# ASSIGNMENT 6

DECEMBER 21, 2021

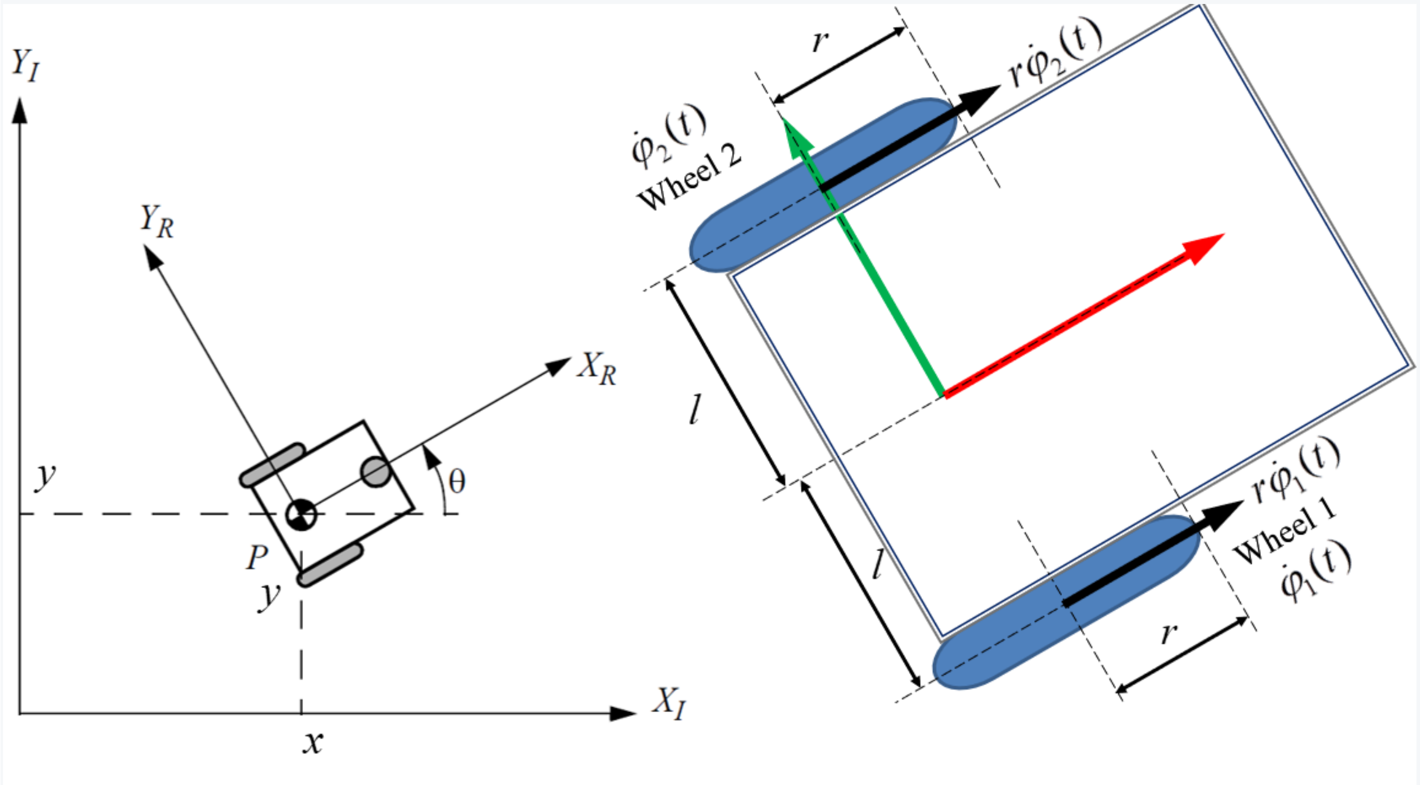
**Submitted to: Prof. Dr. Kemalettin Erbatur**

**Name of Student: Moses Chuka Ebere**
**Student Number: 31491**
**Course: Autonomous Mobile Robotics (ME 525)**

# TABLE OF CONTENTS

# INTRODUCTION

This is a computational assignment on Kalman Filtering. The system under consideration here is that of a differential drive robot.




The state-evolution and measurement equations are as follows:

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + \Delta t \cos\theta \left( \dfrac{ru_{1_k}}{2} + \dfrac{ru_{2_k}}{2} \right) \\ y_{k-1} + \Delta t \sin\theta \left( \dfrac{ru_{1_k}}{2} + \dfrac{ru_{2_k}}{2} \right) \\ \theta_{k-1} + \Delta t \left( \dfrac{ru_{1_k}}{2l} - \dfrac{ru_{2_k}}{2l} \right) \end{bmatrix} + \begin{bmatrix} w_{1_{k-1}} \\ w_{2_{k-1}} \\ w_{3_{k-1}} \end{bmatrix}$$

$$\begin{bmatrix} z_{1_k} \\ z_{2_k} \\ z_{3_k} \end{bmatrix} = \begin{bmatrix} \sqrt{x_k^2 + y_k^2} \\ \operatorname{atan2}(y, x) \\ \theta_k \end{bmatrix} + \begin{bmatrix} v_{1_k} \\ v_{2_k} \\ v_{3_k} \end{bmatrix}$$

$$\boxed{\text{Measured:} \quad \begin{bmatrix} d_k \\ \alpha_k \\ \theta_k \end{bmatrix}}$$

From the above, we observe that the equations are non-linear; hence, we'll need to linearize them and perform Extended Kalman Filtering simulation.

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_{k-1}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k$$

- Let $\mathbf{A}$ be the Jacobian of $f$ with respect to $\mathbf{x}$.

$$\mathbf{A}_{ij} = \frac{\partial f_i}{\partial x_j}(\mathbf{x}_{k-1}, \mathbf{u}_k)$$

- Let $\mathbf{H}$ be the Jacobian of $h$ with respect to $\mathbf{x}$.

$$\mathbf{H}_{ij} = \frac{\partial h_i}{\partial x_j}(\mathbf{x}_k)$$

$$A = \frac{\partial f}{\partial \vec{x}} = \begin{bmatrix} 1 & 0 & -\Delta t\left(\frac{ru_{1_k}}{2} + \frac{ru_{2_k}}{2}\right)\sin\theta \\ 0 & 1 & \Delta t\left(\frac{ru_{1_k}}{2} + \frac{ru_{2_k}}{2}\right)\cos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$H = \frac{\partial h}{\partial \vec{x}} = \begin{bmatrix} \frac{x_k}{\sqrt{x_k^2 + y_k^2}} & \frac{y_k}{\sqrt{x_k^2 + y_k^2}} & 0 \\ -\frac{y}{x_k^2 + y_k^2} & \frac{x}{x_k^2 + y_k^2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# MATLAB CODE

```
m-file name - Assignment_6

clear all
close all

% Sampling parameters
delta_t = 0.001;

% Plant parameters
r = 0.1;
l = 0.6;

% Noise parameters
Q = [0.0001 0  0; 0 0.0001 0; 0 0 0.0001];
R = [0.015 0 0; 0 0.01 0; 0 0 0.01];

% Control parameters
u1_amplitude = 0.1;
u1_frequency = 0.1;
u1_offset = 0;
u2_amplitude = 0.12;
u2_frequency = 0.02;
u2_offset = 0;

% Initial values
P_k_minus_1 = [2 0 0; 0 2 0; 0 0 2];
x_initial = [1; 3; pi/6]; % Contains x, y, and theta
x = x_initial;
x_i_1 = x_initial(1);
x_i_2 = x_initial(2);
x_i_3 = x_initial(3);
delf_delx_initial = [1 0 0;
                     0 1 0;
                     0 0 1];
A = delf_delx_initial;
delh_delx_initial = [x_i_1/sqrt(x_i_1^2 + x_i_2^2)
x_i_2/sqrt(x_i_1^2 + x_i_2^2) 0;
                     -x_i_2/(x_i_1^2 + x_i_2^2)
x_i_1/(x_i_1^2 + x_i_2^2)     0;
                     0                             0
1];
H = delh_delx_initial;
```

```
x_hat_k_minus_1 = x_initial; % A 3x1 vector containing x,
y, and theta.
K_k = P_k_minus_1*H'*inv(H*P_k_minus_1*H' + R);

% Record Arrays
y_list = [];
y_hat_list = [];
u_list = [];
K_k_list = [];
P_k_list = [];

for k = 1:1000
    % Control input (not a feedback control input)
    u_k = [u1_offset + u1_amplitude * sin(2 * pi *
u1_frequency * k * delta_t);
           u2_offset + u2_amplitude * sin(2 * pi *
u2_frequency * k * delta_t)];

    % Prediction Update
    x_hat_minus_k = x_hat_k_minus_1 + [delta_t * 0.5 * r *
(u_k(1) + u_k(2)) * cos(x_hat_k_minus_1(3));
                                       delta_t * 0.5 * r *
(u_k(1) + u_k(2)) * sin(x_hat_k_minus_1(3));
                                       delta_t * 0.5 *
(r/l) * (u_k(1) - u_k(2))];
    delf_delx = [1 0 -delta_t*0.5*r*(u_k(1) +
u_k(2))*sin(x_hat_k_minus_1(3));
                 0 1  delta_t*0.5*r*(u_k(1) +
u_k(2))*cos(x_hat_k_minus_1(3));
                 0 0  1];

    A = delf_delx;
    P_minus_k = A * P_k_minus_1 * A' + Q;

    % Measurement Update
        % Plant Simulation Equation
        x = x + [delta_t * 0.5 * r * (u_k(1) + u_k(2)) *
cos(x(3));
                 delta_t * 0.5 * r * (u_k(1) + u_k(2)) *
sin(x(3));
```

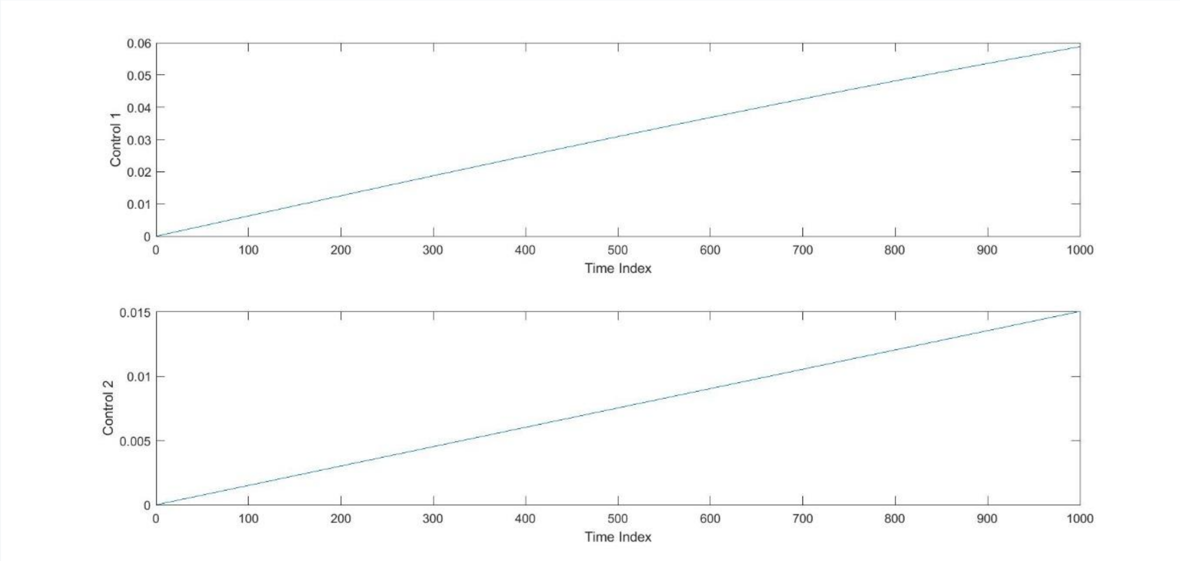
```
                 delta_t * 0.5 * (r/l) * (u_k(1) - u_k(2))]
...
              + [sqrt(Q(1,1))*randn; sqrt(Q(2,2))*randn;
sqrt(Q(3,3))*randn];
          % Measurement Simulation Equation
          d = sqrt(x(1)^2 + x(2)^2);
          alpha = atan2(x(2),x(1));
          theta = x(3);
          d_noisy = d + sqrt(R(1,1)) * randn;
          alpha_noisy = alpha + sqrt(R(2,2)) * randn;
          theta_noisy = theta + sqrt(R(3,3)) * randn;
          z_k = [d_noisy;
                 alpha_noisy;
                 theta_noisy];

          % Computation of K
          delh_delx =
[x_hat_k_minus_1(1)/sqrt(x_hat_k_minus_1(1)^2+x_hat_k_minus
_1(2)^2) ...

x_hat_k_minus_1(2)/sqrt(x_hat_k_minus_1(1)^2+x_hat_k_minus_
1(2)^2) ...
                       0;
                      -
x_hat_k_minus_1(2)/(x_hat_k_minus_1(1)^2+x_hat_k_minus_1(2)
^2) ...

x_hat_k_minus_1(1)/(x_hat_k_minus_1(1)^2+x_hat_k_minus_1(2)
^2) ...
                       0;
                       0 0 1];
          H = delh_delx;
          K_k = P_minus_k*H'*inv(H*P_minus_k*H' + R);
          % State Estimate Correction
          x_hat_k = x_hat_minus_k ...
                  + K_k * (z_k - [sqrt(x_hat_minus_k(1)^2 +
x_hat_minus_k(2)^2);
                                 atan2(x_hat_minus_k(2),
x_hat_minus_k(1));
                                 x_hat_minus_k(3)] ...
                           );
```

```
            % Error Covariance Update
            P_k = (eye(3) - K_k*H) * P_minus_k;

        % Variables for the next cycle
        P_k_minus_1 = P_k;
        x_hat_k_minus_1 = x_hat_k;

        % Array Recording
        y_list = [y_list; x'];
        y_hat_list = [y_hat_list; x_hat_k'];
        u_list = [u_list; u_k'];
        K_k_list = [K_k_list; K_k(1,1)];
        % P_k_list = [P_k_list; P_k];
end

figure %1
subplot(2,1,1)
plot(u_list(:,1))
xlabel('Time Index')
ylabel('Control 1')
subplot(2,1,2)
plot(u_list(:,2))
xlabel('Time Index')
ylabel('Control 2')

figure %2
subplot(2,1,1)
plot(y_list(:,1), 'r')
xlabel('Time Index')
ylabel('Position 1')
subplot(2,1,2)
plot(y_hat_list(:,1), 'k')
xlabel('Time Index')
ylabel('Estimate 1')

figure %3
subplot(2,1,1)
plot(y_list(:,2), 'r')
xlabel('Time Index')
ylabel('Position 2')
subplot(2,1,2)
plot(y_hat_list(:,2), 'k')
```

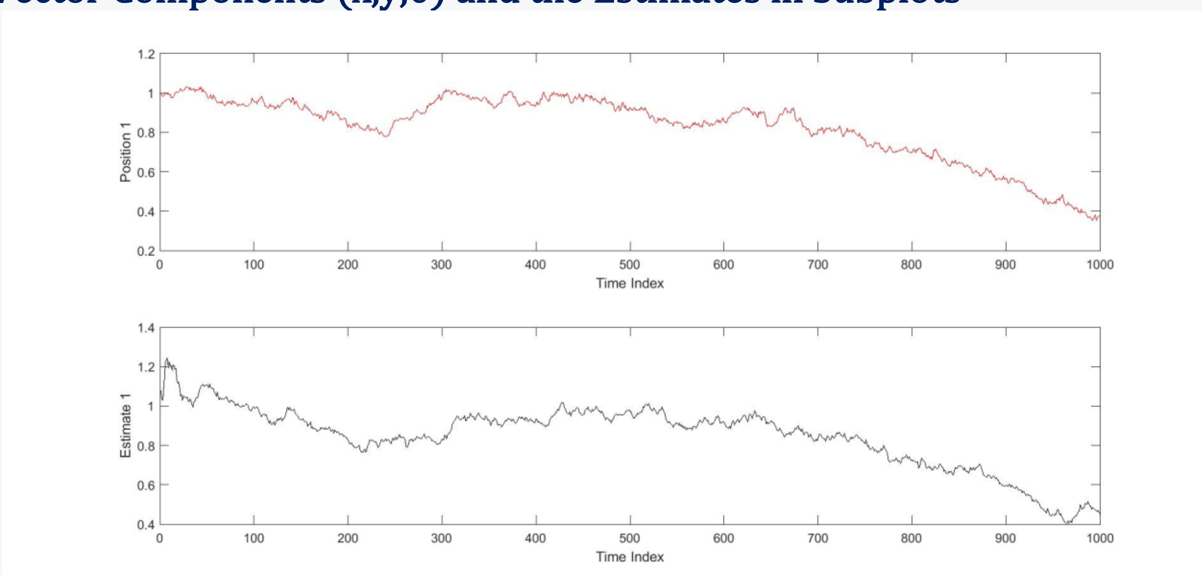
```
xlabel('Time Index')
ylabel('Estimate 2')

figure %4
subplot(2,1,1)
plot(y_list(:,3), 'r')
xlabel('Time Index')
ylabel('Position 3')
subplot(2,1,2)
plot(y_hat_list(:,3), 'k')
xlabel('Time Index')
ylabel('Estimate 3')

figure %5
plot(y_list(:,1), 'r')
hold
plot(y_hat_list(:,1), 'k')
xlabel('Time Index')
ylabel('Position 1')

figure %6
plot(y_list(:,2), 'r')
hold
plot(y_hat_list(:,2), 'k')
xlabel('Time Index')
ylabel('Position 2')

figure %7
plot(y_list(:,3), 'r')
hold
plot(y_hat_list(:,3), 'k')
xlabel('Time Index')
ylabel('Position 3')

figure %8
plot(K_k_list(:,1), 'b')
xlabel('Time Index')
ylabel('Gain')
```

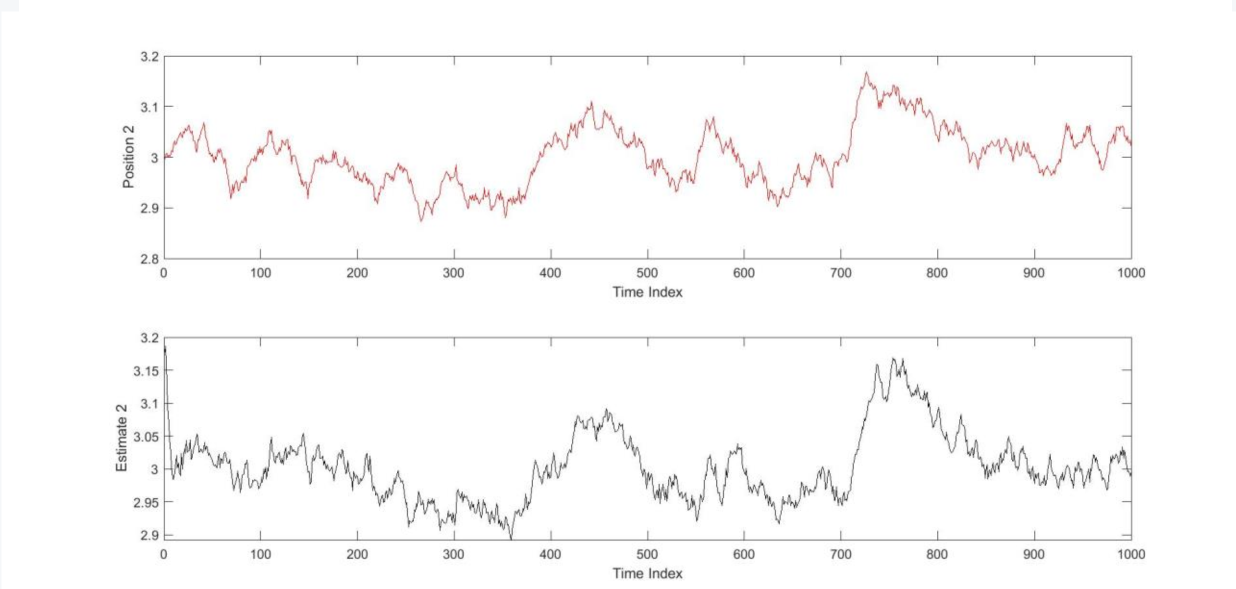
# PLOTS

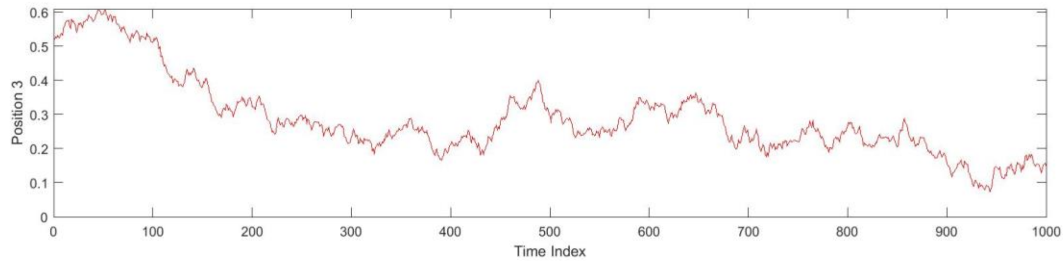
## Control Inputs

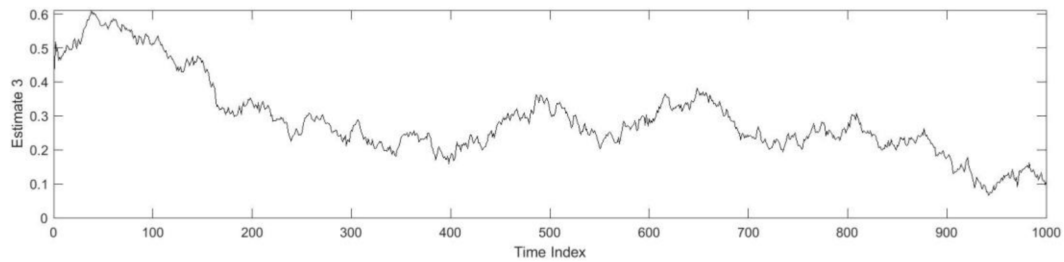



## State Vector Components (x,y,θ) and the Estimates in Subplots

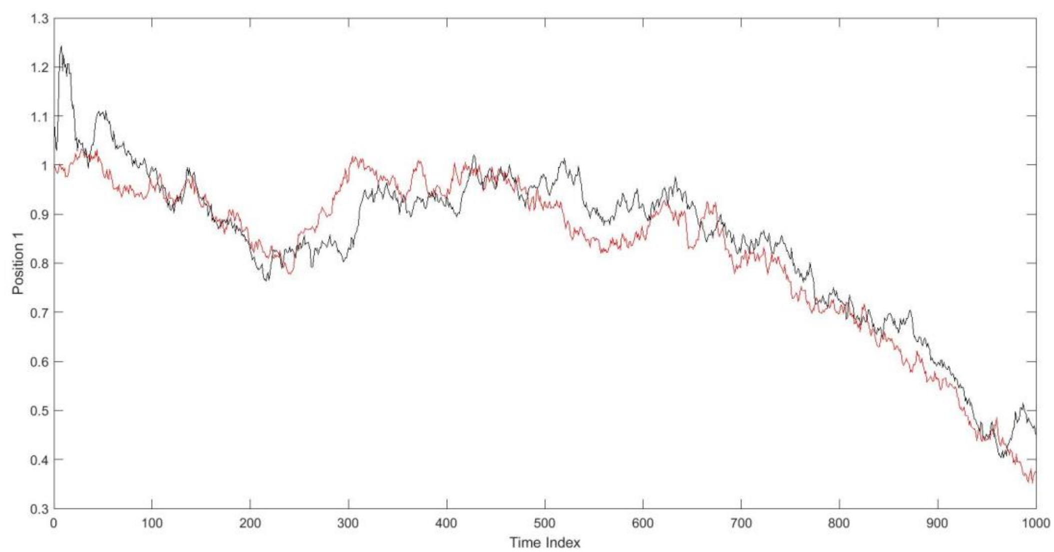


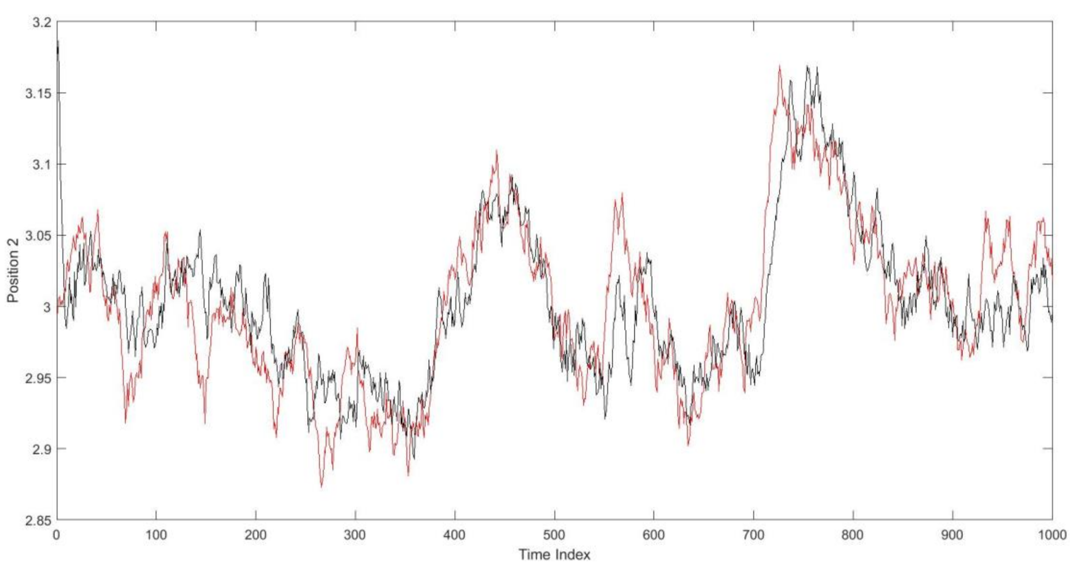



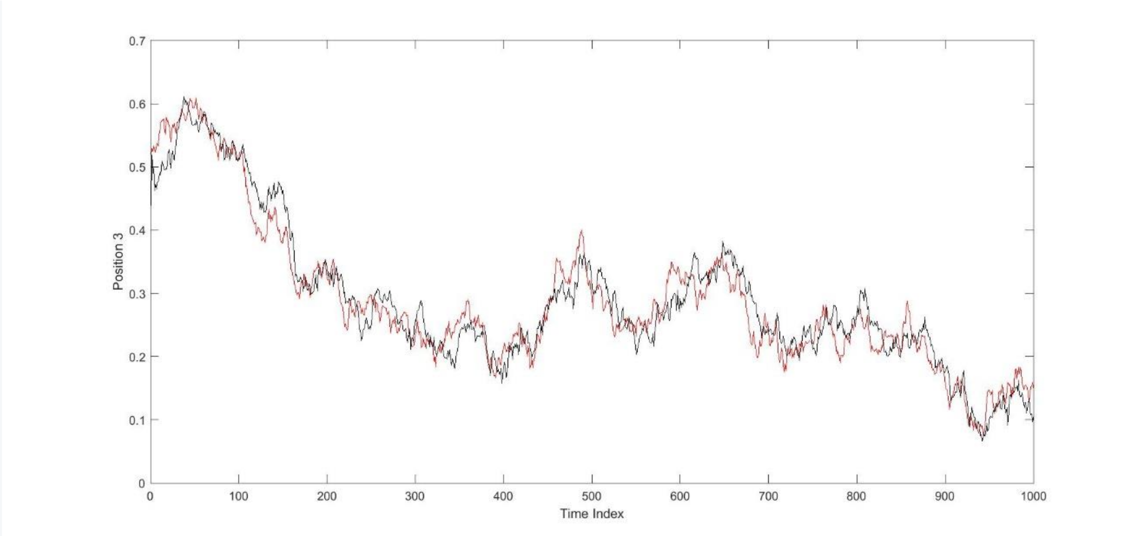


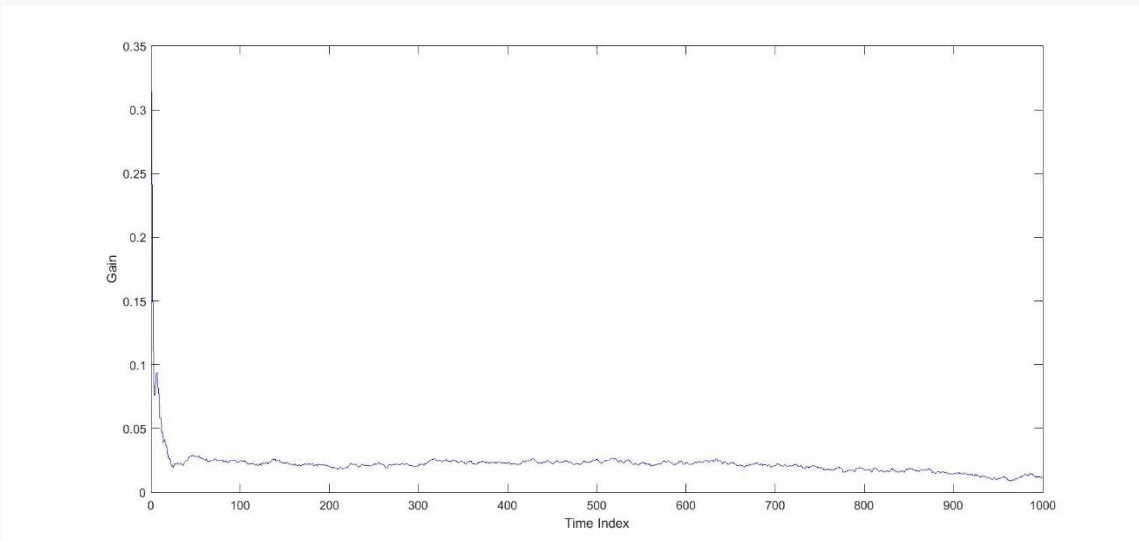

## State Vector Components (x,y,θ) and the Estimates Together






### Kalman Gain




## DISCUSSION

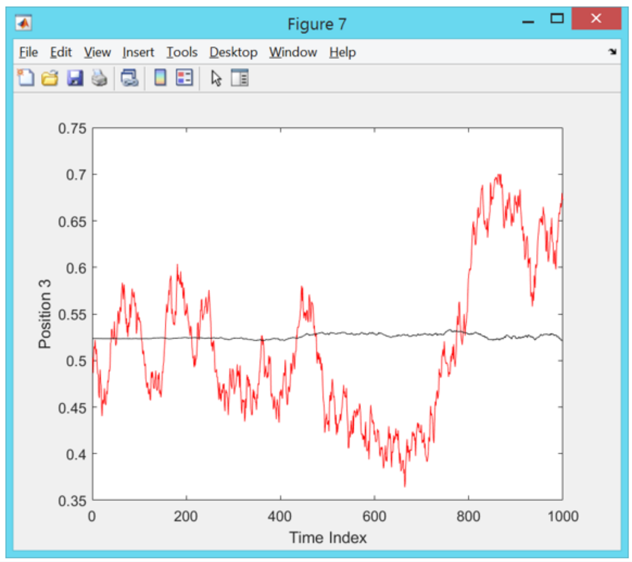
- The results are as expected. The estimates (in black) are very close to the values of the state variables from the plant simulation at every instant.
- Due to the random noise (gaussian) implemented, the results will vary whenever the code is rerun.
- The Kalman gain starts out with random values because we start the simulation with really noisy parameters; however, with more iterations of prediction and correction, the Kalman gain flattens out.

- Comparing these results to those obtained when theta was not part of the measurement equation:

$$\begin{bmatrix} z_{1_k} \\ z_{2_k} \end{bmatrix} = \underbrace{\boxed{\begin{bmatrix} \sqrt{x_k^2 + y_k^2} \\ a\tan 2(y,x) \end{bmatrix}}}_{h} + \begin{bmatrix} v_{1_k} \\ v_{2_k} \end{bmatrix}$$

Measured:

$$\begin{bmatrix} d_k \\ \alpha_k \end{bmatrix}$$




We notice that when theta is not measured, the estimated state variable theta remains almost constant; however, when we explicitly attempt to measure theta as we have done in this assignment, its estimate is close to that from the measurement simulation (below):