Mobile Robotics
HW-2

Task 1

A. **EKF Derivation.**

Velocity Motion Model: $X_t = g(X_{t-1}, u_t)$.

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} -\hat{v}/\hat{\omega}\sin(\theta_{t-1}) + \hat{v}/\hat{\omega}\sin \\ \\ \end{bmatrix}$$

X_t : State vector $\quad X_t = (x_t \; y_t \; \theta_t)^T$

u_t : Control Input $\quad u_t = (v_t \; \omega_t \; \gamma_t)^T$

Motion model: $X_t = g(X_{t-1}, u_t)$.

Velocity Motion model:

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} -v_t/\omega_t \sin(\theta_{t-1}) + v_t/\omega_t \sin(\theta_{t-1} + \omega_t \Delta t) \\ v_t/\omega_t \cos(\theta_{t-1}) - v_t/\omega_t \cos(\theta_{t-1} + \omega_t \Delta t) \\ \omega_t \Delta t + \gamma_t \Delta t \end{bmatrix}$$

G_t : Jacobian of model w.r.t state.

$$[G_t]_{ij} = \frac{\partial (X_t)_i}{\partial (X_{t-1})_j} = \frac{\partial g_i}{\partial x_j}.$$

$$G = \begin{bmatrix} 1 & 0 & -v_t/\omega_t \cos(\theta_{t-1}) + v_t/\omega_t \cos(\theta_{t-1} + \omega \Delta t) \\ 0 & 1 & -v_t/\omega_t \sin(\theta_{t-1}) + v_t/\omega_t \sin(\theta_{t-1} + \omega_t \Delta t) \\ 0 & 0 & 1 \end{bmatrix}$$

V_t : Jacobian of model w.r.t. input

$$[V_t]_{ij} = \frac{\partial (X_t)_i}{\partial (u_t)_j} = \frac{\partial g_i}{\partial u_j}$$

$$V = \begin{bmatrix} (-\sin\theta + \sin(\theta + \omega\Delta t))/\omega & v(\sin\theta - \sin(\theta + \omega\Delta t))/\omega^2 & v\cos(\theta + \omega\Delta t)\Delta t/\omega \\ (\cos\theta - \cos(\theta + \omega\Delta t))/\omega & -v(\cos\theta - \cos(\theta + \omega\Delta t))/\omega^2 & v\sin(\theta + \omega\Delta t)\Delta t/\omega \\ 0 & 0 & 1 \end{bmatrix}$$

There is some noise in the control input. True input $\hat{u}$ is given by:

$$\begin{bmatrix} \hat{v} \\ \hat{\omega} \\ \hat{\gamma} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} v \\ \omega \\ \gamma \end{bmatrix}, \begin{bmatrix} \alpha_1 v^2 + \alpha_2 \omega^2 & 0 & 0 \\ 0 & \alpha_3 v^2 + \alpha_4 \omega^2 & 0 \\ 0 & 0 & \alpha_5 v^2 + \alpha_6 \omega^2 \end{bmatrix} \right).$$

Prediction Step in EKF.

- $\bar{\mu}, \bar{\Sigma}$ are the predicted mean & covariance. We simply use the motion model to find $\bar{\mu}$, and use the linearized model to calculate $\bar{\Sigma}$.

$$\bar{\mu}_t = g(\mu_{t-1}, u_t)$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + V_t M_t V_t^T$$

where M is the noise in the motion command (covariance)

So we've our $\bar{\mu}_t, \bar{\Sigma}_t$ (predicted values).

Sensor Model : $z_t = h(x_t, L_x, L_y)$.

↳ The sensor returns range and bearing values, calculated according to a landmark positioned at (L_x, L_y).

$$\begin{bmatrix} \phi \\ r \end{bmatrix} = \begin{bmatrix} \text{atan2}(L_y - y_t, L_x - x_t) - \theta_t \\ \text{sqrt}((x_t - L_x)^2 + (y_t - L_y)^2) \end{bmatrix}$$

H_t : Jacobian of sensor model w.r.t state

$$[H_t]_{ij} = \frac{\partial z_i}{\partial x_j} \cdot \frac{\partial h_i}{\partial x_j}$$

$$H = \begin{bmatrix} (L_y - y_t)/q & -(L_x - x_t)/q & -1 \\ -(L_x - x_t)/\sqrt{q} & -(L_y - y_t)/\sqrt{q} & 0 \end{bmatrix}, \quad q = (L_x - x_t)^2 + (L_y - y_t)^2$$

There is some noise in the sensor too :

$$\begin{bmatrix} \hat{\phi} \\ \hat{r} \end{bmatrix} \sim N\left(\begin{bmatrix} \phi \\ r \end{bmatrix}, \begin{bmatrix} \sigma_\phi^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix} \right).$$

Correction Step in EKF.

- Here we try to find the real μ and Σ using the previously predicted $\bar{\mu}$ & $\bar{\Sigma}$, and the sensor values.

— Innovation

This is the difference between the predicted & real sensor values

$\bar{z}_t = h(\bar{\mu}_t, L_x, L_y)$.

z_t : the real sensor value.

$v_t = z_t - \bar{z}_t$ (innovation).

— Kalman Gain, (K_t).

Kalman Gain is calculated in the following way:

$S_t = H_t \bar{\Sigma}_t H_t^T + Q_t$, where Q_t is the covariance of sensor noise.

$K_t = \bar{\Sigma}_t H_t^T S_t^{-1}$

— Correction Update

$\mu_t = \bar{\mu}_t + K_t v_t$,

$\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$.

∴ Using both prediction & correction step we find the moments of Gaussian distribution (i.e mean & covariance) at each step.

Underlying assumption is that our distribution is Gaussian (although a gaussian distribution doesn't remain gaussian after passing through a non-linear f^n).

B. UKF Derivation .

— We use the same motion model and sensor model as in EKF.

— We also use the same control and sensor noise (covariance).

We've an augmented state, which includes the zero-mean gaussian control noise and zero-mean gaussian sensor noise as seperate states.

∴ Mean & covariance of augmented state is:

$$\mu^a_{t-1} = \left[\mu^{aT}_{t-1} \quad [0\ 0\ 0]^* \quad [0\ 0]^* \right]^T .$$

$$\Sigma^a_{t-1} = \begin{bmatrix} \Sigma_{t-1} & 0 & 0 \\ 0 & M_t & 0 \\ 0 & 0 & Q_t \end{bmatrix}$$

∴ Augmented state dimension = 8.

$X^a = [X^x , X^v , X^s]$

(X^x: state, X^v: input noise, X^s: sensor noise.)

Sigma points

We generate $(2n+1)$ sigma points to approximate the distribution. Here n = dimension of augmented state.

$$x[0] = \mu$$
$$x[i] = \mu + \sqrt{(n+k)\Sigma} \qquad i = 1, \dots n$$
$$x[i] = \mu - \sqrt{(n+k)\Sigma} \qquad i = n+1, n+2, \dots 2n$$

where k is a parameter. We chose $k = 2$.

$w[0] = k/(n+k)$
$w[i] = 1/2(n+k)$, $i = 1, 2, \dots 2n$
weights

Prediction Step in UKF

Pass all the sigma points through the non-linear motion model. Ofcourse, only (x, y, θ) of the augmented state get updated.

For all sigma-points $x[i]$:

$$\bar{x}_t^x[i] = g(x_{t-1}^x[i], u_t + x_{t-1}^v[i]).$$

Using these, we compute predicted mean & covariance:

$$\bar{\mu}_t = \sum_{i=0}^{2n} w_i \bar{x}_{i,t}^x$$

$\bar{x}_{i,t}$ is same as $x_t[i]$

$$\bar{\Sigma}_t = \sum_{i=0}^{2n} w_i (\bar{x}_{i,t}^x - \bar{\mu}_t)(\bar{x}_{i,t}^x - \bar{\mu}_t)^T$$

Correction Step in UKF

$$\bar{z}_t[i] = h(\bar{\mu}_t, L_x, L_y) + x_{t-1}^v[i]$$
$$\bar{z}_t = \sum_{i=0}^{2n} w_i x_{t-1}^u[i].$$

} Predicted sensor value.

$$\nu_t = z_t - \bar{z}_t \qquad \text{(innovation)}.$$

$$S_t = \sum_{i=0}^{2n} w_i (\bar{z}_{i,t} - z_t)(\bar{z}_{i,t} - z_t)^T$$
$$\Sigma_t^{x,z} = \sum_{i=0}^{2n} w_i (\bar{x}_{i,t}^x - \bar{\mu}_t)(\bar{z}_{i,t} - z_t)^T$$
$$K_t = \Sigma_t^{x,z} S_t^{-1} \qquad \text{(Kalman Gain)}$$

$$\mu_t = \bar{\mu}_t + K_t \nu_t$$
$$\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$$

} Correction update.

~~In the EKF model, we didn't assume our distribution to be gaussian, but essentially~~

C. PF Derivation.

- Particle Filter doesn't assume that the posterior belief is gaussian. It is a non-parametric filter.
- The belief is represented by a set of particles, each particle representing a possible state. Each particle is assigned a weight. Higher the weight, higher is the probability that the particle represents the true state.

Prediction Step in PF.

For all particles $x[i]$:

$$\bar{x}_t[i] = g(x_{t-1}[i], u_t).$$

(u_t — noisy control input (*))

where g is the non-linear motion model.

Correction Step in PF

→ In this we update the weight of each particle, using the sensor readings.

$$\bar{w}_t[i] = w_{t-1}[i] \cdot p(z_t | x_t[i]).$$

$$w_t[i] = \frac{\bar{w}_t[i]}{\sum_{i=1}^{M} \bar{w}_t[i]}$$

$p(z|x)$ can be calculated in MATLAB using mvnpdf.

$\bar{z}_t[i] = h(L_x, L_y, \bar{x}_t[i])$.

$v_t[i] = \bar{z}_t[i] - z_t$ where z_t is the real value

$\gg \bar{w}_t[i] = mvnpdf(v_t[i], [0;0], Q_t)$

(adding zero mean Gaussian Sensor noise)

(*) Noise is added in control input by adding a random vector derived from Cholesky factorization of M_t.

$\gg u = u + chol(M) * randn(3,1)$

(zero-mean Gaussian Control Noise.)

Particle Filter has 2 additional steps, after prediction and correction step.

~~Estimating final mean & variance~~ :

$$\mu_{x,t} = \sum_{i=0}^{M} w_t[i]\, x_t^x[i]$$

$$\mu_{y,t} = \sum_{i=0}^{M} w_t[i]\, x_t^y[i]$$

$$\mu_{\cos\theta} = \sum_{i=0}^{M} w_t[i] \cos(x_t^\theta[i])$$

$$\mu_{\sin\theta} = \sum_{i=0}^{M} w_t[i] \sin(x_t^\theta[i])$$

$$\mu_{\theta,t} = \operatorname{atan2}(\mu_{\sin\theta}, \mu_{\cos\theta})$$

$$\mu_t = [\mu_{x,t} \quad \mu_{y,t} \quad \mu_{\theta,t}]^T$$

$$\Sigma_t =$$

Resampling Step :

Here we use a low variance sampler to sample particles for the next time step. Resampling is done with replacement, and probability of choosing a particle is proportional to its weight.

Algorithm : Low_Variance Sampler (X_t, W_t)

```
X̄_t = ϕ
r = rand(0, 1/M)        // M is no. of particles.
c = w_t[1]
i = 1.
for m = 1 to M do
    U = r + (m-1)/M
    while U > c
        i = i + 1
        c = c + w_t[i].
    endwhile
    add x_t[i] to X̄_t
endfor
return X̄_t
```

Final Mean and Variance.

Final mean and variance is calculated using the resampled particles.

$$\mu_{x,t} = \sum_{i=0}^{M} x_t^x[i]/M \;,\quad \mu_{y,t} = \sum_{i=0}^{M} x_t^y[i]/M.$$

$$s_{cos,t} = \sum_{i=0}^{M} \cos(x_t^\theta[i]) \;,\quad s_{sin,t} = \sum_{i=0}^{M} \sin(x_t^\theta[i])$$

$$\mu_t^\theta = \operatorname{atan2}(s_{sin,t}, s_{cos,t})$$

$$\mu_t = [\mu_t^x, \mu_t^y, \mu_t^\theta]^T$$

$$\Sigma_t = \sum_{i=0}^{M} (x_t[i] - \mu_t)(x_t[i] - \mu_t)^T / M$$

~~for the next step, we can either~~

D. InEKF Derivation.

μ is in $SE(2)$. ie. ~~μ~~

$$\mu = \begin{bmatrix} \cos(\mu_\theta) & -\sin(\mu_\theta) & \mu_x \\ \sin(\mu_\theta) & \cos(\mu_\theta) & \mu_y \\ 0 & 0 & 1 \end{bmatrix}$$

Prediction ~~Propagation~~ Step in InEKF

First we generate the twist matrix

$$\mu_t = [x \;\; y \;\; \theta]^T$$

$$\bar{\mu}_t = g(\mu_t, u) \quad \text{~~(Modeling predicted mean)~~}$$

(calculating predicted mean using N-L motion model).

$$\bar{\mu} = \begin{bmatrix} \cos(\bar{\mu}_{t,\theta}) & -\sin(\bar{\mu}_{t,\theta}) & \bar{\mu}_{t,x} \\ \sin(\bar{\mu}_{t,\theta}) & \cos(\bar{\mu}_{t,\theta}) & \bar{\mu}_{t,y} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\hat{\xi} = \log(\mu \bar{\mu}^{-1})$$

All this was done to generate $\hat{\xi}$ (twist matrix).

Propagation Step in InEKF

$$\xi_{new} = \text{Chol}(Q) \cdot \text{randn}(3,1). \qquad Q = \text{control input cov.}$$

$\xi^\wedge = \xi^\wedge + \xi^\wedge_{Noise}$ (adding noise to the twist matrix).

where $\wedge$ is the wedge operator,

ie. if $\xi = \begin{bmatrix} \omega_{1\times1} \\ v_{2\times1} \end{bmatrix}$ $\xi^\wedge = \begin{bmatrix} \omega^\wedge & v \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -\omega & v_x \\ \omega & 0 & v_y \\ 0 & 0 & 1 \end{bmatrix}$.

Propagation Step
$$\begin{cases} \bar{\mu} = \mu \exp(\xi^\wedge). \\ \bar{\Sigma} = \Sigma + Ad_\mu Q Ad_\mu^T \end{cases}$$

where Ad_μ is the adjoint representation of $SE(2)$ element μ.

$$Ad_\mu = \begin{bmatrix} \cos(\mu_\theta) & -\sin(\mu_\theta) & \mu_y \\ \sin(\mu_\theta) & \cos(\mu_\theta) & -\mu_x \\ 0 & 0 & 1 \end{bmatrix}$$

Correction Step in InEKF.

- We don't use the standard sensor model h
- We've 2 landmarks for sensing.

$$H = \begin{bmatrix} -1 & 0 & L_{y1} \\ 0 & -1 & -L_{x1} \\ - & 0 & L_{y2} \\ 0 & -1 & -L_{x2} \end{bmatrix} \quad b_1 = \begin{bmatrix} L_{x1} \\ L_{y1} \\ 1 \end{bmatrix} \quad b_2 = \begin{bmatrix} L_{x2} \\ L_{y2} \\ 1 \end{bmatrix}.$$

Incorporating sensor noise:

$S_{Noise} = \begin{bmatrix} N & 0 \\ 0 & N \end{bmatrix}$ where N is top 2×2 matrix of $\bar{\mu} \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} \bar{\mu}^T$

Finding Kalman Gain:

$S = H \bar{\Sigma} H^T + S_{Noise}$

$L = \bar{\Sigma} H^T S^{-1}$

Calculating innovation:

$v_c = \begin{bmatrix} v_{1c} \\ v_{2c} \end{bmatrix}$ where v_{ic} is top 2×1 vector of $\bar{\mu} Y_i - b_i$

$v = (L v_c)^\wedge$

Measurement update:

$\mu = \exp(v) \bar{\mu}$

$\Sigma = (I - LH) \bar{\Sigma} (I - LH)^T + L S_{Noise} L^T$

Extended Kalman Filter Plots

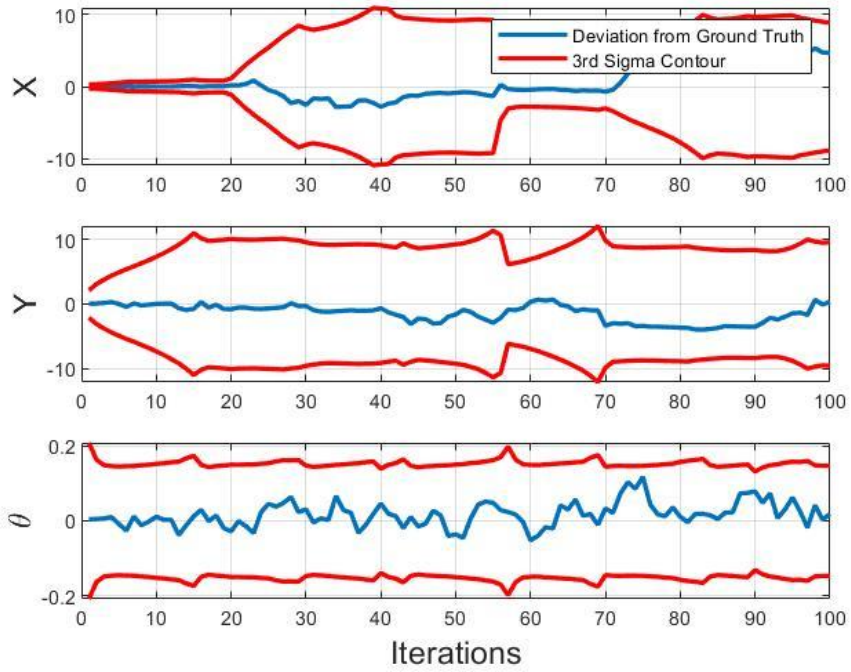

Figure 1: EKF: Deviation and 3-Sigma Countour plot

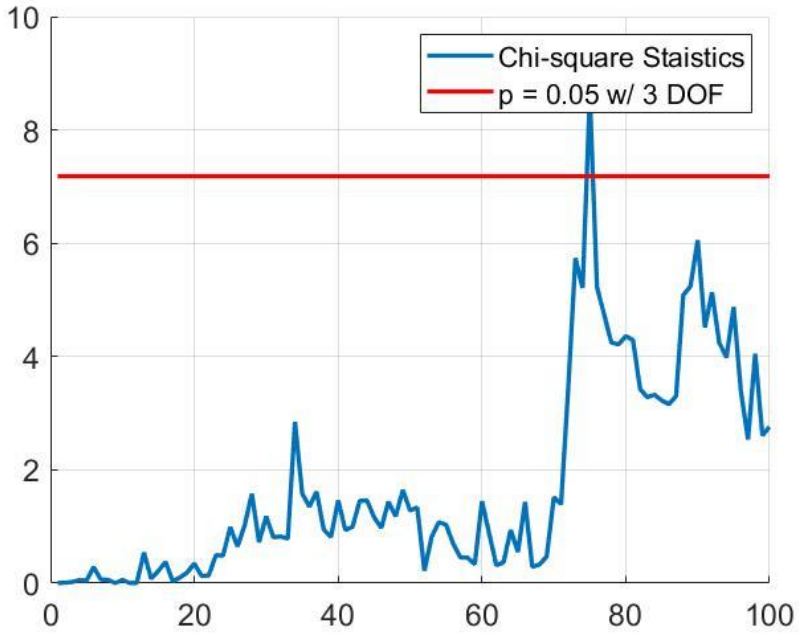

Figure 2: EKF: Chi-square Statistics

Unscented Kalman Filter plots

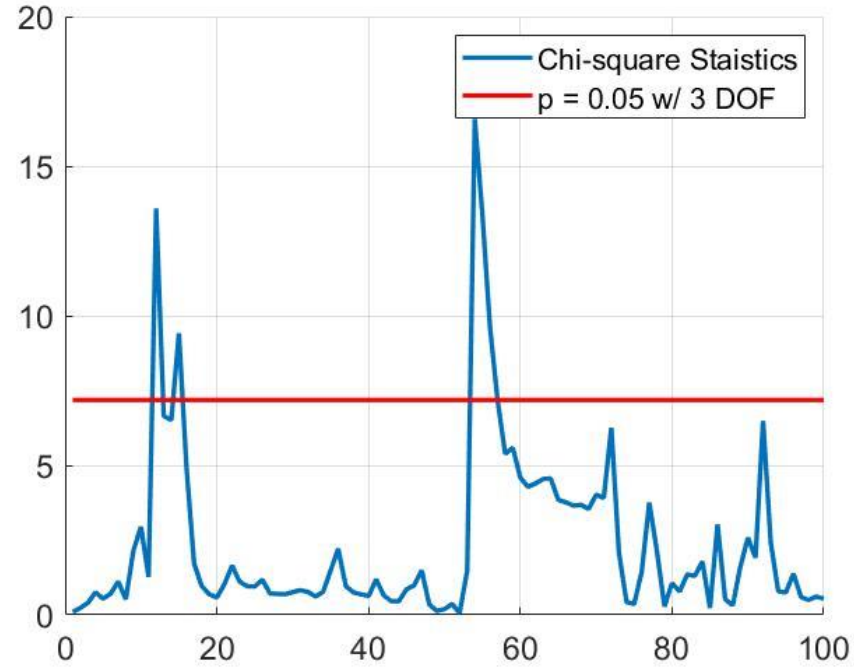

Figure 3: UKF-Chi square Statistics

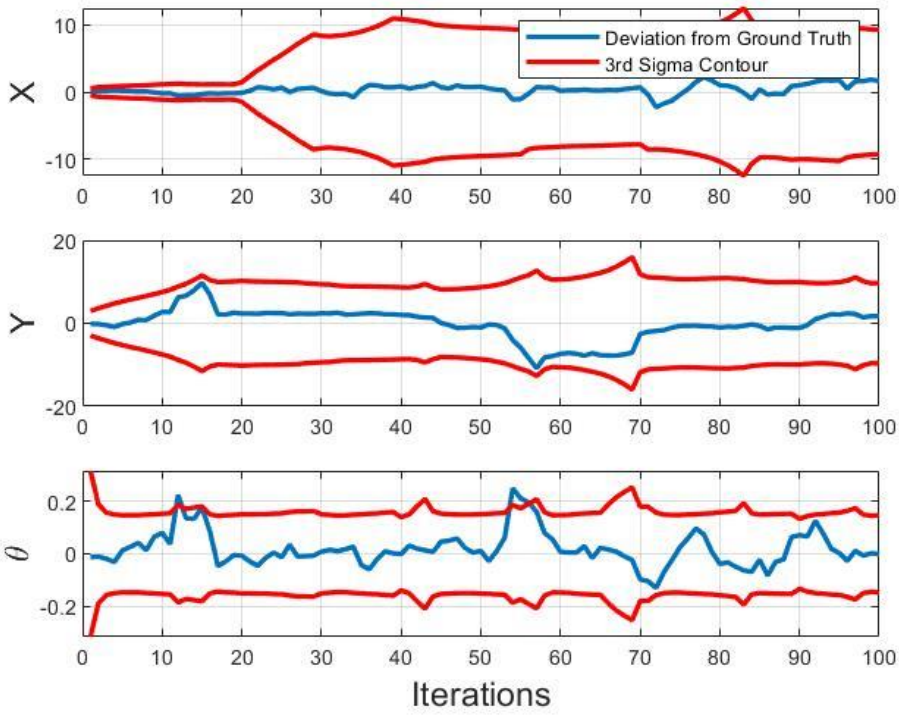

Figure 4: UKF-Deviation and 3-Sigma Plots

Particle Filter

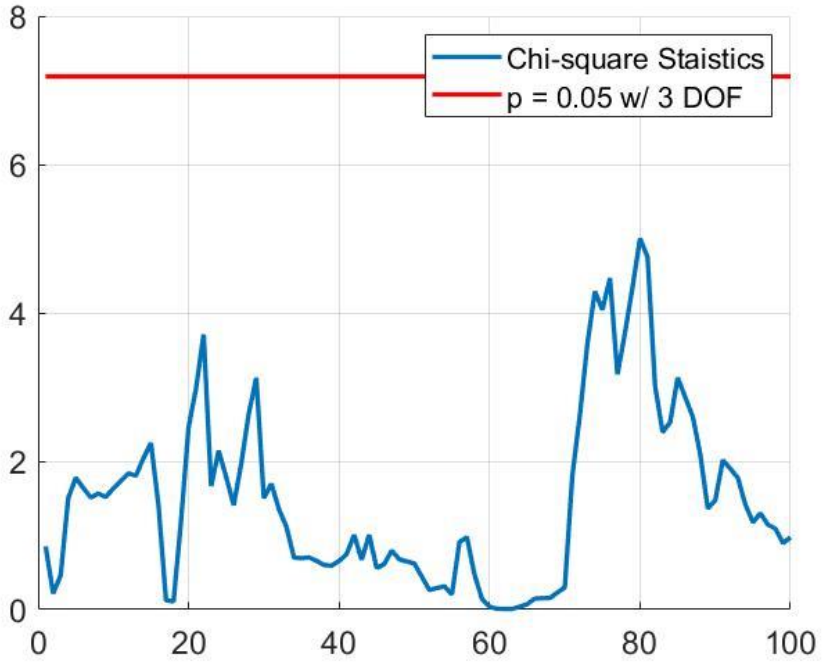

Figure 5: PF-Chi square Statistics

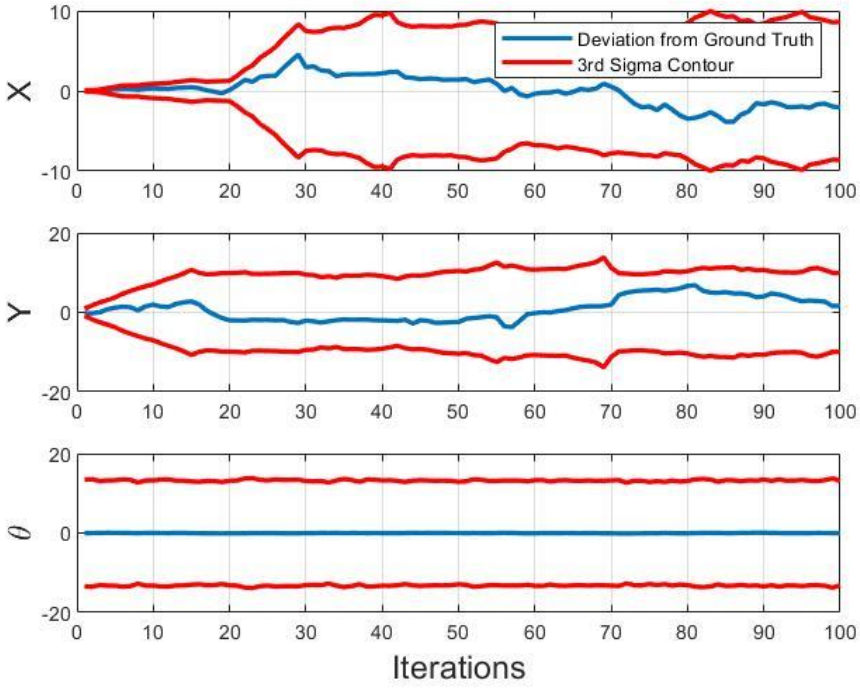

Figure 6: PF-Deviation and 3-Sigma Contours

Right Invariant Extended Kalman Filter

These plots were made using the lieToCartesian.p file

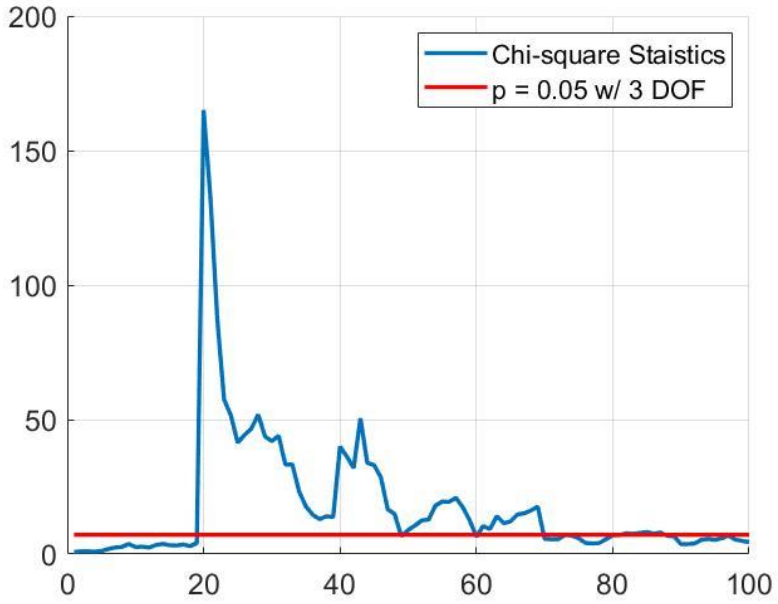

Figure 7: InEKF-Chi square Statistics

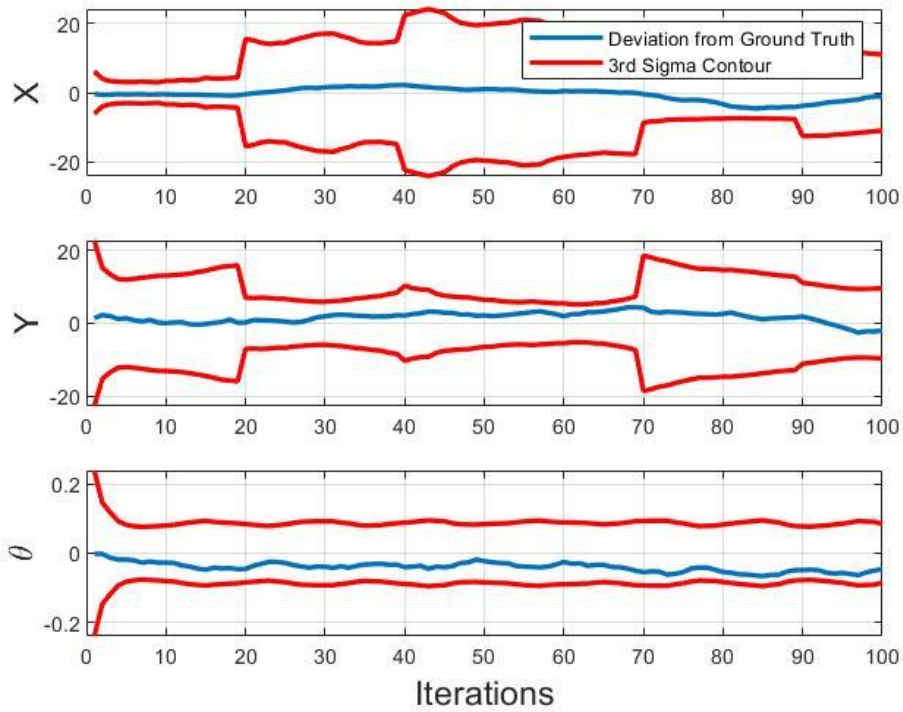

Figure 8: InEKF-Deviation and 3-Sigma Contours