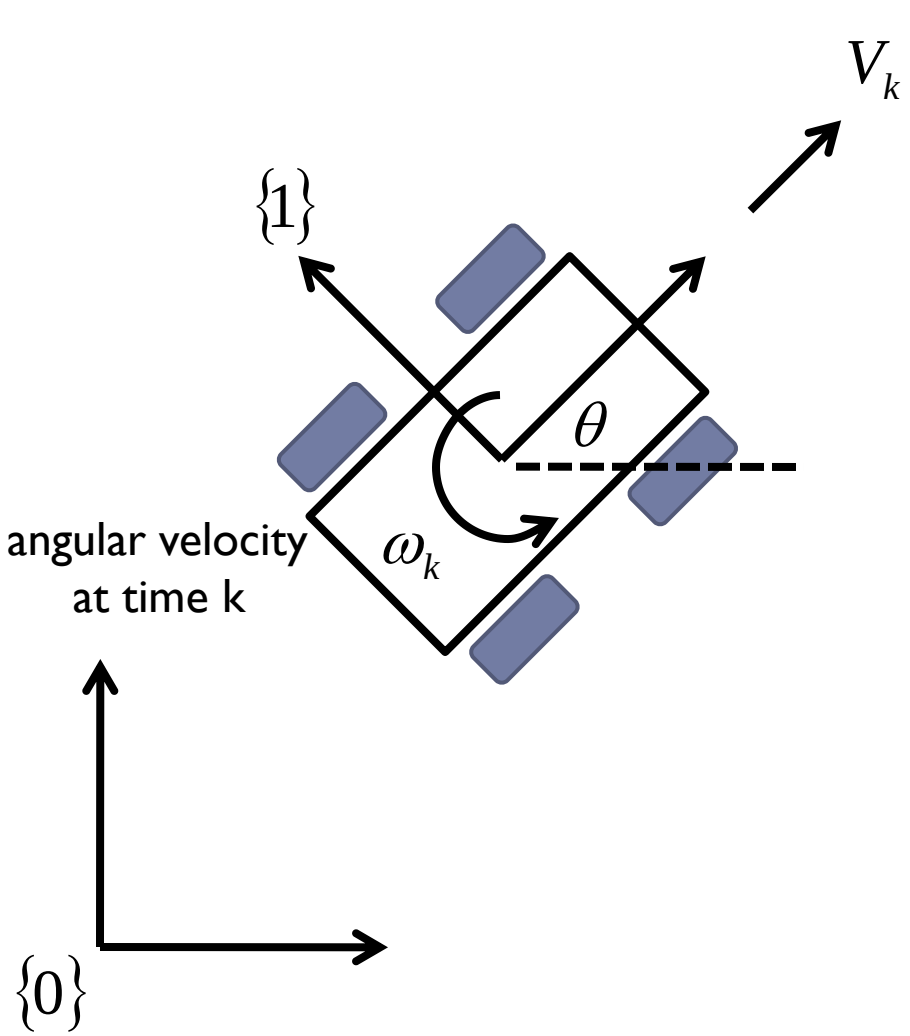
Day 22

Extended Kalman Filter

Simple Mobile Robot



"state"
pose in world
coordinates

$$x = \begin{bmatrix} X_k \\ Y_k \\ \theta_k \end{bmatrix}$$

"control input"
velocities in world
coordinates

$$u_k = \begin{bmatrix} V_k \cos\theta_k \\ V_k \sin\theta_k \\ \omega_k \end{bmatrix}$$

Plant Model

- assuming time steps of size 1 and forward speed of V_k

$$x_{k+1} = \begin{bmatrix} X_k + V_k \cos\theta_k \\ Y_k + V_k \cos\theta_k \\ \theta_k + \omega_k \end{bmatrix} + v_k$$
$$= f(x_k, u_k) + v_k$$

Measurement Model

- assume the robot can measure the distance to a beacon fixed in space at position $[x_b \;\; y_b]^{\mathrm{T}}$

$$
\begin{aligned}
z_k &= \sqrt{\left(X_k - x_b\right)^2 + \left(Y_k - y_b\right)^2} + w_k \\
&= h\left(x_k\right) + w_k
\end{aligned}
$$

Matlab Simulation

```
% beacon locations in world
Nbeacon = 4;
beacon = [1 9 9 1;
          1 1 9 9];

% TRUE STATE
% robot starts at [9 5] facing +y in world
% and travels in a circle of radius 4
Nstates = 360;
ang = [0:359] * pi/180;
xtrue = [4*cos(ang) + 5;
         4*sin(ang) + 5;
         ang + pi/2];

% TRUE MEASUREMENTS
delta = repmat(beacon(:), 1, length(ang)) - repmat(xtrue(1:2,:), 4, 1);
delta = delta .* delta;
ztrue = [sqrt(sum(delta(1:2,:)));
         sqrt(sum(delta(3:4,:)));
         sqrt(sum(delta(5:6,:)));
         sqrt(sum(delta(7:8,:)))];
```

```
% NOISY MEASUREMENTS
z = ztrue + 0.2*randn(Nbeacon, Nstates);

% plant noise covariance
Cv = [0.1*0.1 0       0;
      0       0.1*0.1 0;
      0       0       0.01*0.01];

% measurement noise covariance
Cw = 0.2*0.2*eye(4);
```

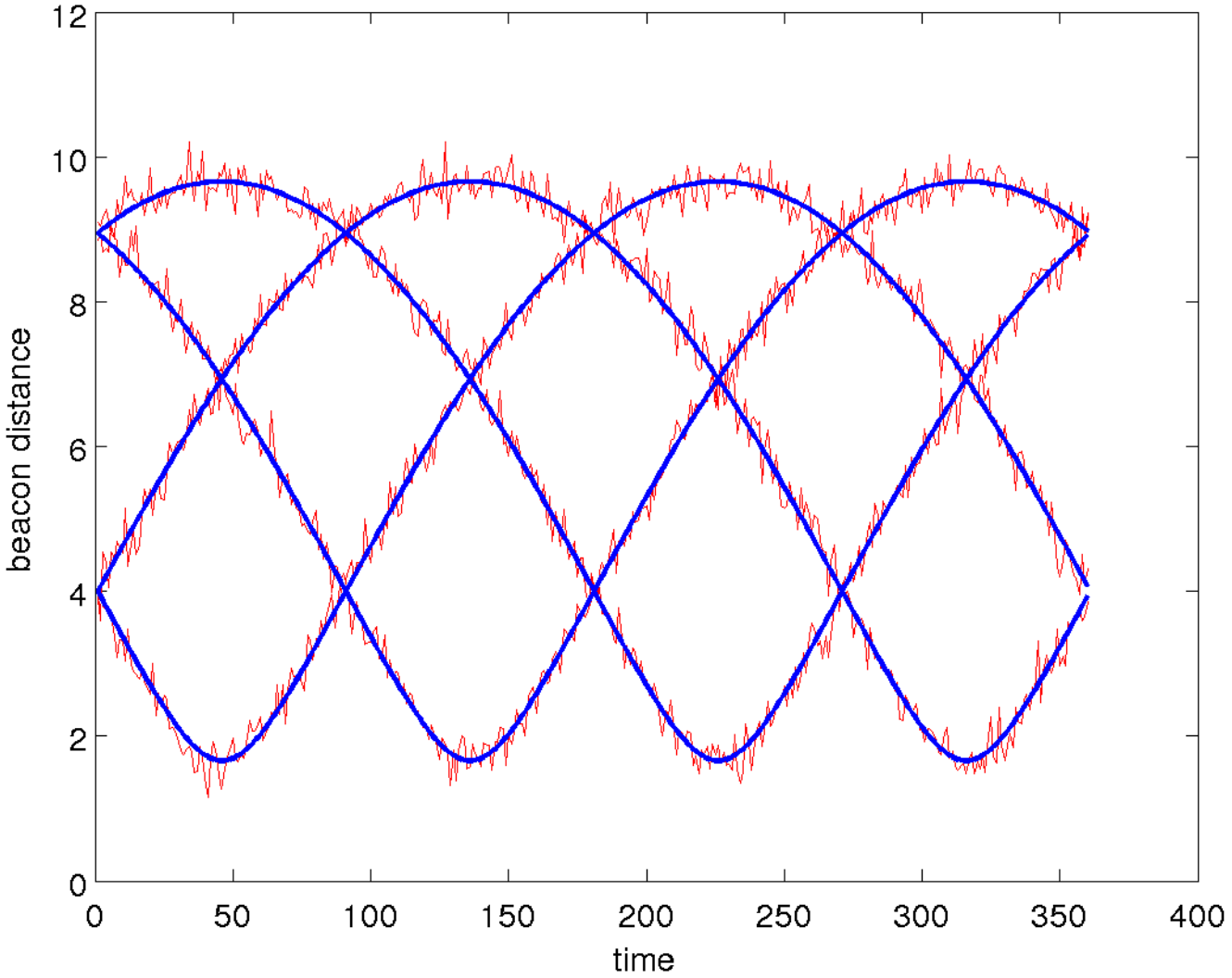
```
% EKF ESTIMATION
x = zeros(3, Nstates);
P = zeros(3, 3, Nstates);

% linear velocity (radius * angular velocity)
V = 4 * (2*pi/360);

% initial state and covariance
x(:,1) = [0 0 pi/2]';
P(:,:,1) = eye(3);
```

```
for b = 2:Nstates
    % the previous time index
    a = b - 1;

    % individual elements of state
    xk = x(1,a);
    yk = x(2,a);
    thetak = x(3,a);

    % predict state
    xpred = x(:,a) + [V*cos(thetak);
                      V*sin(thetak);
                      pi/180];

    % Jacobian of plant model
    Jplant = [1 0 -V*sin(thetak);
              0 1  V*cos(thetak);
              0 0 1];

    % predict state covariance
    Ppred = Jplant * P(:,:,a) * Jplant' + Cv;
```

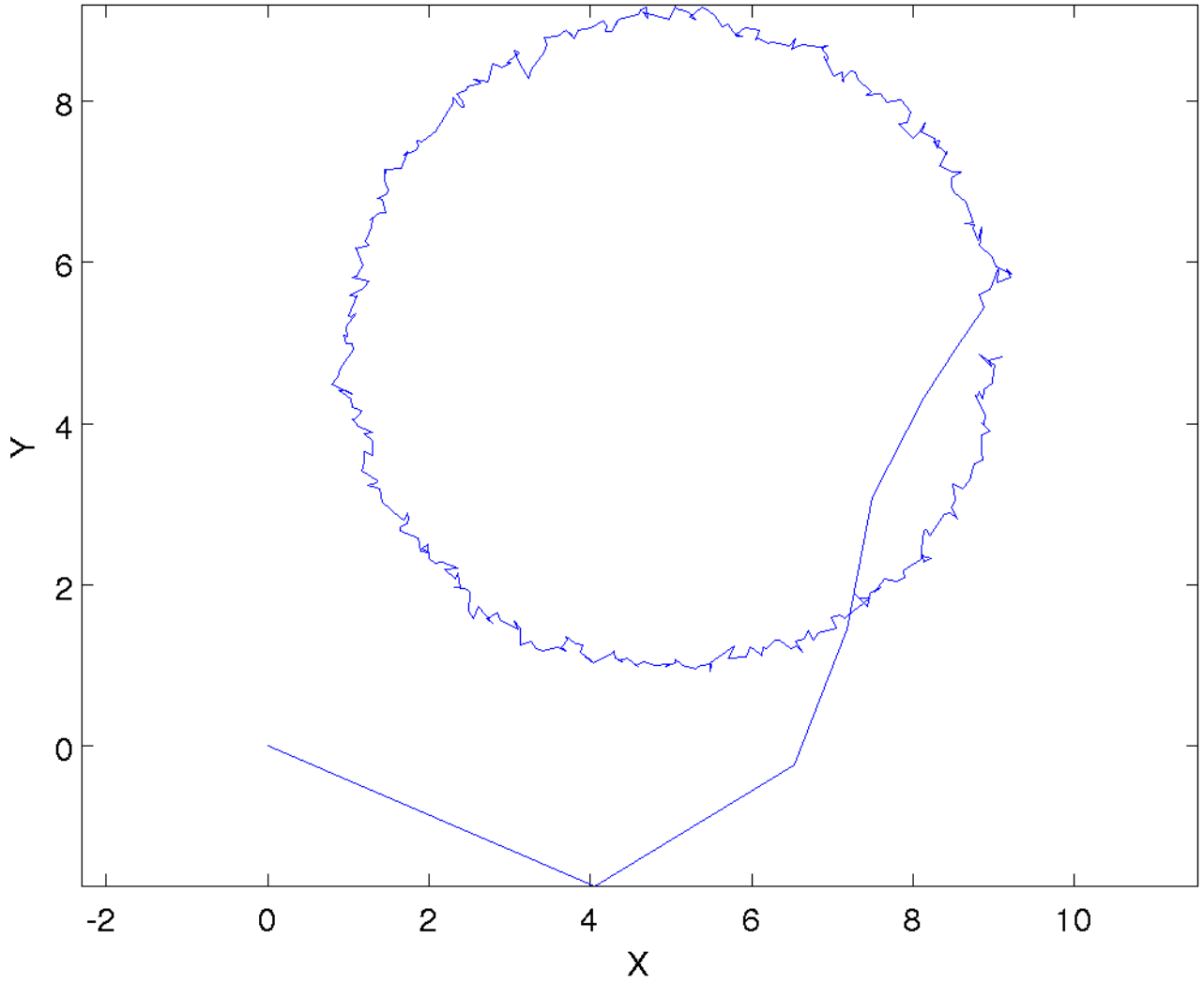
```
% Jacobian of measurement model
    c1 = 1 / sqrt((xk - beacon(1,1))^2 + (yk - beacon(2,1))^2);
    c2 = 1 / sqrt((xk - beacon(1,2))^2 + (yk - beacon(2,2))^2);
    c3 = 1 / sqrt((xk - beacon(1,3))^2 + (yk - beacon(2,3))^2);
    c4 = 1 / sqrt((xk - beacon(1,4))^2 + (yk - beacon(2,4))^2);

    Jmeas = [c1*(xk - beacon(1,1))  c1*(yk - beacon(2,1))  0;
             c2*(xk - beacon(1,2))  c2*(yk - beacon(2,2))  0;
             c3*(xk - beacon(1,3))  c3*(yk - beacon(2,3))  0;
             c4*(xk - beacon(1,4))  c4*(yk - beacon(2,4))  0];
```

```
% measurement prediction
    zpred = [sqrt((xpred(1) - beacon(1,1))^2 + (xpred(2) - beacon(2,1))^2);
             sqrt((xpred(1) - beacon(1,2))^2 + (xpred(2) - beacon(2,2))^2);
             sqrt((xpred(1) - beacon(1,3))^2 + (xpred(2) - beacon(2,3))^2);
             sqrt((xpred(1) - beacon(1,4))^2 + (xpred(2) - beacon(2,4))^2)];

    % innovation
    r = z(:,b) - zpred;

    % innovation covariance
    S = Jmeas * Ppred * Jmeas' + Cw;

    % Kalman gain
    K = Ppred * Jmeas' * inv(S);

    % new state estimate
    x(:,b) = xpred + K * r;

    % new covariance estimate
    P(:,:,b) = (eye(3) - K * Jmeas) * Ppred;
end
```

Real and Noisy Measurements



Estimated Position (poor initial state estimate)



Estimated Position (good initial state estimate)