UNIVERSIDAD DE GUADALAJARA
CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Sistemas Inteligentes III

Red neuronal de kalman extendido

Nombre:	Sebastian Martínez Muñiz 217567225

23 de mayo de 2023, Guadalajara

Contenido

Introducción

Se plantea un robot diferencial con dos ruedas actuadas y una rueda loca al cual se buscara simular con el método de Euler además de diseñar un estimador por medio de una red neuronal con kalman extendido.

El robot a simular se muestra en la siguiente imagen.

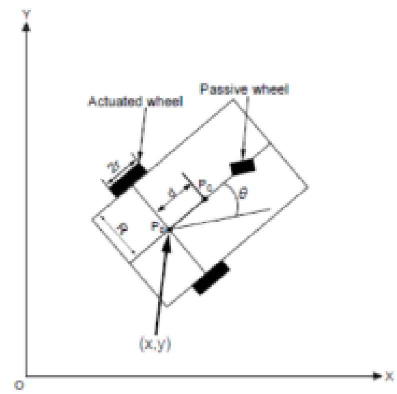


Figura 4.1: Robot Móvil con dos ruedas actuadas.

Modelo del robot

Para el robot que se muestra en a figura 4.1 se puede obtener el siguiente modelo en espacio de estados.

$$\dot{x_1} = J(x_1)x_2 \tag{1}$$

$$\dot{x_2} = M^{-1} - C(\dot{x_1})x_2 - Dx_2/\tau_d + NK_T x_3 \tag{2}$$

$$\dot{x_3} = L_a^{-1}(u - R_a x_3 - NK_E x_2) \tag{3}$$

Donde tenemos las siguientes variables.

$$x_1 = \begin{bmatrix} x_{11} & x_{12} & x_{13} \end{bmatrix}^T$$

$$x_2 = \begin{bmatrix} x_{21} & x_{22} \end{bmatrix}^T$$

$$x_3 = \begin{bmatrix} x_{31} & x_{32} \end{bmatrix}^T$$

$$J(x_1) = \begin{bmatrix} cos(x_{13}) & cos(x_{13}) \\ sin(x_{13}) & sin(x_{13}) \\ R^{-1} & R^{-1} \end{bmatrix}$$

$$M = \begin{bmatrix} m_{11} & m_{12} \\ m_{12} & m_{11} \end{bmatrix}$$

$$C(k) = 0.5R^{-1}r^2m_cd \begin{bmatrix} 0 & \dot{x_{13}} \\ -\dot{x_{13}} & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} d_{11} & 0 \\ 0 & d_{11} \end{bmatrix}$$

De los cuales se usaran los siguientes parámetros.

$$m_{11} = 0.25R^{-2}r^2(mR^2 + I) + I_w$$

$$m_{12} = 0.25R^{-2}r^2(mR^2 - I)$$

$$m = m_c + 2m_w$$

$$I = m_cd^2 + 2m_wR^2 + i_c + 2I_m$$

$$\tau = \begin{bmatrix} \tau_1 & \tau_2 \end{bmatrix}^T$$

$$\tau_d = \begin{bmatrix} \tau_{d1} & \tau_{d2} \end{bmatrix}^T$$

En base a la siguiente tabla de parámetros físicos.

$R = 0.75m$	$I_m = 0.0025kgm^2$
$d = 0.3m$	$R_a = diag[2.5, 2.5]\Omega$
$r = 0.15m$	$L_a = diag[0.048, 0.048]H$
$m_c = 30kg$	$K_E = diag[0.02, 0.02]V/(rad/s)$
$m_w = 1kg$	$N = diag[62.55, 62.55]$
$I_c = 15.625kgm^2$	$K_T = diag[0.2613, 0.2613]Nm/A$
$I_w = 0.005kgm^2$	$d_{m1} = d_{m2} = 0.5N$

Para poder ver la evolución de nuestras variables de estado es necesario integrar las ecuación, en nuestro caso para la simulación se realizara por el método de Euler, donde se va integrando por pasos.

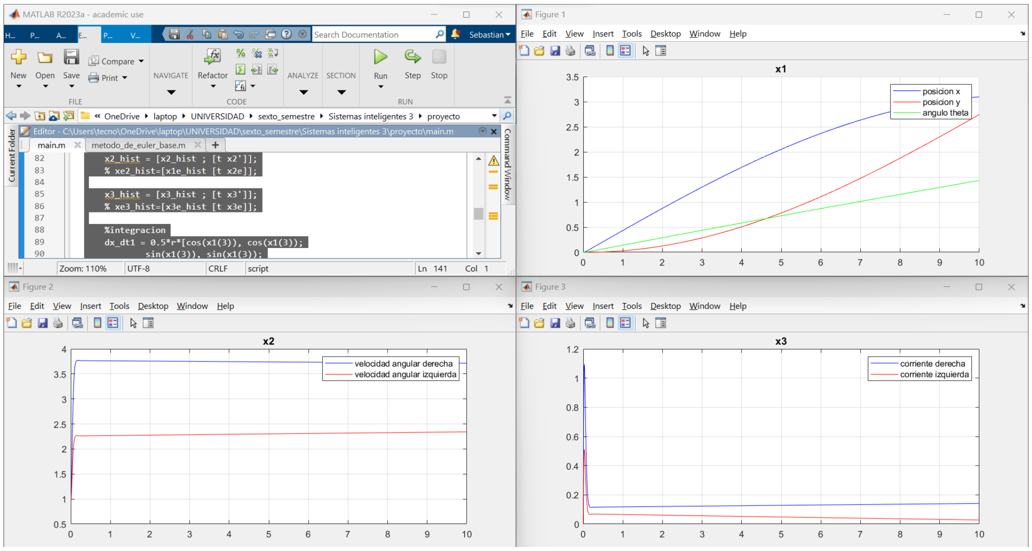
En cada paso de calcula dx_dt donde aquí se pone la ecuación de estado, y para el calculo de la integral se suma el estado anterior mas dx_dt multiplicado por nuestro paso de tiempo. Quedando el siguiente código.

```
clear all
close all
clc

%% Declaracion de los parametros fisicos

R = 0.75;
d=0.3;
r=0.15;
m_c=30;
m_w = 1;
I_c = 15.625;
I_w=0.005;
I_m = 0.0025;
R_a = eye(2,2) * 2.5;
L_a = eye(2,2) * 0.048;
K_e = eye(2,2) * 0.02;
N= eye(2,2) * 62.55;
K_t = eye(2,2) * 0.2613;
d_m1 = 0.5;
d_m2 = d_m1;

%% declaracion de las matrices y vectores necesarios



% estado inicial x1
x1 = [0;1;2]; % [x;y;theta]
x2 = [0.5*pi;0.5*pi]; %velocidad angular de las ruedas


x3 = [0,0];  % corrientes


% declaramos los limites de tiempo
t0 = 0;
tf = 10;
%declaracion de las condiciones iniciales
x10 = [0 0 0]';
x20 = [1 1]';
```

```
x30 = [0 0]';

% declaracion de la estimacion inicial
x1e =[1 2 1]';
x2e =[2 1]';
x3e =[0.5 0]';

% incializamos los vectores
% valores reales
x1_hist = [];
x2_hist = [];
x3_hist = [];
% valores medidos
xe1_hist=[];
xe2_hist=[];
xe3_hist=[];


dt = 0.01; %definimos el periodo de tiempo

x1 = x10;
x2 = x20;
x3 = x30;

m=m_c + 2*m_w;
I = m_c*d^2+2*m_w*R^2+I_c+2*I_m;
m_1 = 0.25*R^-2*r^2*(m*R^2+I)+I_w;
m_2 = 0.25*R^-2*r^2*(m*R^2-I);
M = [m_1 m_2;m_2 m_1];
D = eye(2,2) * d_m1;
T_d = [0.001 0.001]';


%comenzamos con la integracion de las variables
figure(1)
hold on
for t = t0:dt:tf

    x1_hist = [x1_hist ; [t x1']];
    % xe1_hist=[x1e_hist [t x1e]];

    x2_hist = [x2_hist ; [t x2']];
    % xe2_hist=[x1e_hist [t x2e]];

    x3_hist = [x3_hist ; [t x3']];
    % xe3_hist=[x3e_hist [t x3e]];

    %integracion
    dx_dt1 = 0.5*r*[cos(x1(3)), cos(x1(3));
```

```
             sin(x1(3)), sin(x1(3));
             R^-1, -R^-1] * x2;

    x1 = x1 + dx_dt1*dt;

    C = 0.5*R^-1*r^2*m_c*d*[0, dx_dt1(3);
                             -dx_dt1(3), 0];

    dx_dt2 = M^-1*(-C*x2-D*x2-T_d+N*K_t*x3);
    x2 = x2 + dx_dt2*dt;

    u= [5 3]';

    dx_dt3 = L_a^(-1)*(u-R_a*x3-N*K_e*x2);
    x3 = x3 + dx_dt3*dt;

end

plot(x1_hist(:,1),x1_hist(:,2),'-b')
hold on
plot(x1_hist(:,1),x1_hist(:,3),'-r')
hold on
plot(x1_hist(:,1),x1_hist(:,4),'-g')
hold on
grid on
legend('posicion x','posicion y','angulo theta');
title('x1');

figure (2)
plot(x2_hist(:,1),x2_hist(:,2),'-b')
hold on
plot(x2_hist(:,1),x2_hist(:,3),'-r')
hold on
grid on
legend('velocidad angular derecha','velocidad angular ...
    izquierda');
title('x2');


figure (3)
plot(x3_hist(:,1),x3_hist(:,2),'-b')
hold on
plot(x3_hist(:,1),x3_hist(:,3),'-r')
hold on
grid on
legend('corriente derecha','corriente izquierda');
title('x3');
```

Del cual podemos observar las siguientes gráficas.



Donde s puede ver que efectivamente nuestra simulación hace lo deseado.

Filtro de Kalman

El filtro de Kalman es muy utilizado en la actualidad, principalmente como observador o estimador, esto quiere decir, que por medio del filtro se puede llegar a una estimación de estado aunque no tengamos los estados censados, claro esta de una propiedad de los sistemas que se llama observabilidad la cual depende de la matriz de estados A y la matriz de entrada B, claro esta, que al tratarse de un sistema meramente no lineal se deben de usar las ecuaciones que representan al sistema, aunque por otra parte de puede modificar el algoritmo de tal manera que solo es necesario agregar las matrices.

Los observadores o estimadores constan de dos partes fundamentales, la perdición y la corrección de los cuales el filtro de kalman no esta exento.
Dicho de otro modo debemos conocer la perdición de nuestro estado, una vez hecha la predicción tenemos que calcular una ganancia y finalmente agregar la corrección en base al error, siguiendo así las siguientes ecuación.

Considerando el siguiente sistema no lineal.

$$\dot{x_k} = f(x_{k-1}, u_k) + w_k$$

$$z_k = h(x_k) + v_k$$

Fase de predicción

Estimación

Estimación *a priori* $\hat{\mathbf{x}}_{k|k-1} = \mathbf{\Phi}_k \, \mathbf{x}_{k-1|k-1}$

Covarianza del error asociada a la estimación *a priori* $\mathbf{P}_{k|k-1} = \mathbf{\Phi}_k \mathbf{P}_{k-1|k-1} \mathbf{\Phi}_k^{\mathrm{T}} + \mathbf{Q}_k$

Fase de corrección

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^{\mathrm{T}} (\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$$

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1}$$

Donde ϕ_k es la matriz de transición de estados.

z_k es el vector de medición al momento k.

R_k es la matriz de varianza del ruido de las mediciones.

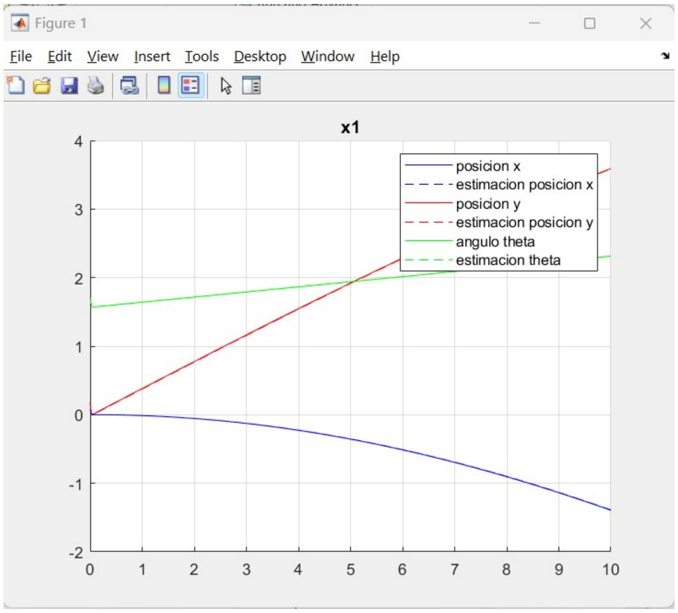
Implementación

De lo anterior tenemos el siguiente apartado de código.

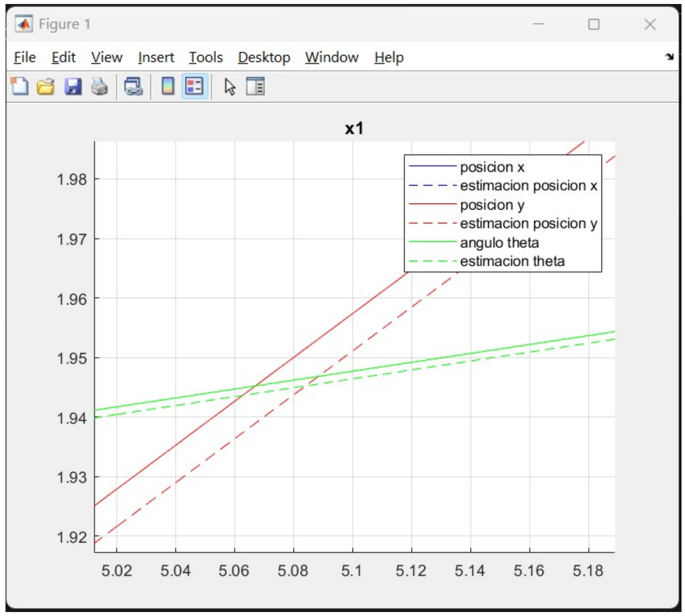
```
% Inicializamos variables
A = eye(3); % Matriz de transici n de estado
C_k = eye(3,3); % Matriz de observaci n

Q = eye(3) * 0.01; % Covarianza del ruido de proceso
R_k = eye(3) * 0.1; % Covarianza del ruido de medici n


P1e = eye(3); % Estimaci n inicial de la covarianza

xe1_hist = [];
P1e_hist = [];
x1e_pred = A * x1e;
for t = t0:dt:tf
    P1e_pred = A * P1e * A' + Q;

    y_1 = C_k * x1; % Medici n real
    K = P1e_pred * C_k' / (C_k * P1e_pred * C_k' + R_k); % Ganancia de Kalman
    x1e = x1e_pred + K * (y_1 - C_k * x1e_pred);
    P1e = (eye(3) - K * C_k) * P1e_pred;

```

```
    xe1_hist=[xe1_hist; [t x1e']];
    P1e_hist = [P1e_hist, diag(P1e)'];
end

plot(x1_hist(:,1),x1_hist(:,2),'-b')
hold on
plot(xe1_hist(:,1),xe1_hist(:,2),'--b')
hold on

plot(x1_hist(:,1),x1_hist(:,3),'-r')
hold on
plot(xe1_hist(:,1),xe1_hist(:,3),'--r')
hold on

plot(x1_hist(:,1),x1_hist(:,4),'-g')
hold on
plot(xe1_hist(:,1),xe1_hist(:,4),'--g')
hold on
grid on
legend('posicion x','estimacion posicion x','posicion ...
   y','estimacion posicion y','angulo theta','estimacion theta');
title('x1');
```

El cual nos entrega la siguiente gráfica.



La cual a priori no nos muestra el estimador, y esto sucede principalmente porque el error que existe entre ellos es muy pequeño por lo que se puede efectuar un zoom.



Como se puede apreciar, ahora somos capaces de ver como nuestro estado estimado (linea punteada) se hacer al estado real (linea solida).

Solo nos queda alimentar nuestro filtro de kalman con una red neuronal.

Al implementar con una red neuronal obtenemos el siguiente codigo el cual nos otorga el siguiente reusltado.

```
clear all
close all
clc

%% Declaracion de los parametros fisicos

R = 0.75;
d=0.3;
r=0.15;
m_c=30;
m_w = 1;
I_c = 15.625;
I_w=0.005;
I_m = 0.0025;
R_a = eye(2,2) * 2.5;
L_a = eye(2,2) * 0.048;
K_e = eye(2,2) * 0.02;
N= eye(2,2) * 62.55;
K_t = eye(2,2) * 0.2613;
d_m1 = 0.5;
d_m2 = d_m1;
```

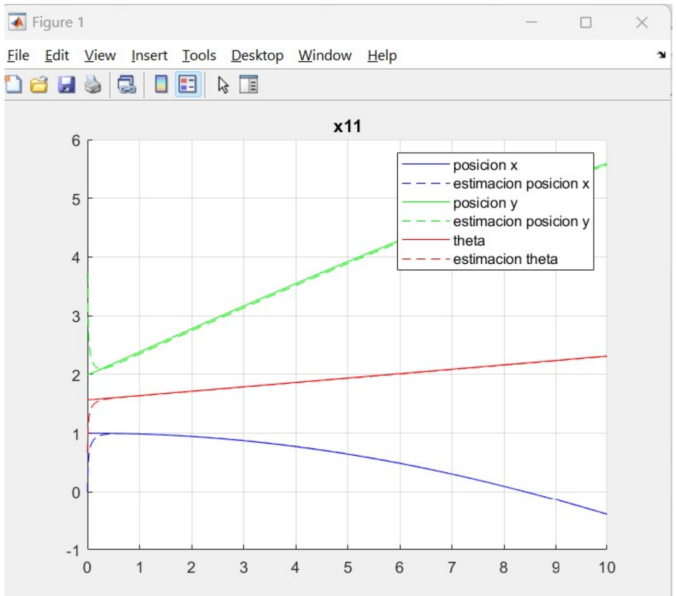
```

%% declaracion de las matrices y vectores necesarios



% estado inicial x1
x1 = [0;1;2]; % [x;y;theta]
x2 = [0.5*pi;0.5*pi]; %velocidad angular de las ruedas
x3 = [0,0];  % corrientes


% declaramos los limites de tiempo
t0 = 0;
tf = 10;
%declaracion de las condiciones iniciales
x10 = [1 2 pi/2]';
x20 = [1 1]';
x30 = [0 0]';

% declaracion de la estimacion inicial
x1e =[1 2]';
x2e =[2 1]';
x3e =[0.5 0]';

% incializamos los vectores
% valores reales
x1_hist = [];
x2_hist = [];
x3_hist = [];

% valores estimados
xe1_hist=[];
xe2_hist=[];
xe3_hist=[];


dt = 0.01; %definimos el periodo de tiempo

x1 = x10;
x2 = x20;
x3 = x30;

m=m_c + 2*m_w;
I = m_c*d^2+2*m_w*R^2+I_c+2*I_m;
m_1 = 0.25*R^-2*r^2*(m*R^2+I)+I_w;
m_2 = 0.25*R^-2*r^2*(m*R^2-I);
M = [m_1 m_2;m_2 m_1];
D = eye(2,2) * d_m1;
T_d = [0.001 0.001]';
```

```
%--------------------- variables de kalman
p1 = 1e2;
p2 = 1e2;
p3 = 1e2;
q1 = 5e-1;
q2 = 5e-1;
q3 = 5e-1;
R1 = 1e2;
R2 = 1e2;
R3 = 1e2;

g1=1;
g3=1;
g2=1;

W1 = [0 0]';
W2 = [1 3]';
W3 = [0 0]';
P1 = p1* eye(2,2);
P2 = p2* eye(2,2);
P3 = p3* eye(2,2);

Q1 = q1* eye(2,2);
Q2 = q2* eye(2,2);
Q3 = q3* eye(2,2);


A = eye(3); % Matriz de transici n de estado
C_k = eye(3,3); % Matriz de observaci n

Q = eye(3) * 0.01; % Covarianza del ruido de proceso
R_k = eye(3) * 0.1; % Covarianza del ruido de medici n

P1e = eye(3); % Estimaci n inicial de la covarianza

xe1_hist = [];
P1e_hist = [];

%comenzamos con la integracion de las variables

Z1 = [0 0 0]';

for t = t0:dt:tf
    Z1 = [tanh(x1(1)), tanh(x1(1))]';
    Z2 = [tanh(x1(2)), tanh(x1(3))]';
    Z3 = [tanh(x1(3)), tanh(x1(3))]';
```

```
    x1_hist = [x1_hist ; [t x1']];


    x2_hist = [x2_hist ; [t x2']];
    % xe2_hist=[x1e_hist [t x2e]];

    x3_hist = [x3_hist ; [t x3']];
    % xe3_hist=[x3e_hist [t x3e]];

    %integracion
    dx_dt1 = 0.5*r*[cos(x1(3)), cos(x1(3));
        sin(x1(3)), sin(x1(3));
        R^-1, -R^-1] * x2;
    x1 = x1 + dx_dt1*dt;


    C = 0.5*R^-1*r^2*m_c*d*[0, dx_dt1(3);
                            -dx_dt1(3), 0];

    dx_dt2 = M^-1*(-C*x2-D*x2-T_d+N*K_t*x3);
    x2 = x2 + dx_dt2*dt;

    u= [4 3]';

    dx_dt3 = L_a^(-1)*(u-R_a*x3-N*K_e*x2);
    x3 = x3 + dx_dt3*dt;


    y_1 = C_k * x1; % Medici n real
    % --------------------- Filtro de Kalman
    H1 = Z1;
    K1 = P1*H1*inv(R1+H1'*P1*H1);
    W1 = W1+g1*K1*(x1(1)-x1e(1));
    P1 = P1-K1*H1'*P1+Q1;

    H2 = Z2;
    K2 = P2*H2*inv(R2+H2'*P2*H2);
    W2 = W2+g2*K2*(x1(2)-x2e(1));
    P2 = P2-K2*H2'*P2+Q2;

    H3 = Z3;
    K3 = P3 * H3*inv(R3+H3'*P3*H3);
    W3 = W3+g3*K3*(x1(3)-x3e(1));
    P3 = P3-K3*H3'*P3+Q3;

    x1e = W1'*Z1+0.0001*u(1) + 0.0001*u(2);
    x2e = W2'*Z2+0.0001*u(1) + 0.0001*u(2);
    x3e = W3'*Z3+0.0001*u(1) + 0.0001*u(2);
```

```

    % x1e_pred = A * x1e;
    % P1e_pred = A * P1e * A' + Q;
    %

    % K = P1e_pred * H1'*inv(H1 * P1e_pred * H1' + R_k); % ...
       Ganancia de Kalman
    % x1e = x1e_pred + K * (y_1 - (C_k * x1e_pred));
    % P1e = (eye(3) - K * H1) * P1e_pred;

    xe1_hist=[xe1_hist; [t x1e']];
    xe2_hist=[xe2_hist; [t x2e']];
    xe3_hist=[xe3_hist; [t x3e']];
    %P1e_hist = [P1e_hist, P1];


end
figure(1)
hold on
plot(x1_hist(:,1),x1_hist(:,2),'-b')
plot(xe1_hist(:,1),xe1_hist(:,2),'--b')

plot(x1_hist(:,1),x1_hist(:,3),'-g')
plot(xe2_hist(:,1),xe2_hist(:,2),'--g')

plot(x1_hist(:,1),x1_hist(:,4),'-r')
plot(xe3_hist(:,1),xe3_hist(:,2),'--r')

grid on
legend('posicion x','estimacion posicion x','posicion ...
   y','estimacion posicion y');
title('x11');

return
figure (2)
plot(x2_hist(:,1),x2_hist(:,2),'-b')
plot(x2_hist(:,1),x2_hist(:,3),'-r')
hold on
grid on
legend('velocidad angular derecha','velocidad angular ...
   izquierda');
title('x2');

figure (3)
plot(x3_hist(:,1),x3_hist(:,2),'-b')
hold on
plot(x3_hist(:,1),x3_hist(:,3),'-r')
hold on
grid on
```

```
legend('corriente derecha','corriente izquierda');
title('x3');
% haganlo funcionar para las otras dos x
```



Que como podemos observar la aproximacion es muy buena.

Conclusiones

No siempre se pueden tener sensores en todos los robots por lo que el uso de observadores es bueno, podemos ir desde e uso del filtro de kalman o bien podemos usar una red neuronal para encontrar los pesos óptimos para poder tener la observación de los estados, no obstante esto no siempre es posible de efectuar según la linealidad del sistema.