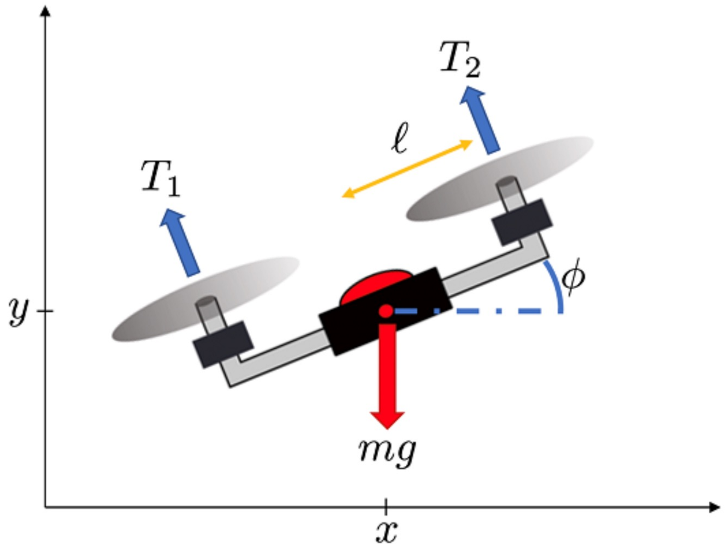
State Estimation of a Planar Quadcopter

Member: Howard Chang, Willian Lin, Joslyn Chen

UNIVERSITY of WASHINGTON

Dynamics Model

$$
\begin{bmatrix} x \\ v_x \\ y \\ v_y \\ \phi \\ \omega \end{bmatrix} = \begin{bmatrix} v_x \\ \dfrac{-(T_1+T_2)\sin\phi - C_D^v v_x}{m} \\ v_y \\ \dfrac{(T_1+T_2)\cos\phi - C_D^v v_y}{m} - g \\ \omega \\ \dfrac{(T_2-T_1)\ell - C_D^\phi \omega}{I_{yy}} \end{bmatrix}
$$



Extended Kalman Filter

Linear

$$
\begin{aligned}
x_t &= A_t x_{t-1} + B_t u_t + \varepsilon_t \; . \\
z_t &= C_t x_t + \delta_t
\end{aligned}
$$

Nonlinear

$$
\begin{aligned}
x_t &= g(u_t, x_{t-1}) + \varepsilon_t \\
z_t &= h(x_t) + \delta_t \; .
\end{aligned}
$$

S. Thrun, Probablistic Robotics, 2005

Extended Kalman Filter – Linearization

A Gaussian projected through g is typically non-Gaussian.

Taylor Expansion (First Order)

$$g'(u_t, x_{t-1}) \quad := \quad \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}}$$

Once g, h is linearized, the mechanics of belief propagation are equivalent to those of the Kalman filter.

S. Thrun, Probablistic Robotics, 2005

Extended Kalman Filter – Algorithm

Algorithm Extended_Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$**):**

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

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

$$K_t = \bar{\Sigma}_t\, H_t^T (H_t\, \bar{\Sigma}_t\, H_t^T + Q_t)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$$

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

return μ_t, Σ_t

S. Thrun, Probablistic Robotics, 2005

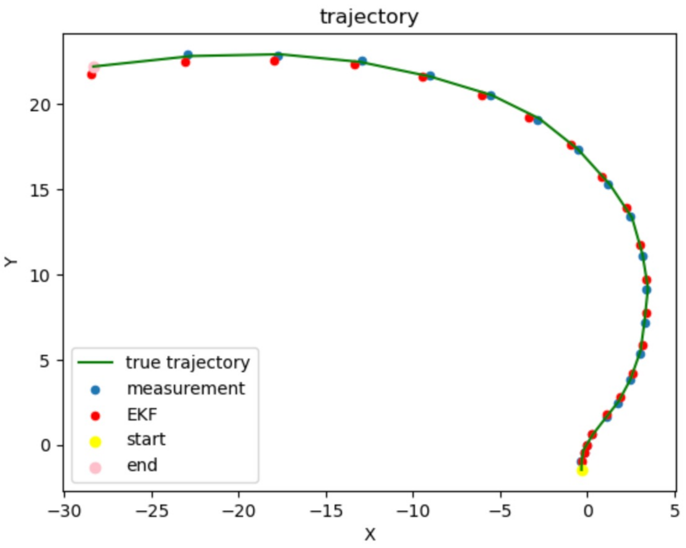
Code

```
def EKF(bot:BasePlanarQuadrotor,nominal_xs_us,control_sequence,x0,sigma,Q,H,R,N,dt):
    m, n = H.shape #H: R^(3*6)  = y_shape * x_shape
    X_hat = np.zeros((N+1,n)) #estimated state
    P = np.zeros((N+1,n,n))

    X = nominal_xs_us[0]
    Y = nominal_xs_us[1]

    X_hat[0] = x0 #initialize mean
    P[0] = sigma #initialize covariance matrix

    for i in range(N):
        #get linearized A and B at each estimated data point X_hat
        lin_dyn = LinearizeDynamics(bot.discrete_step,X_hat[i],control_sequence[i],dt)
        A = lin_dyn.get_A()
        B = lin_dyn.get_B()

        #estimate
        Xhat_pred = A@X_hat[i] + B@control_sequence[i]
        P_pred = A@P[i]@A.T + Q

        #update
        # S = H@P[i]@H.T + R
        S = H@P_pred@H.T + R
        S_inv = np.linalg.inv(S)
        K = P_pred@H.T@S_inv
        X_hat[i+1] = Xhat_pred + K@(Y[i+1]-H@Xhat_pred)
        # P[i+1] = P_pred - K@S@K.T
        P[i+1] = (np.identity(n)-K@H) @ P_pred

    return X_hat
```

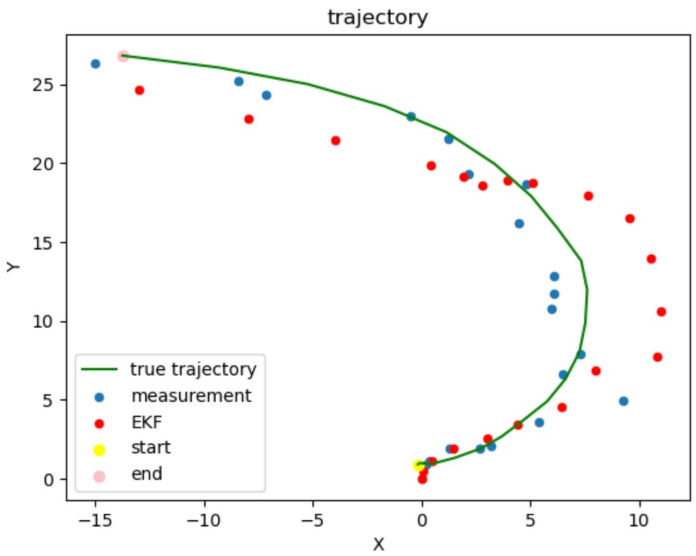
https://colab.research.google.com/drive/1DzZFVGXEzzdiJED9jnHNq_st-61X7rgE?usp=sharing

Discussion

Different sample noise size



Measurement Noise Covariance = $0.01*I_6$



Measurement Noise Covariance = $1.0*I_6$

Discussion

Highly nonlinearity ⇒ too large linearize errors

CAN'T deal with discountinuous dynamics

CAN only deal with unimodal distribution ⇒ Gaussian noise

Heavy computation for jacobian

Discussion

Highly nonlinearity ⇒ too large linearize errors

CAN'T deal with discountinuous dynamics

CAN only deal with unimodal distribution ⇒ Gaussian noise

Heavy computation for jacobian

How to improve estimation?

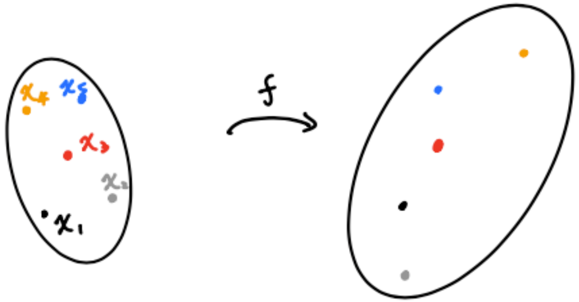
Unscented Kalman Filter (UKF)

Generate sigma points X_i

Calculate weight w_i

Transform sigma points through nonlinear function f

Recompute Gaussian



University of Maryland, ENAE 788M

Sigma Points

$$\sum_{\forall i} W_i = 1$$

$$\mu = \sum_{\forall i} W_i \mathcal{X}_i$$

cov Σ or $P = \sum_{\forall i} W_i (\mathcal{X}_i - \mu)(\mathcal{X}_i - \mu)^T$

n states, we generate $2n+1$ sigma pts.

$$\mathcal{X}_0 = \mu$$

$$\mathcal{X}_i = \begin{cases} \mu + \left(\sqrt{(n+\lambda)\Sigma}\right)_i, & i \in 1, 2, \dots, n \\ \mu - \left(\sqrt{(n+\lambda)\Sigma}\right)_i, & i \in n+1, n+2, \dots, 2n \end{cases}$$

Matrix sqrt

state dim.

scaling parameter

column vector

Weights

mean =

$$W_{0,m} = \frac{\lambda}{n+\lambda}$$

$$W_{i,m} = \frac{1}{2(n+\lambda)}, \quad i = 1, \dots, 2n$$

covariance =

$$W_{0,c} = W_{0,m} + (1 - \alpha^2 + \beta)$$

$$W_{i,c} = \frac{1}{2(n+\lambda)}, \quad i \in 1, \dots, 2n$$

$\kappa \geq 0$

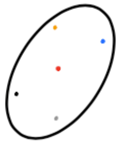
$\alpha \in (0, 1]$

$\lambda = \alpha^2 (n + \kappa) - n$

κ, α, λ all influence the spread of sigma pts from mean

$\beta = 2$ is optimal for Gaussian Noise

$\kappa, \alpha, \lambda \uparrow$



$\kappa, \alpha, \lambda \downarrow$

UNIVERSITY of WASHINGTON

Steps

Prediction Step:

$$\bar{\mu}_t = \sum_{i=0}^{2n} w_{i,m} f(\mathcal{X}_{t-1,i}, u_t)$$

$$\mathcal{X}_{t,0} = \mu_{t-1}$$

$$\mathcal{X}_{t,i} = \mu_{t-1} \pm \left(\sqrt{(n+\lambda)\Sigma_{t-1}}\right)_i$$

$$\bar{\Sigma}_t = \sum_{i=0}^{2n} w_{i,c}\left(f(\mathcal{X}_{t-1,i}, u_t) - \bar{\mu}_t\right)\left(f(\mathcal{X}_{t-1,i}, u_t) - \bar{\mu}_t\right)^T + Q_t$$

Update Step:

$$\mathcal{Z}_t = g(\mathcal{X}_t, 0)$$

$$\hat{z}_t = \sum_{i=0}^{2n} w_{i,m} \mathcal{Z}_{t,i}$$

$$S_t = \sum_{i=0}^{2n} w_{i,c}\left(\mathcal{Z}_{t,i} - \hat{z}_t\right)\left(\mathcal{Z}_{t,i} - \hat{z}_t\right)^T + R_t$$

$$\bar{\Sigma}_{\mathrm{t,xz}} = \sum_{i=0}^{2n} w_{i,c}\left(\mathcal{X}_{t,i} - \bar{\mu}_t\right)\left(\mathcal{Z}_{t,i} - \hat{z}_t\right)^T$$

$$K_t = \bar{\Sigma}_{\mathrm{t,xz}} S_t^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$$

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

University of Maryland, ENAE 788M

Comparison EKF vs UKF