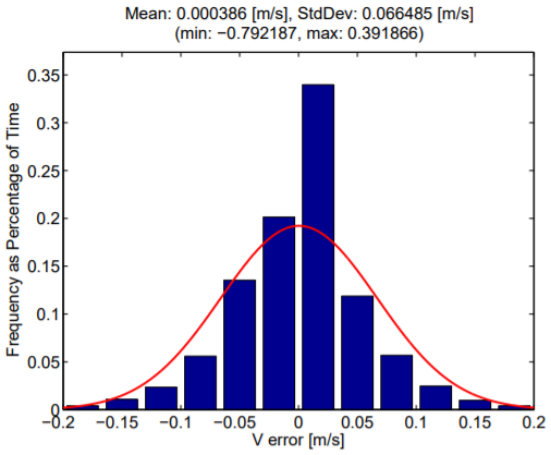
Assignment 2

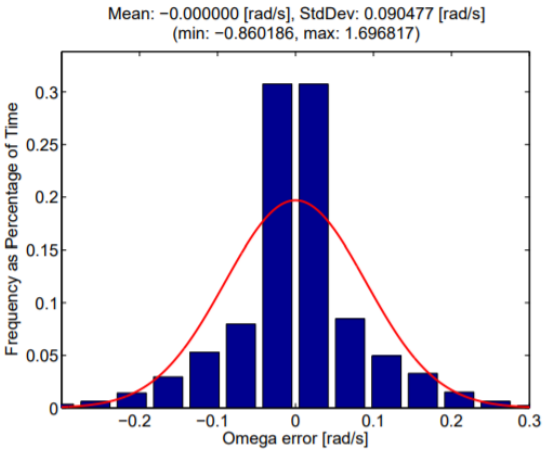
AER 1513 – Prof. Barfoot

Sam Weinberg (1005347634)

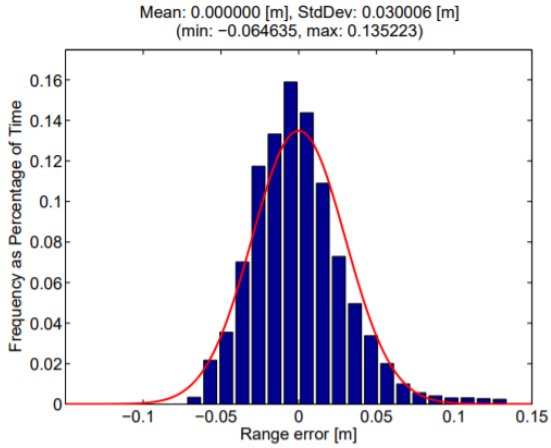
1)



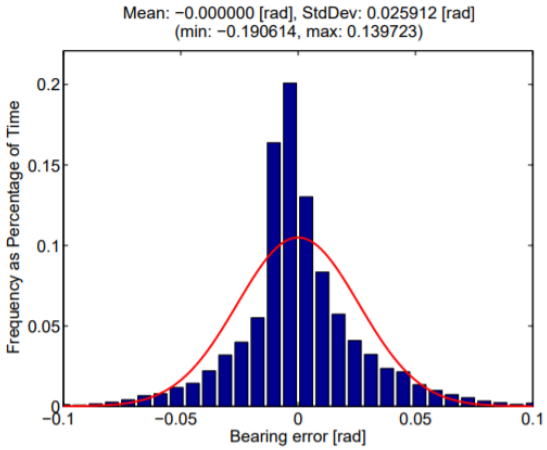
(a) Translational speed errors.



(b) Rotational speed errors.



(a) Range errors.



(b) Bearing errors.

Based on the above data, we can observe that the data approximately follows a Gaussian distribution centred about zero. Therefore, the zero-mean Gaussian assumption seems reasonable. We can use the standard deviations from the data to derive expressions for the noise model variances.

For the motion model, we can use the variances for the velocity and angular velocity (σ_v^2 and σ_ω^2) from the first two plots as follows:

$$\boldsymbol{w_k} \sim \mathcal{N}(0, \boldsymbol{Q_k})$$

$$\boldsymbol{Q_k} = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\omega^2 \end{bmatrix}$$

For the observation model, we can use the variances for the range and bearing measurements (σ_r^2 and σ_b^2) from the last two plots as follows:

$$\boldsymbol{n}_k^l \sim \mathcal{N}\left(0, \boldsymbol{R}_k^l\right)$$

$$\boldsymbol{R}_k^l = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_b^2 \end{bmatrix}$$

We will derive $\boldsymbol{Q_k}'$ and $\boldsymbol{R}_k^l{}'$ using the Jacobians from question 2.

2) The motion model is given by:

$$\underbrace{\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}}_{\mathbf{x}_k} = \underbrace{\underbrace{\begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix}}_{\mathbf{x}_{k-1}} + T \begin{bmatrix} \cos\theta_{k-1} & 0 \\ \sin\theta_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \left(\underbrace{\begin{bmatrix} v_k \\ \omega_k \end{bmatrix}}_{\mathbf{u}_k} + \mathbf{w}_k \right)}_{\mathbf{h}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k)}$$

The Jacobian of the motion model is given by:

$$\boldsymbol{F}_{k-1} = \left. \frac{\partial \boldsymbol{h}(\boldsymbol{x}_{k-1}, \boldsymbol{v}_k, \boldsymbol{w}_k)}{\partial \boldsymbol{x}_{k-1}} \right|_{\hat{x}_{k-1}, v_k, 0}$$

$$\boldsymbol{F}_{k-1} = \begin{bmatrix} \frac{\partial h_1}{\partial x_{k-1}} & \frac{\partial h_1}{\partial y_{k-1}} & \frac{\partial h_1}{\partial \theta_{k-1}} \\ \frac{\partial h_2}{\partial x_{k-1}} & \frac{\partial h_2}{\partial y_{k-1}} & \frac{\partial h_2}{\partial \theta_{k-1}} \\ \frac{\partial h_3}{\partial x_{k-1}} & \frac{\partial h_3}{\partial y_{k-1}} & \frac{\partial h_3}{\partial \theta_{k-1}} \end{bmatrix}$$

Where $\boldsymbol{h} = \boldsymbol{x} = [h_1\ h_2\ h_3]^T = [x_k\ y_k\ \theta_k]^T$

It is apprant that the diagonals of the matrix will be 1 from the derivative of the first term. In addition, the expressions for x_k and y_k have dependencies on θ_{k-1}. Filling in the derivatives, we have:

$$\boldsymbol{F}_{k-1} = \begin{bmatrix} 1 & 0 & -T(v_k + w_k)sin\theta_{k-1} \\ 0 & 1 & T(v_k + w_k)cos\theta_{k-1} \\ 0 & 0 & 1 \end{bmatrix}$$

We substitute $w_k = 0$ to obtain:

$$\boldsymbol{F}_{k-1} = \begin{bmatrix} 1 & 0 & -T(v_k)sin\theta_{k-1} \\ 0 & 1 & T(v_k)cos\theta_{k-1} \\ 0 & 0 & 1 \end{bmatrix}$$

In addition, we require the following Jacobian for the process noise model:

$$\boldsymbol{w}_k' = \left. \frac{\partial \boldsymbol{h}(\boldsymbol{x}_{k-1}, \boldsymbol{v}_k, \boldsymbol{w}_k)}{\partial \boldsymbol{w}_k} \right|_{\hat{x}_{k-1}, v_k, 0} \boldsymbol{w}_k$$

If we consider $\boldsymbol{w_k} = [w_1\ w_2]^T$, then we have:

$$\left.\frac{\partial \boldsymbol{h}(\boldsymbol{x}_{k-1}, \boldsymbol{v}_k, \boldsymbol{w}_k)}{\partial \boldsymbol{w}_k}\right|_{\hat{x}_{k-1}, v_k, 0} = \begin{bmatrix} \frac{\partial h_1}{\partial w_1} & \frac{\partial h_1}{\partial w_2} \\ \frac{\partial h_2}{\partial w_1} & \frac{\partial h_2}{\partial w_2} \\ \frac{\partial h_3}{\partial w_1} & \frac{\partial h_3}{\partial w_2} \end{bmatrix} = \begin{bmatrix} Tcos\theta_{k-1} & 0 \\ Tsin\theta_{k-1} & 0 \\ 0 & T \end{bmatrix}$$

$$\boldsymbol{w}'_k = \left.\frac{\partial \boldsymbol{h}(\boldsymbol{x}_{k-1}, \boldsymbol{v}_k, \boldsymbol{w}_k)}{\partial \boldsymbol{w}_k}\right|_{\hat{x}_{k-1}, v_k, 0} \boldsymbol{w}_k = \begin{bmatrix} Tcos\theta_{k-1} & 0 \\ Tsin\theta_{k-1} & 0 \\ 0 & T \end{bmatrix} \boldsymbol{w_k}$$

Now to obtain $\boldsymbol{Q'_k}$, we consider the variance of $\boldsymbol{w'_k}$. Recall the following property of variance for a constant, a, and random variable x:

$$Var[ax] = a^2 Var[x]$$

Therefore, we have:

$$\boldsymbol{Q'_k} = Var[\boldsymbol{w'_k}] = Var\left[\begin{bmatrix} Tcos\theta_{k-1} & 0 \\ Tsin\theta_{k-1} & 0 \\ 0 & T \end{bmatrix} \boldsymbol{w_k}\right] = \begin{bmatrix} Tcos\theta_{k-1} & Tsin\theta_{k-1} & 0 \\ 0 & 0 & T \end{bmatrix} \begin{bmatrix} Tcos\theta_{k-1} & 0 \\ Tsin\theta_{k-1} & 0 \\ 0 & T \end{bmatrix} Q_k$$

$$\boldsymbol{Q'_k} = \begin{bmatrix} Tcos\theta_{k-1} & Tsin\theta_{k-1} & 0 \\ 0 & 0 & T \end{bmatrix} \begin{bmatrix} Tcos\theta_{k-1} & 0 \\ Tsin\theta_{k-1} & 0 \\ 0 & T \end{bmatrix} \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\omega^2 \end{bmatrix}$$

$$\boldsymbol{Q'_k} = \begin{bmatrix} (Tcos\theta_{k-1})^2\sigma_v^2 & T^2 cos\theta_{k-1} sin\theta_{k-1}\sigma_v^2 & 0 \\ T^2 cos\theta_{k-1} sin\theta_{k-1}\sigma_v^2 & (Tsin\theta_{k-1})^2\sigma_r^2 & 0 \\ 0 & 0 & \sigma_\omega^2 \end{bmatrix}$$

Now for the observation model:

$$\underbrace{\begin{bmatrix} r_k^l \\ \phi_k^l \end{bmatrix}}_{\mathbf{y}_k^l} = \underbrace{\begin{bmatrix} \sqrt{(x_l - x_k - d\cos\theta_k)^2 + (y_l - y_k - d\sin\theta_k)^2} \\ \text{atan2}(y_l - y_k - d\sin\theta_k, x_l - x_k - d\cos\theta_k) - \theta_k \end{bmatrix}}_{\mathbf{g}(\mathbf{x}_k, \mathbf{n}_k^l)} + \mathbf{n}_k^l$$

The Jacobian of the motion model is given by:

$$G_k = \left.\frac{\partial \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{n}_k)}{\partial \boldsymbol{x}_k}\right|_{\bar{\boldsymbol{x}}_k, 0} = \begin{bmatrix} \frac{\partial g_1}{\partial x_k} & \frac{\partial g_1}{\partial y_k} & \frac{\partial g_1}{\partial \theta_k} \\ \frac{\partial g_2}{\partial x_k} & \frac{\partial g_2}{\partial y_k} & \frac{\partial g_2}{\partial \theta_k} \end{bmatrix}$$

Where $\boldsymbol{g} = [g_1\ g_2]^T = \left[r_k^l\ \phi_k^l\right]^T$. Start by taking the derivatives with respect to the range equation:

$$\frac{\partial g_1}{\partial x_k} = \frac{\partial}{\partial x_k}\left(\sqrt{(x_l - x_k - dcos\theta_k)^2 + (y_l - y_k - dsin\theta_k)^2}\right)$$

We can make the following substitutions:

$$u(x_k) = (x_l - x_k - dcos\theta_k)$$

$$v(u) = u^2 + (y_l - y_k - dsin\theta_k)^2$$

$$g_1 = \sqrt{v}$$

Invoking the chain rule, we have:

$$\frac{\partial g_1}{\partial x_k} = \frac{\partial g_1}{\partial v}\frac{\partial v}{\partial u}\frac{\partial u}{\partial x_k} = \left(\frac{1}{2\sqrt{v}}\right)(2u)(-1) = -\frac{u}{\sqrt{v}}$$

$$\frac{\partial g_1}{\partial x_k} = -\frac{(x_l - x_k - dcos\theta_k)}{\sqrt{(x_l - x_k - dcos\theta_k)^2 + (y_l - y_k - dsin\theta_k)^2}}$$

The rest of the derivatives are calculated using the chain rule in a similar manner. The results are:

$$\frac{\partial g_1}{\partial y_k} = -\frac{(y_l - d\cos(\theta_k) - y_k)}{\sqrt{(x_l - x_k - dcos\theta_k)^2 + (y_l - y_k - dsin\theta_k)^2}}$$

$$\frac{\partial g_1}{\partial \theta_k} = \frac{d\big(sin\theta_k(x_k - x_l - dcos\theta_k) - cos\theta_k(y_l - y_k - sin\theta_k)\big)}{\sqrt{(x_l - x_k - dcos\theta_k)^2 + (y_l - y_k - dsin\theta_k)^2}}$$

$$\frac{\partial g_2}{\partial x_k} = \frac{(y_l - y_k - sin\theta_k)}{(x_l - x_k - dcos\theta_k)^2 + (y_l - y_k - dsin\theta_k)^2}$$

$$\frac{\partial g_2}{\partial y_k} = -\frac{(x_l - x_k - cos\theta_k)}{(x_l - x_k - dcos\theta_k)^2 + (y_l - y_k - dsin\theta_k)^2}$$

$$\frac{\partial g_2}{\partial \theta_k} = -\frac{dsin\theta_k(y_l - y_k - sin\theta_k)}{(x_l - x_k - dcos\theta_k)^2 + (y_l - y_k - dsin\theta_k)^2} - \frac{dcos\theta_k(x_l - x_k - cos\theta_k)}{(x_l - x_k - dcos\theta_k)^2 + (y_l - y_k - dsin\theta_k)^2} - 1$$

In addition, we require the following Jacobian for the process noise model:

$$\boldsymbol{n_k^{l\,\prime}} = \left.\frac{\partial \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{n_k^l})}{\partial \boldsymbol{n_k^l}}\right|_{\check{x},0} \boldsymbol{n_k^l}$$

$$\left.\frac{\partial \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{n_k^l})}{\partial \boldsymbol{n_k^l}}\right|_{\check{x},0} = \begin{bmatrix} \frac{\partial g_1}{\partial n_1} & \frac{\partial g_1}{\partial n_2} \\ \frac{\partial g_2}{\partial n_1} & \frac{\partial g_2}{\partial n_2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Taking the variance, we have:

$$Var\left[\boldsymbol{n_k^{l\,\prime}}\right] = \boldsymbol{R_k^{l\,\prime}} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_b^2 \end{bmatrix}$$

3) The original equations of the predictor step of the Kalman filter remain unchanged:

$$\breve{\boldsymbol{P}} = \boldsymbol{F_{k-1}}\widehat{\boldsymbol{P}}_{\boldsymbol{k-1}}\boldsymbol{F_{k-1}^T} + \boldsymbol{Q_k'}$$

$$\breve{\boldsymbol{x}} = f(\widehat{\boldsymbol{x}}_{\boldsymbol{k-1}}, \boldsymbol{v_k}, \boldsymbol{0})$$

For each observed landmark, l, we will need to calculate the Jacobian of the observation model $\boldsymbol{G_k^l}$, the and the innovation $\left(y_k^l - g(\breve{x}, 0)\right)$. The equations are then altered by stacking the matrices for each landmark on top of each other as follows:

$$\boldsymbol{G_k} = \begin{bmatrix} \boldsymbol{G_k^1} \\ \boldsymbol{G_k^2} \\ \dots \\ \boldsymbol{G_k^l} \end{bmatrix}$$

Each matrix G_k^l is 2x3, so G_k is 2lx3.

$$\boldsymbol{y_k} - \boldsymbol{g}(\breve{\boldsymbol{x}}, \boldsymbol{0}) = \begin{bmatrix} \boldsymbol{y_k^1} - \boldsymbol{g}(\breve{\boldsymbol{x}}, \boldsymbol{0}) \\ \boldsymbol{y_k^2} - \boldsymbol{g}(\breve{\boldsymbol{x}}, \boldsymbol{0}) \\ \dots \\ \boldsymbol{y_k^l} - \boldsymbol{g}(\breve{\boldsymbol{x}}, \boldsymbol{0}) \end{bmatrix}$$

Each innovation is 2x1, the stacked innovations are 2lx1.

Additionally, the measurement noise matrix $\boldsymbol{R_k'}$ is expanded:

$$\boldsymbol{R_k'} = \begin{bmatrix} \boldsymbol{R_k^{1\prime}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R_k^{2\prime}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{R_k^{l\prime}} \end{bmatrix}$$

Each $\boldsymbol{R_k^{l\prime}}$ is 2x2, so $\boldsymbol{R_k'}$ is 2lx2l.

Using these alterations to account for multiple measurements, the remaining formulas are:

$$\boldsymbol{K_k} = \breve{\boldsymbol{P}}_{\boldsymbol{k}}\boldsymbol{G_k^T}\left(\boldsymbol{G_k}\breve{\boldsymbol{P}}_{\boldsymbol{k}}\boldsymbol{G_k^T} + \boldsymbol{R_k'}\right)$$

$$\widehat{\boldsymbol{P}}_{\boldsymbol{k}} = (\boldsymbol{1} - \boldsymbol{K_k}\boldsymbol{G_k})\breve{\boldsymbol{P}}_{\boldsymbol{k}}$$

$$\widehat{\boldsymbol{x}}_{\boldsymbol{k}} = \breve{\boldsymbol{x}}_{\boldsymbol{k}} + \boldsymbol{K_k}\left(\boldsymbol{y_k} - \boldsymbol{g}(\breve{\boldsymbol{x}}, \boldsymbol{0})\right)$$

Where $\boldsymbol{K_k}$ is 3x2l, $\widehat{\boldsymbol{P}}_{\boldsymbol{k}}$ is 3x3, and $\widehat{\boldsymbol{x}}_{\boldsymbol{k}}$ is 3x1.

4)

a) The required plots are displayed below.

$$r_{max} = 5$$

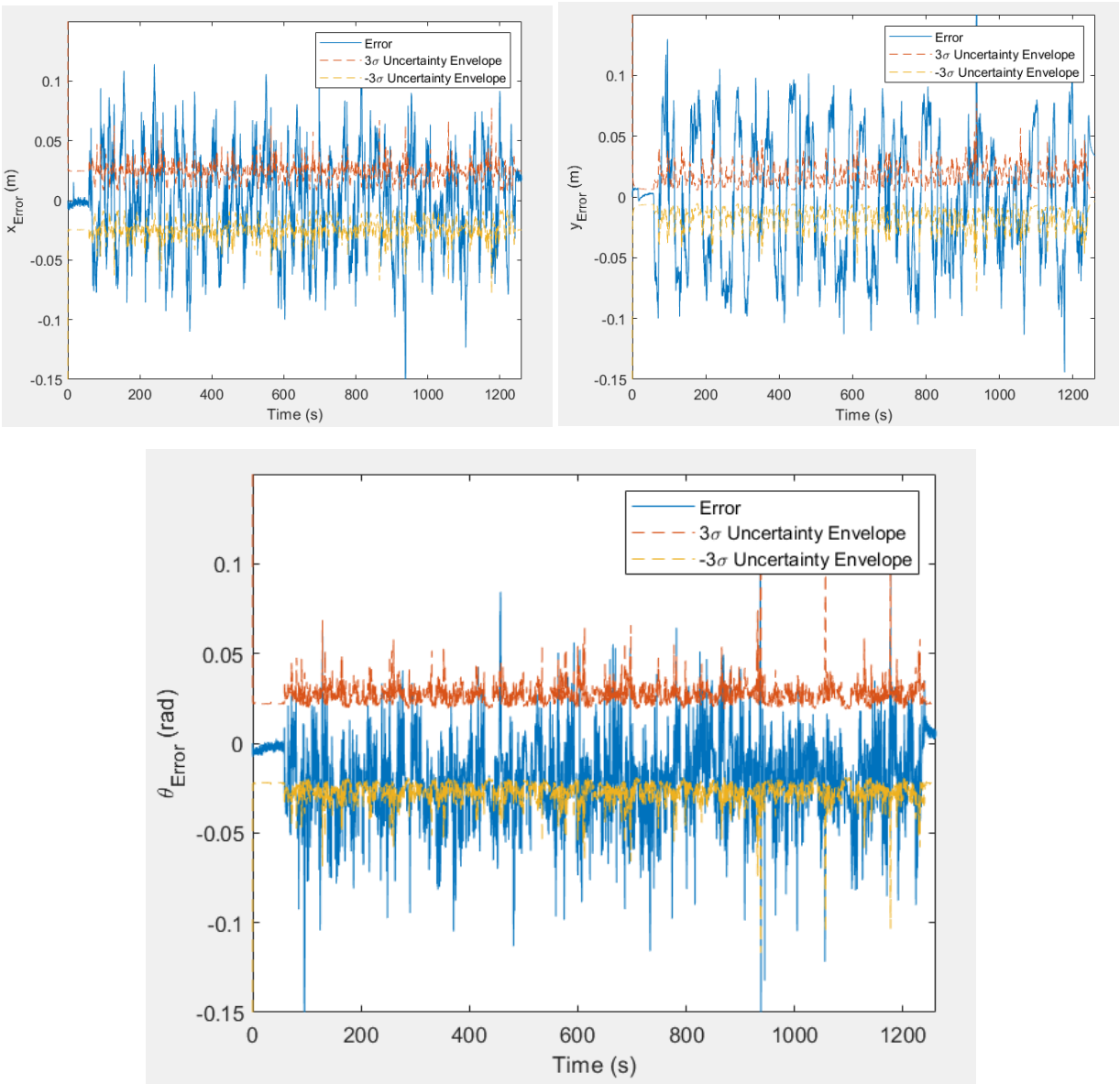


Figure 1: Error plots for x (top left), y (top right), and θ (bottom) using a maximum landmark range threshold of r_{max} = 5m.

$$r_{max} = 3$$

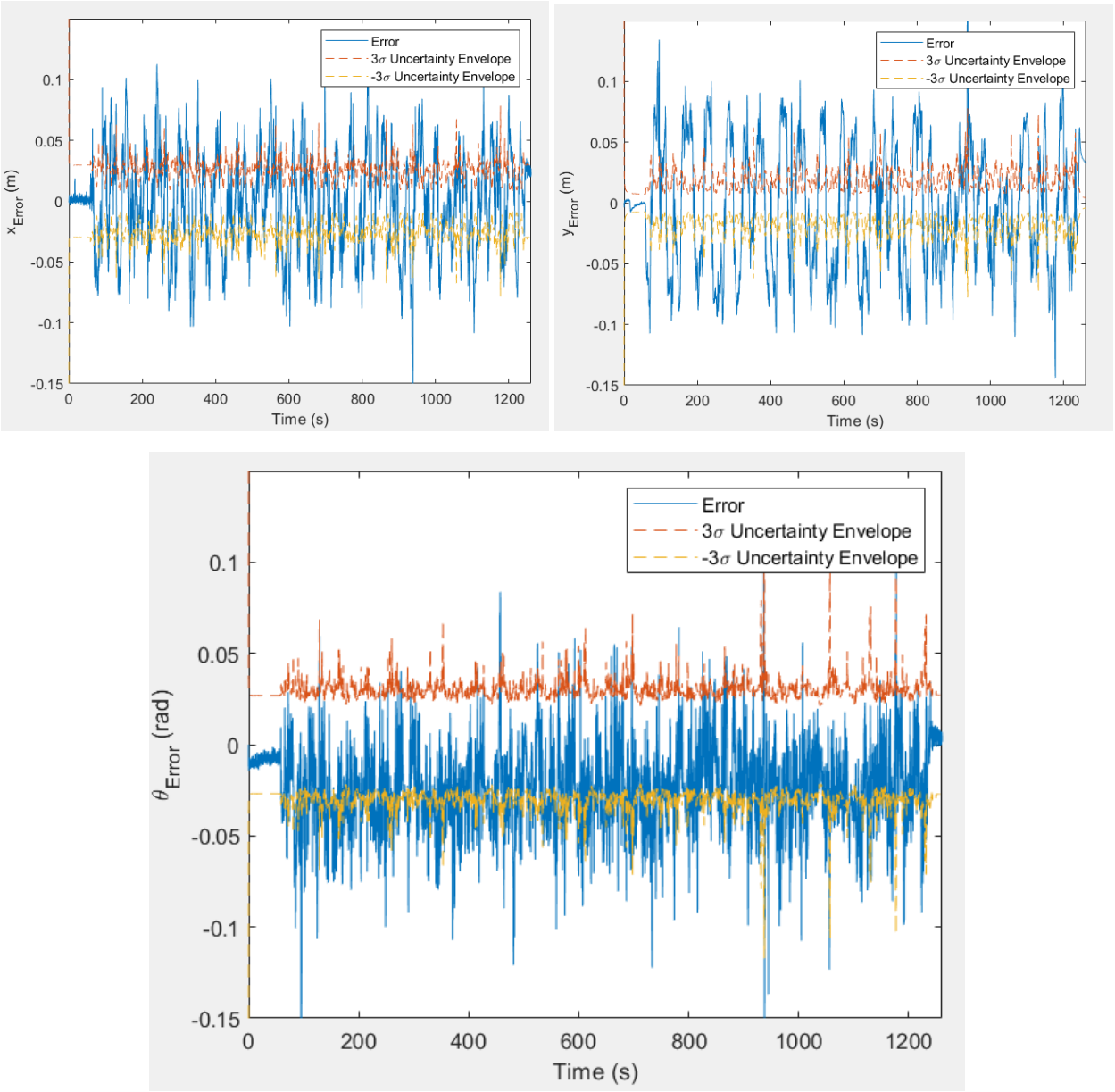


Figure 2: Error plots for x (top left), y (top right), and θ (bottom) using a maximum landmark range threshold of r_{max} = 3m.

$$r_{max} = 1m$$

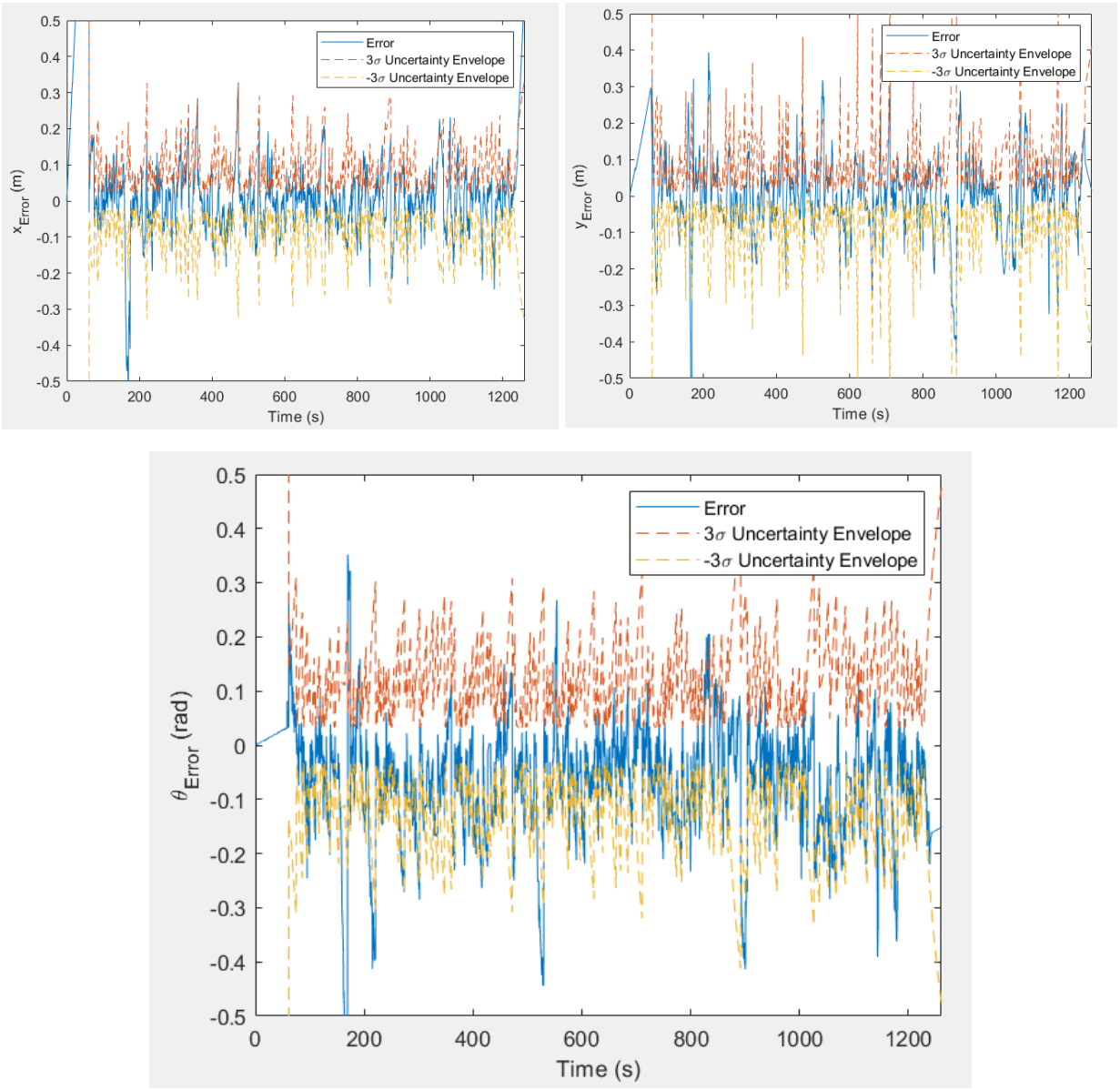


Figure 3: Error plots for x (top left), y (top right), and θ (bottom) using a maximum landmark range threshold of r_{max} = 1m.

Comments: Overall, the plots demonstrate that the results achieved by the estimator are reasonable (error ~0.2 [m or rad]). The x and y error tends to oscillate outside of the uncertainty envelope, indicating that the estimator may be overcorrecting based on the measurements or that the uncertainty is underestimated. The effect of reducing the range threshold is clearly demonstrated. Using a lower threshold reduces the average error, but also induces some outliers. This is attributed to the fact that when there are no landmarks within 1m of the robot the estimator is forced to use dead reckoning such that the error grows unbound until a new landmark within 1m is observed. This can be viewed in the attached movie file, where the ellipse grows until the robot is within 1m of a landmark.

b) The estimator was able to converge for incredibly poor initial positions. The estimator still converges approximately halfway through its route with an initial guess of $\hat{x}_0 = (10e100, 10e100, 0.1)$.

$$r_{max} = 5$$

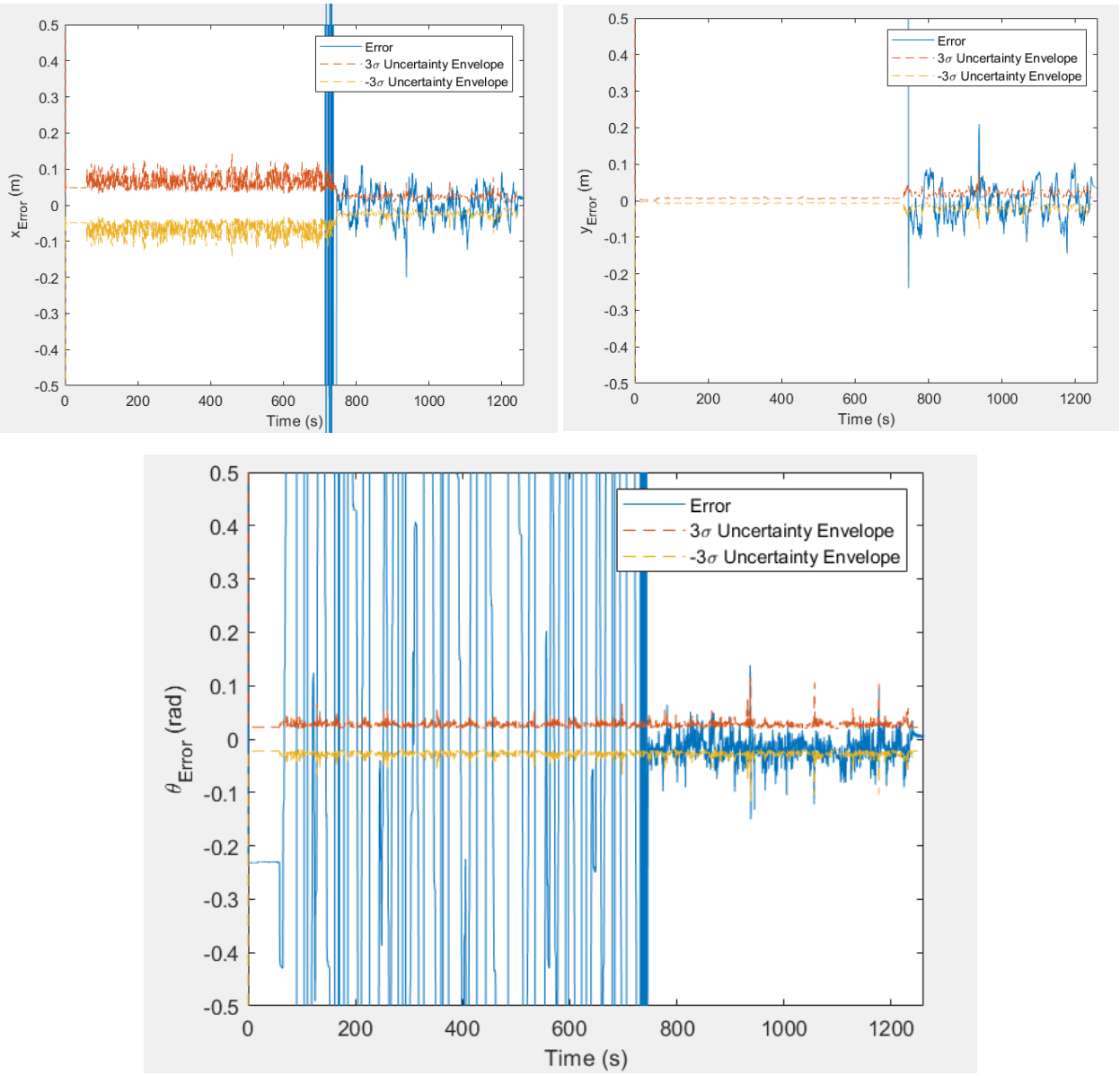


Figure 4: Error plots for x (top left), y (top right), and θ (bottom) using a maximum landmark range threshold of r_{max} = 5m. The initial condition was pushed to (10e100, 10e100, 0.1) and the estimator still converges to the true value.

$$r_{max} = 3$$

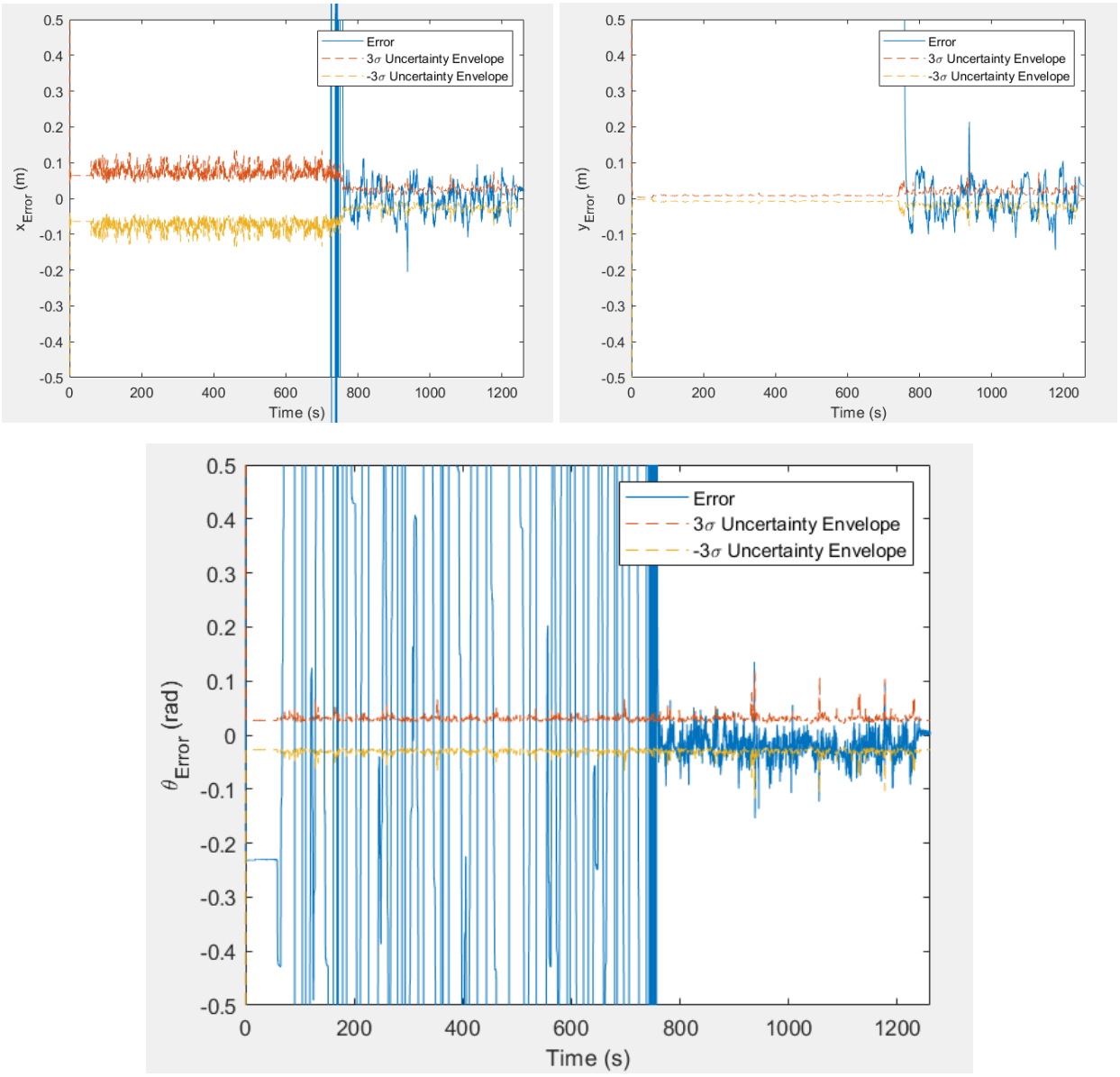


Figure 5: Error plots for x (top left), y (top right), and θ (bottom) using a maximum landmark range threshold of r_{max} = 1m. The initial condition was pushed to (10e100, 10e100, 0.1) and the estimator still converges to the true value.

$$r_{max} = 1$$

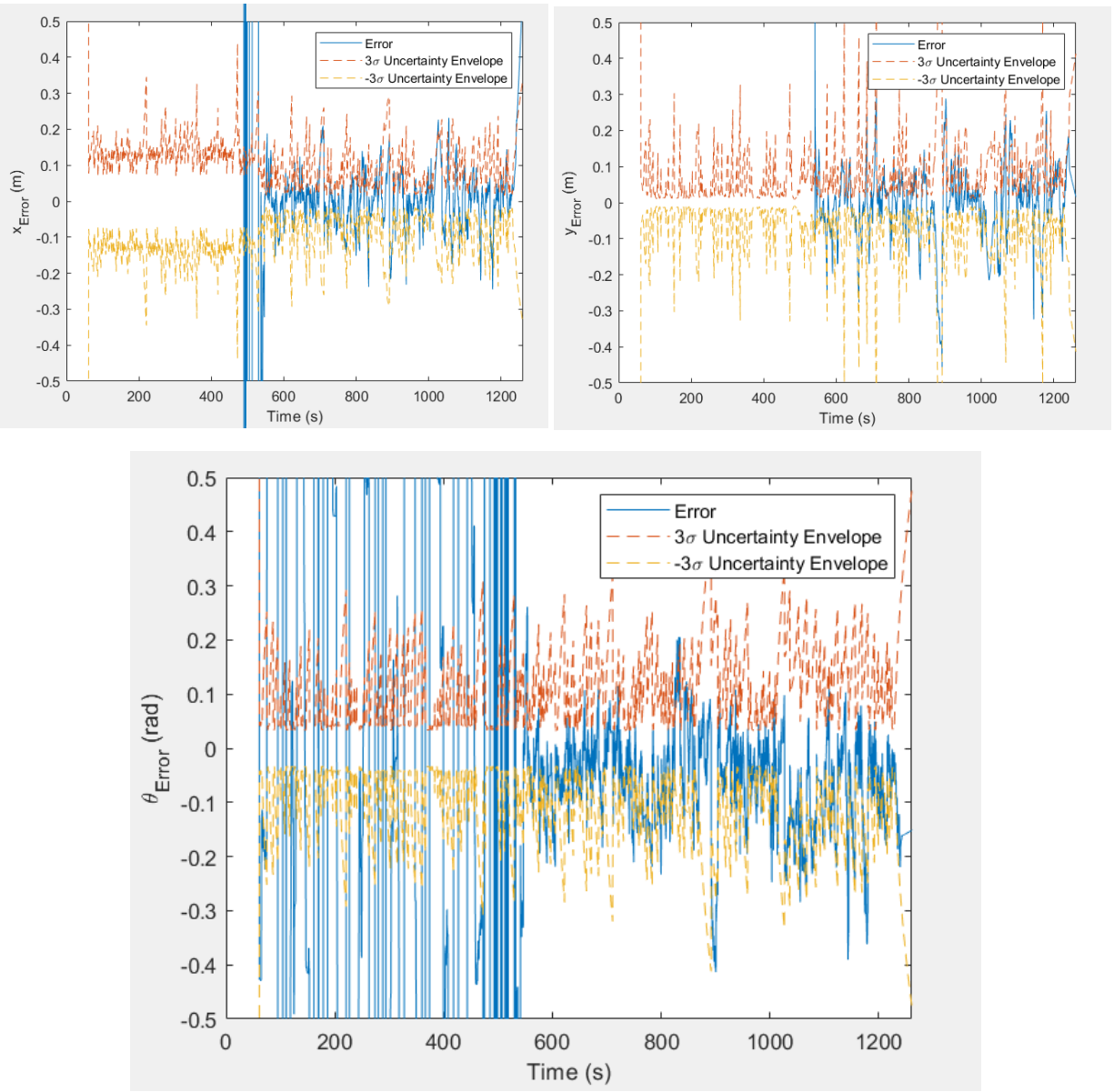


Figure 6: Error plots for x (top left), y (top right), and θ (bottom) using a maximum landmark range threshold of r_{max} = 1m. The initial condition was pushed to (10e100, 10e100, 0.1) and the estimator still converges to the true value. Limiting r_{max} to 1m has actually enabled the estimator to converge sooner than larger values.

Comments: These results were incredibly interesting and demonstrated the capabilities of the EKF. Pushing the initial condition as far as 10e100 away from the initial condition still allowed the EKF to converge using landmark measurements, due to the large size of the innovations. Reducing the range threshold to 1m had the interesting impact of converging the estimate approximately 200 seconds earlier.

c) The true state values were substituted into the Jacobians Q'_k, F'_{k-1}, and G'_k. The results are displayed below for various values of r_{max}.

$$r_{max} = 5$$

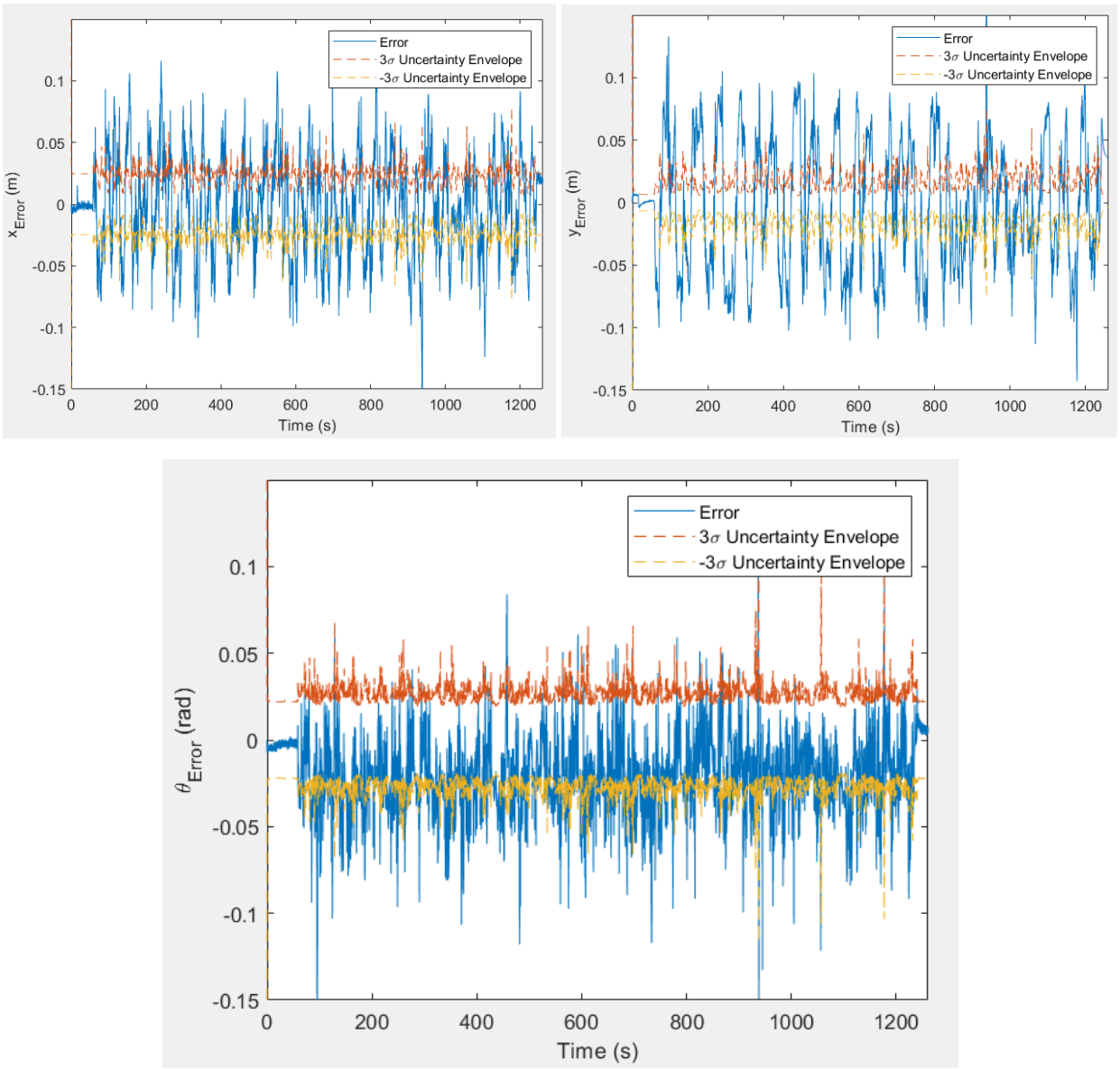


Figure 7: Error plots for x (top left), y (top right), and θ (bottom) using a maximum landmark range threshold of r_{max} = 5m. The initial condition was set to the true initial state and the true state values were used to evaluate the Jacobians of the EKF.

$$r_{max} = 3$$

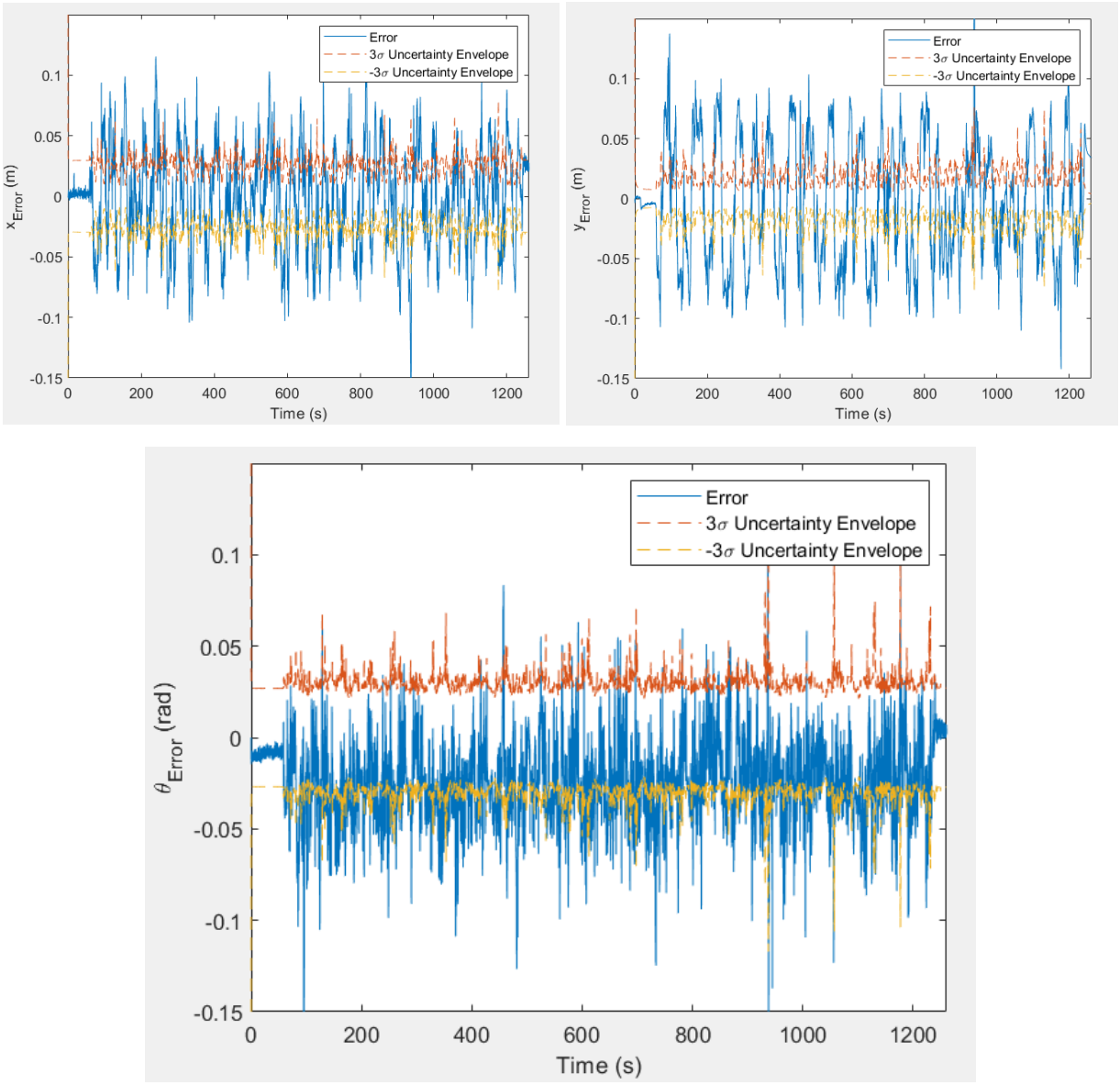


Figure 8: Error plots for x (top left), y (top right), and θ (bottom) using a maximum landmark range threshold of r_{max} = 3m. The initial condition was set to the true initial state and the true state values were used to evaluate the Jacobians of the EKF.

$$r_{max} = 1$$

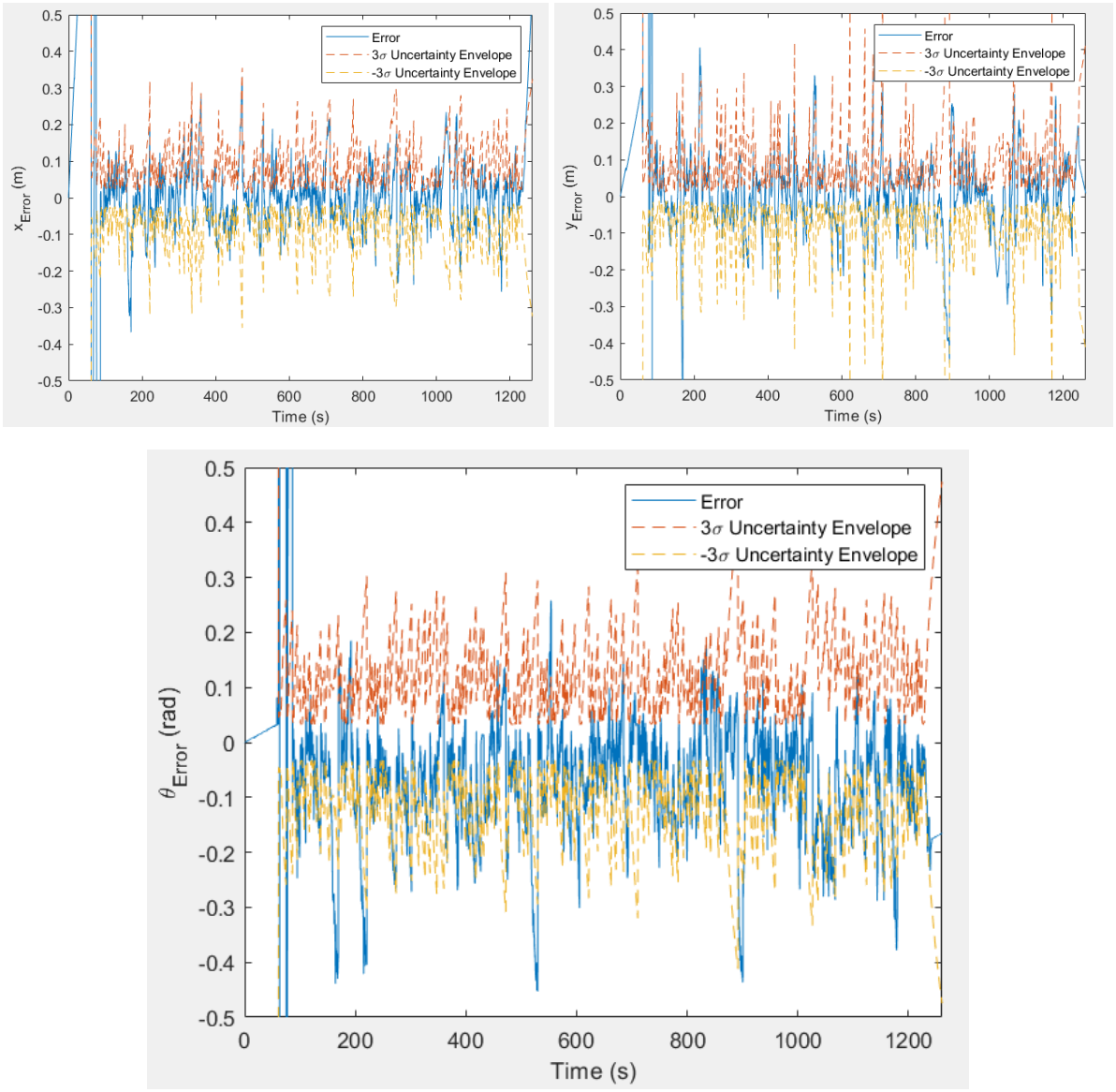


Figure 9: Error plots for x (top left), y (top right), and θ (bottom) using a maximum landmark range threshold of r_{max} = 1m. The initial condition was set to (1, 1, 0.1) and the true state values were used to evaluate the Jacobians of the EKF.

Comments: When contrasting the Cramer-Rao Lower Bound EKF results with part (a), we actually notice that they're fairly similar. The interpretation of this may be that for this particular problem, our EKF estimator is operating near the CRLB in part (a). For a larger range threshold (r_{max} = 5m), we are able to encompass more measurements into our correction step. This has the impact of shrinking uncertainty on the estimate, verifying the assertion made by the CRLB.

AER1513 Assignment 2

Sam Weinberg

```
clc
close all
clear all

load dataset2

% Sampling period
T = 0.1;

% Data length
K = length(v);

% User defined Range Threshold
r_max = 1;

% Intialize matrices
state_hat = zeros(K,3);
x_hat = zeros(K,1);
y_hat = zeros(K,1);
theta_hat = zeros(K,1);
P_hat = zeros(3,3,K);

% Initial values
state_hat(1,:) = [10e100 10e100 0.1].';
x_hat(1) = state_hat(1,1);
y_hat(1) = state_hat(1,2);
theta_hat(1) = state_hat(1,3);
P_hat(:,:,1) = diag([1, 1, 0.1]);

for k = 2:1:K

    % Define process noise matrix
    Q = [v_var*(T*cos(theta_hat(k - 1)))^2, v_var*(T^2*cos(theta_hat(k
 - 1))*sin(theta_hat(k - 1))), 0;
         v_var*(T^2*cos(theta_hat(k - 1))*sin(theta_hat(k - 1))),
 v_var*(T*sin(theta_hat(k - 1)))^2, 0;
         0, 0, om_var*T^2];

    % Jacobian motion model
    F = [1 0 -T*v(k)*sin(theta_hat(k - 1));
         0 1 T*v(k)*cos(theta_hat(k - 1));
         0 0 1];

    % Predictor
    P_check = F*P_hat(:,:,k - 1)*F.' + Q;
    x_check = x_hat(k - 1) + T*v(k)*cos(theta_hat(k - 1));
    y_check = y_hat(k - 1) + T*v(k)*sin(theta_hat(k - 1));
    theta_check = theta_hat(k - 1) + T*om(k);
```

```
    state_check = [x_check, y_check, theta_check].';

    % For each landmark
    cnt = 0;
    G = zeros(2,3);
    innov = zeros(2,1);
    R = zeros(2,2);

    for i = 1:1:17
        if r(k,i) ~= 0 && r(k,i) < r_max

            cnt = cnt + 1;

            % Jacobian observation model
            dg1_dx = -(l(i,1) - x_check - d*cos(theta_check))/
((l(i,1) - x_check - d*cos(theta_check))^2 + (l(i,2) - y_check -
 d*sin(theta_check))^2)^(1/2);
            dg1_dy = -(l(i,2) - y_check - d*sin(theta_check))/
((l(i,1) - x_check - d*cos(theta_check))^2 + (l(i,2) - y_check -
 d*sin(theta_check))^2)^(1/2);
            dg1_dtheta = d*(sin(theta_check)*(l(i,1) - x_check
 - d*cos(theta_check)) - cos(theta_check)*(l(i,2) - y_check -
 d*sin(theta_check)))/((l(i,1) - x_check - d*cos(theta_check))^2 +
 (l(i,2) - y_check - d*sin(theta_check))^2)^(1/2);

            dg2_dx = (l(i,2) - y_check - d*sin(theta_check))/
((l(i,1) - x_check - d*cos(theta_check))^2 + (l(i,2) - y_check -
 d*sin(theta_check))^2);
            dg2_dy = -(l(i,1) - x_check - d*cos(theta_check))/
((l(i,1) - x_check - d*cos(theta_check))^2 + (l(i,2) - y_check -
 d*sin(theta_check))^2);
            dg2_dtheta = -d*sin(theta_check)*(l(i,2)
 - y_check - d*sin(theta_check))/((l(i,1) - x_check -
 d*cos(theta_check))^2 + (l(i,2) - y_check - d*sin(theta_check))^2)
 - d*cos(theta_check)*(l(i,1) - x_check - d*cos(theta_check))/
((l(i,1) - x_check - d*cos(theta_check))^2 + (l(i,2) - y_check -
 d*sin(theta_check))^2) - 1;

            G(2*cnt - 1:2*cnt, 1:3) = [dg1_dx dg1_dy dg1_dtheta;
            dg2_dx dg2_dy dg2_dtheta];

            % Define measurement noise matrix
            R(2*cnt - 1:2*cnt, 2*cnt - 1:2*cnt) = diag([r_var,
 b_var]);

            % Calculate innovation
            r_pred = ((l(i,1) - x_check - d*cos(theta_check))^2 +
 (l(i,2) - y_check - d*sin(theta_check))^2)^(1/2);
            b_pred = atan2(l(i,2) - y_check - d*sin(theta_check),
 l(i,1) - x_check - d*cos(theta_check)) - theta_check;

            if b_pred > pi
                b_pred = b_pred - 2*pi;
```

```
            elseif b_pred < -pi
                b_pred = b_pred + 2*pi;
            end

            innov(2*cnt - 1:2*cnt, 1) = [r(k,i); b(k,i)] - [r_pred;
 b_pred];
        end
    end

    % Kalman Gain
    K = P_check*G.'/(G*P_check*G.' + R);

    % Corrector
    if cnt == 0
        P_hat(:,:,k) = P_check;
        state_hat(k,:) = state_check;
        x_hat(k) = state_hat(k,1);
        y_hat(k) = state_hat(k,2);
        theta_hat(k) = state_hat(k,3);
    else
        P_hat(:,:,k) = (eye(3) - K*G)*P_check;
        state_hat(k,:) = state_check + K*innov;
        x_hat(k) = state_hat(k,1);
        y_hat(k) = state_hat(k,2);
        theta_hat(k) = state_hat(k,3);
    end

    % Keep theta between -pi and pi
    theta_hat(k) = wrapToPi(theta_hat(k));

end

% Extract covariances
for n = 1:1:length(x_true)
    x_var(n) = P_hat(1,1,n);
    y_var(n) = P_hat(2,2,n);
    theta_var(n) = P_hat(3,3,n);
end

% Error Plots
figure(1)
plot(t, x_hat - x_true)
hold on
plot(t, 3*sqrt(x_var), '--')
hold on
plot(t, -3*sqrt(x_var), '--')
axis([0 max(t) -0.5 0.5])
xlabel("Time (s)")
ylabel("x_E_r_r_o_r (m)")
legend("Error","3\sigma Uncertainty Envelope", "-3\sigma Uncertainty
 Envelope")
hold off

figure(2)
```

```
plot(t, y_hat - y_true)
hold on
plot(t, 3*sqrt(y_var), '--')
hold on
plot(t, -3*sqrt(y_var), '--')
axis([0 max(t) -0.5 0.5])
xlabel("Time (s)")
ylabel("y_E_r_r_o_r (m)")
legend("Error","3\sigma Uncertainty Envelope", "-3\sigma Uncertainty
 Envelope")
hold off

figure(3)
plot(t, wrapToPi(theta_hat - th_true))
hold on
plot(t, 3*sqrt(theta_var), '--')
hold on
plot(t, -3*sqrt(theta_var), '--')
axis([0 max(t) -0.5 0.5])
xlabel("Time (s)")
ylabel("\theta_E_r_r_o_r (rad)")
legend("Error","3\sigma Uncertainty Envelope", "-3\sigma Uncertainty
 Envelope")
hold off

% Animation
h = figure(4);
axis tight manual
filename = 'assignment2.gif';
plot(l(:,1), l(:,2),'.k')
hold on
est = animatedline('Marker','o','LineStyle','None','Color','red');
tru = animatedline('Marker','o','LineStyle','None','Color','blue');
elip = animatedline('Marker','.','LineStyle','None','Color','red');
axis([-2 10 -3 3])

for m = 1:1:length(x_true)

    % Create uncertainty ellipse
    s = -2 * log(1 - 0.99); % 0.99 for 3*sigma

    [V, D] = eig(P_hat(1:2,1:2,m) * s);

    t = linspace(0, 2 * pi);
    a = (V * sqrt(D)) * [cos(t(:))'; sin(t(:))'];

    clearpoints(elip)
    clearpoints(est)
    clearpoints(tru)

    addpoints(elip, a(1, :) + x_hat(m), a(2, :) + y_hat(m));
    addpoints(est,x_hat(m),y_hat(m))
    addpoints(tru,x_true(m),y_true(m))
```

```
        drawnow limitrate

        % Capture the plot as an image
        frame = getframe(h);
        im = frame2im(frame);
        [imind,cm] = rgb2ind(im,256);
        % Write to the GIF File
        if m == 1
            imwrite(imind,cm,filename,'gif', 'Loopcount',inf);
        elseif mod(m,2) == 0
            imwrite(imind,cm,filename,'gif','WriteMode','append');
        end

    end
```

Published with MATLAB® R2018a