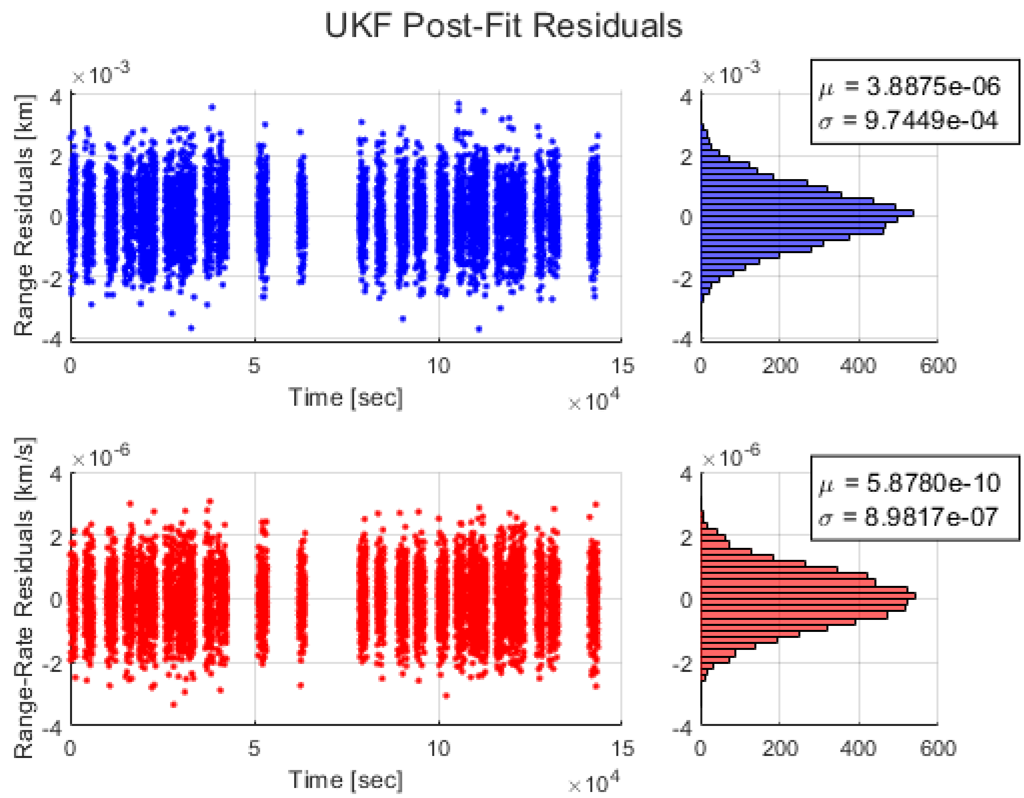
as for results, the UKF produces a state estimate with state errors that approach noise and residuals with mean + standard deviation that we'd expect.

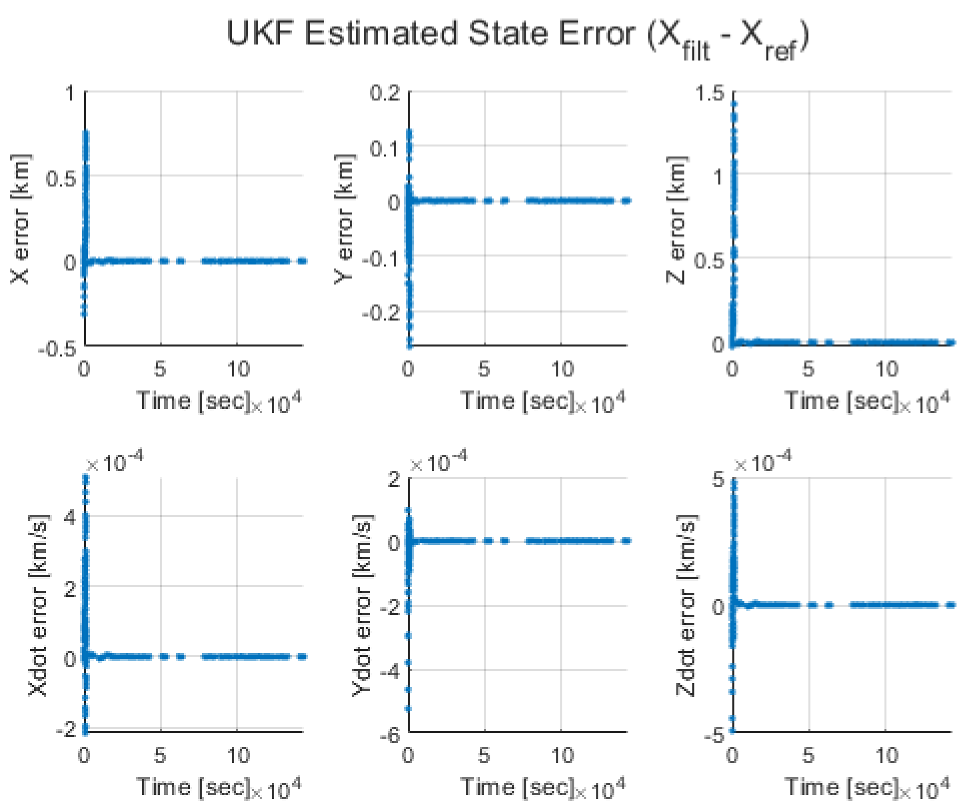
UKF pre/postfit RMS: 0.9713

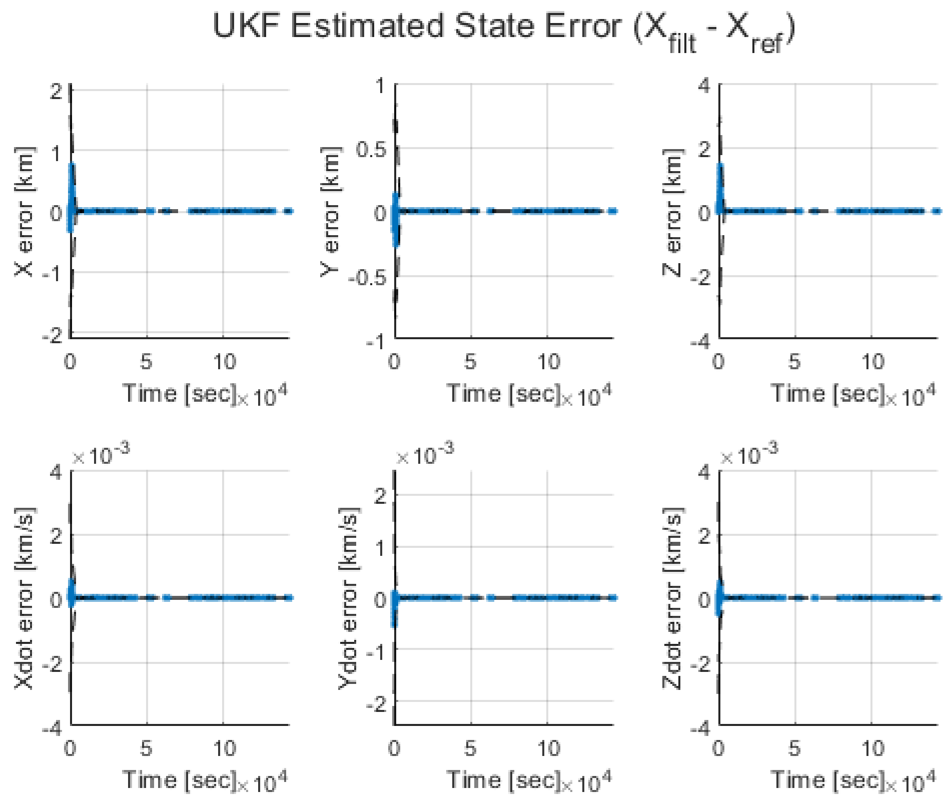
Problem 1e. Include J3 in filter dynamics

```
Problem 1e. Add J3 to filter dynamics

    Running UKF:
Prefit RMS: 0.9370
Postfit RMS: 0.9370
```




UKF Post-Fit Residuals
Range Residuals [km]
×10^-3
Time [sec]
×10^4
μ = 3.8875e-06
σ = 9.7449e-04
Range-Rate Residuals [km/s]
×10^-6
μ = 5.8780e-10
σ = 8.9817e-07


UKF Estimated State Error (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]
×10^4
×10^-4



Published with MATLAB® R2023b