
UNIVERSIDAD DE GUADALAJARA
CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS



Sistemas Inteligentes III

Red neuronal de kalman extendido

Nombre:	Sebastian Martínez Muñoz 217567225
---------	------------------------------------

23 de mayo de 2023, Guadalajara

Contenido

Introducción	2
Modelo del robot	2
Filtro de Kalman	7
Fase de predicción	7
Fase de corrección	8
Implementación	8
Conclusiones	15

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.

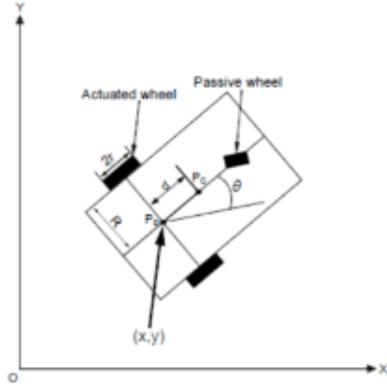


Figura 4.1: Robot Móvil con dos ruedas actuadas.

Modelo del robot

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

$$\dot{x}_1 = J(x_1)x_2 \quad (1)$$

$$\dot{x}_2 = M^{-1} - C(\dot{x}_1)x_2 - Dx_2/\tau_d + NK_T x_3 \quad (2)$$

$$\dot{x}_3 = L_a^{-1}(u - R_a x_3 - NK_E x_2) \quad (3)$$

Donde tenemos las siguientes variables.

$$x_1 = [x_{11} \quad x_{12} \quad x_{13}]^T$$

$$x_2 = [x_{21} \quad x_{22}]^T$$

$$x_3 = [x_{31} \quad x_{32}]^T$$

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

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

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

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

De los cuales se usaran los siguientes parámetros.

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

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

$$m = m_c + 2m_w$$

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

$$\tau = [\tau_1 \quad \tau_2]^T$$

$$\tau_d = [\tau_{d1} \quad \tau_{d2}]^T$$

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.

```

1 clear all
2 close all
3 clc
4
5 %% Declaracion de los parametros fisicos
6
7 R = 0.75;
8 d=0.3;
9 r=0.15;
10 m_c=30;
11 m_w = 1;
12 I_c = 15.625;
13 I_w=0.005;
14 I_m = 0.0025;
15 R_a = eye(2,2) * 2.5;
16 L_a = eye(2,2) * 0.048;
17 K_e = eye(2,2) * 0.02;
18 N= eye(2,2) * 62.55;
19 K_t = eye(2,2) * 0.2613;
20 d_m1 = 0.5;
21 d_m2 = d_m1;
22
23 %% declaracion de las matrices y vectores necesarios
24
25
26
27 % estado inicial x1
28 x1 = [0;1;2]; % [x;y;theta]
29 x2 = [0.5*pi;0.5*pi]; %velocidad angular de las ruedas
30
31
32 x3 = [0,0]; % corrientes
33
34
35 % declaramos los limites de tiempo
36 t0 = 0;
37 tf = 10;
38 %declaracion de las condiciones iniciales
39 x10 = [0 0 0]';
40 x20 = [1 1]';

```

```

41 x30 = [0 0]';
42
43 % declaracion de la estimacion inicial
44 x1e =[1 2 1]';
45 x2e =[2 1]';
46 x3e =[0.5 0]';
47
48 % incializamos los vectores
49 % valores reales
50 x1_hist = [];
51 x2_hist = [];
52 x3_hist = [];
53 % valores medidos
54 xe1_hist=[];
55 xe2_hist=[];
56 xe3_hist=[];
57
58
59 dt = 0.01; %definimos el periodo de tiempo
60
61 x1 = x10;
62 x2 = x20;
63 x3 = x30;
64
65 m=m_c + 2*m_w;
66 I = m_c*d^2+2*m_w*R^2+I_c+2*I_m;
67 m_1 = 0.25*R^-2*r^2*(m*R^2+I)+I_w;
68 m_2 = 0.25*R^-2*r^2*(m*R^2-I);
69 M = [m_1 m_2;m_2 m_1];
70 D = eye(2,2) * d_m1;
71 T_d = [0.001 0.001]';
72
73
74 %comenzamos con la integracion de las variables
75 figure(1)
76 hold on
77 for t = t0:dt:tf
78
79     x1_hist = [x1_hist ; [t x1']];
80     % xe1_hist=[x1e_hist [t x1e]];
81
82     x2_hist = [x2_hist ; [t x2']];
83     % xe2_hist=[x1e_hist [t x2e]];
84
85     x3_hist = [x3_hist ; [t x3']];
86     % xe3_hist=[x3e_hist [t x3e]];
87
88     %integracion
89     dx_dt1 = 0.5*r*[cos(x1(3)), cos(x1(3))];

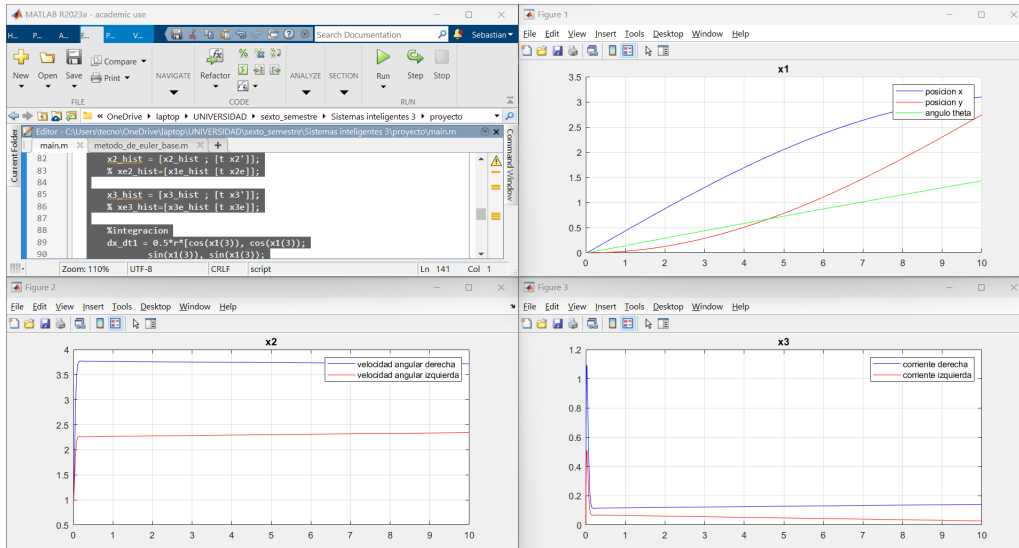
```

```

90         sin(x1(3)), sin(x1(3));
91         R^-1, -R^-1] * x2;
92
93     x1 = x1 + dx_dt1*dt;
94
95     C = 0.5*R^-1*r^2*m_c*d*[0, dx_dt1(3);
96                             -dx_dt1(3), 0];
97
98     dx_dt2 = M^-1*(-C*x2-D*x2-T_d+N*K_t*x3);
99     x2 = x2 + dx_dt2*dt;
100
101     u= [5 3]';
102
103     dx_dt3 = L_a^(-1)*(u-R_a*x3-N*K_e*x2);
104     x3 = x3 + dx_dt3*dt;
105
106 end
107
108 plot(x1_hist(:,1),x1_hist(:,2),'-b')
109 hold on
110 plot(x1_hist(:,1),x1_hist(:,3),'-r')
111 hold on
112 plot(x1_hist(:,1),x1_hist(:,4),'-g')
113 hold on
114 grid on
115 legend('posicion x','posicion y','angulo theta');
116 title('x1');
117
118 figure (2)
119 plot(x2_hist(:,1),x2_hist(:,2),'-b')
120 hold on
121 plot(x2_hist(:,1),x2_hist(:,3),'-r')
122 hold on
123 grid on
124 legend('velocidad angular derecha','velocidad angular ...
125         izquierda');
126 title('x2');
127
128 figure (3)
129 plot(x3_hist(:,1),x3_hist(:,2),'-b')
130 hold on
131 plot(x3_hist(:,1),x3_hist(:,3),'-r')
132 hold on
133 grid on
134 legend('corriente derecha','corriente izquierda');
135 title('x3');

```

Del cual podemos observar las siguientes gráficas.



Donde se 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 se puede modificar el algoritmo de tal manera que solo es necesario agregar las matrices.

Los observadores o estimadores constan de dos partes fundamentales, la predicción y la corrección de los cuales el filtro de kalman no esta exento.

Dicho de otro modo debemos conocer la predicción 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*

$$\hat{\mathbf{x}}_{k|k-1} = \Phi_k \mathbf{x}_{k-1|k-1}$$

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

Fase de corrección

$$\begin{aligned}\tilde{\mathbf{y}}_k &= \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1} \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1}\end{aligned}$$

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.

```

1  % Inicializamos variables
2  A = eye(3); % Matriz de transición de estado
3  C_k = eye(3,3); % Matriz de observación
4
5  Q = eye(3) * 0.01; % Covarianza del ruido de proceso
6  R_k = eye(3) * 0.1; % Covarianza del ruido de medición
7
8
9  P1e = eye(3); % Estimación inicial de la covarianza
10
11  x1e_hist = [];
12  P1e_hist = [];
13  x1e_pred = A * x1e;
14  for t = t0:dt:tf
15      P1e_pred = A * P1e * A' + Q;
16
17      y_1 = C_k * x1; % Medición real
18      K = P1e_pred * C_k' / (C_k * P1e_pred * C_k' + R_k); % ...
          Ganancia de Kalman
19      x1e = x1e_pred + K * (y_1 - C_k * x1e_pred);
20      P1e = (eye(3) - K * C_k) * P1e_pred;
21

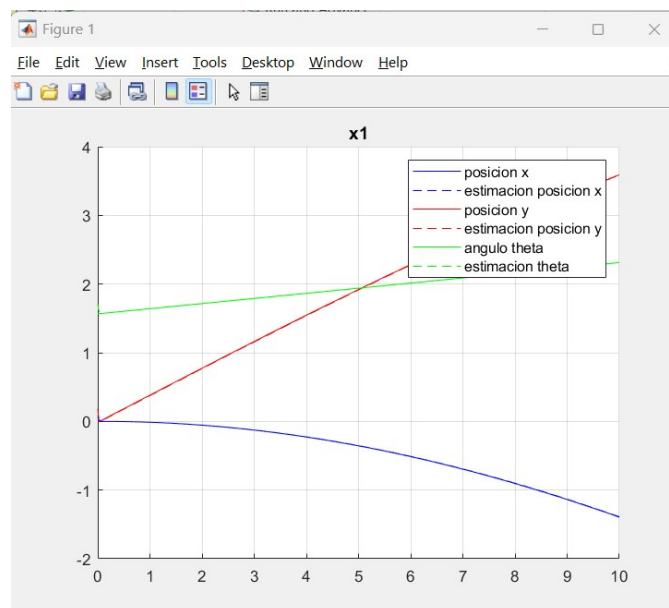
```

```

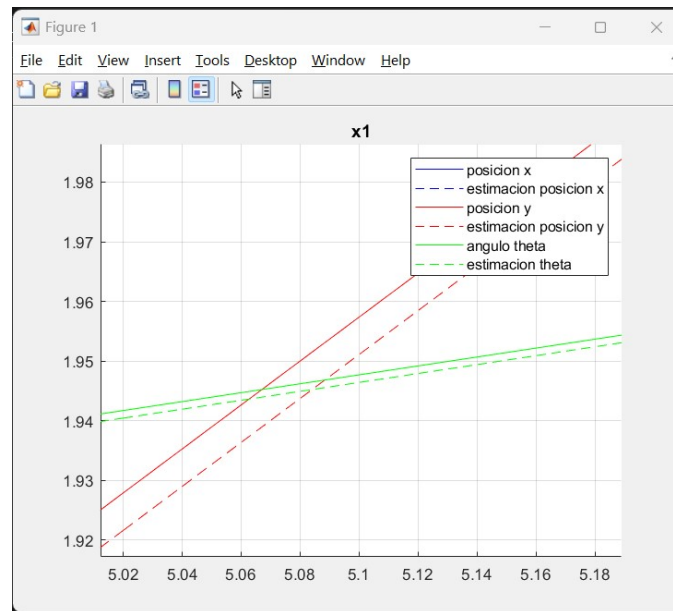
22     xe1_hist=[xe1_hist; [t x1e']];
23     Ple_hist = [Ple_hist, diag(P1e)'];
24 end
25
26 plot(x1_hist(:,1),x1_hist(:,2),'-b')
27 hold on
28 plot(xe1_hist(:,1),xe1_hist(:,2),'--b')
29 hold on
30
31 plot(x1_hist(:,1),x1_hist(:,3),'-r')
32 hold on
33 plot(xe1_hist(:,1),xe1_hist(:,3),'--r')
34 hold on
35
36 plot(x1_hist(:,1),x1_hist(:,4),'-g')
37 hold on
38 plot(xe1_hist(:,1),xe1_hist(:,4),'--g')
39 hold on
40 grid on
41 legend('posicion x','estimacion posicion x','posicion ...
      y','estimacion posicion y','angulo theta','estimacion theta');
42 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 (línea punteada) se hace al estado real (línea sólida).

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

Al implementar con una red neuronal obtenemos el siguiente código el cual nos otorga el siguiente resultado.

```

1 clear all
2 close all
3 clc
4
5 %% Declaracion de los parametros fisicos
6
7 R = 0.75;
8 d=0.3;
9 r=0.15;
10 m_c=30;
11 m_w = 1;
12 I_c = 15.625;
13 I_w=0.005;
14 I_m = 0.0025;
15 R_a = eye(2,2) * 2.5;
16 L_a = eye(2,2) * 0.048;
17 K_e = eye(2,2) * 0.02;
18 N= eye(2,2) * 62.55;
19 K_t = eye(2,2) * 0.2613;
20 d_m1 = 0.5;
21 d_m2 = d_m1;

```

```

22
23 %% declaracion de las matrices y vectores necesarios
24
25
26
27 % estado inicial x1
28 x1 = [0;1;2]; % [x;y;theta]
29 x2 = [0.5*pi;0.5*pi]; %velocidad angular de las ruedas
30 x3 = [0,0]; % corrientes
31
32
33 % declaramos los limites de tiempo
34 t0 = 0;
35 tf = 10;
36 %declaracion de las condiciones iniciales
37 x10 = [1 2 pi/2]';
38 x20 = [1 1]';
39 x30 = [0 0]';
40
41 % declaracion de la estimacion inicial
42 x1e =[1 2]';
43 x2e =[2 1]';
44 x3e =[0.5 0]';
45
46 % incializamos los vectores
47 % valores reales
48 x1_hist = [];
49 x2_hist = [];
50 x3_hist = [];
51
52 % valores estimados
53 xe1_hist=[];
54 xe2_hist=[];
55 xe3_hist=[];
56
57
58 dt = 0.01; %definimos el periodo de tiempo
59
60 x1 = x10;
61 x2 = x20;
62 x3 = x30;
63
64 m=m_c + 2*m_w;
65 I = m_c*d^2+2*m_w*R^2+I_c+2*I_m;
66 m_1 = 0.25*R^-2*r^2*(m*R^2+I)+I_w;
67 m_2 = 0.25*R^-2*r^2*(m*R^2-I);
68 M = [m_1 m_2;m_2 m_1];
69 D = eye(2,2) * d_m1;
70 T_d = [0.001 0.001]';

```

```

71
72
73
74 %----- variables de kalman
75 p1 = 1e2;
76 p2 = 1e2;
77 p3 = 1e2;
78 q1 = 5e-1;
79 q2 = 5e-1;
80 q3 = 5e-1;
81 R1 = 1e2;
82 R2 = 1e2;
83 R3 = 1e2;
84
85 g1=1;
86 g3=1;
87 g2=1;
88
89 W1 = [0 0]';
90 W2 = [1 3]';
91 W3 = [0 0]';
92 P1 = p1* eye(2,2);
93 P2 = p2* eye(2,2);
94 P3 = p3* eye(2,2);
95
96 Q1 = q1* eye(2,2);
97 Q2 = q2* eye(2,2);
98 Q3 = q3* eye(2,2);
99
100
101 A = eye(3); % Matriz de transición de estado
102 C_k = eye(3,3); % Matriz de observación
103
104 Q = eye(3) * 0.01; % Covarianza del ruido de proceso
105 R_k = eye(3) * 0.1; % Covarianza del ruido de medición
106
107 P1e = eye(3); % Estimación inicial de la covarianza
108
109 xe1_hist = [];
110 P1e_hist = [];
111
112 %comenzamos con la integracion de las variables
113
114 Z1 = [0 0 0]';
115
116 for t = t0:dt:tf
117     Z1 = [tanh(x1(1)), tanh(x1(1))]' ;
118     Z2 = [tanh(x1(2)), tanh(x1(3))]' ;
119     Z3 = [tanh(x1(3)), tanh(x1(3))]' ;

```

```

120
121     x1_hist = [x1_hist ; [t x1']];
122
123
124     x2_hist = [x2_hist ; [t x2']];
125     % xe2_hist=[x1e_hist [t x2e]];
126
127     x3_hist = [x3_hist ; [t x3']];
128     % xe3_hist=[x3e_hist [t x3e]];
129
130     %integracion
131     dx_dt1 = 0.5*r*[cos(x1(3)), cos(x1(3));
132                   sin(x1(3)), sin(x1(3));
133                   R^-1, -R^-1] * x2;
134     x1 = x1 + dx_dt1*dt;
135
136
137     C = 0.5*R^-1*r^2*m_c*d*[0, dx_dt1(3);
138                             -dx_dt1(3), 0];
139
140     dx_dt2 = M^-1*(-C*x2-D*x2-T_d+N*K_t*x3);
141     x2 = x2 + dx_dt2*dt;
142
143     u= [4 3]';
144
145     dx_dt3 = L_a^(-1)*(u-R_a*x3-N*K_e*x2);
146     x3 = x3 + dx_dt3*dt;
147
148
149     y_1 = C_k * x1; % Medicin real
150     % ----- Filtro de Kalman
151     H1 = Z1;
152     K1 = P1*H1*inv(R1+H1'*P1*H1);
153     W1 = W1+g1*K1*(x1(1)-x1e(1));
154     P1 = P1-K1*H1'*P1+Q1;
155
156     H2 = Z2;
157     K2 = P2*H2*inv(R2+H2'*P2*H2);
158     W2 = W2+g2*K2*(x1(2)-x2e(1));
159     P2 = P2-K2*H2'*P2+Q2;
160
161     H3 = Z3;
162     K3 = P3 * H3*inv(R3+H3'*P3*H3);
163     W3 = W3+g3*K3*(x1(3)-x3e(1));
164     P3 = P3-K3*H3'*P3+Q3;
165
166     x1e = W1'*Z1+0.0001*u(1) + 0.0001*u(2);
167     x2e = W2'*Z2+0.0001*u(1) + 0.0001*u(2);
168     x3e = W3'*Z3+0.0001*u(1) + 0.0001*u(2);

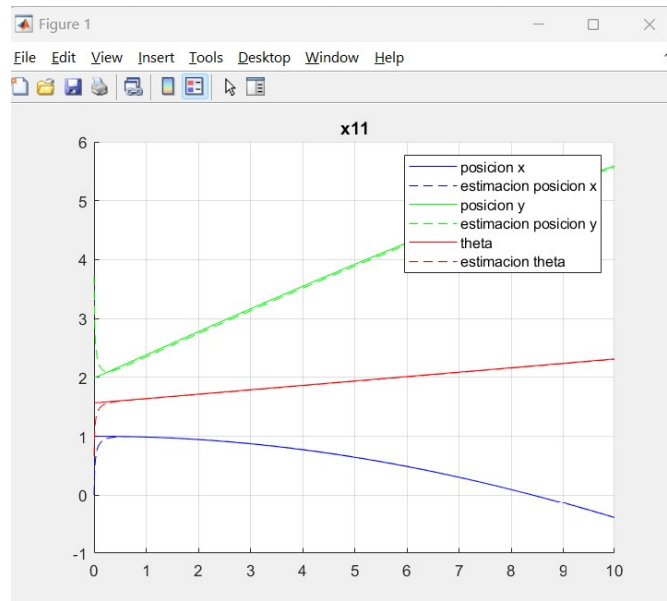
```

```

169
170     % x1e_pred = A * x1e;
171     % P1e_pred = A * P1e * A' + Q;
172     %
173
174     % K = P1e_pred * H1' * inv(H1 * P1e_pred * H1' + R_k); % ...
        Ganancia de Kalman
175     % x1e = x1e_pred + K * (y_1 - (C_k * x1e_pred));
176     % P1e = (eye(3) - K * H1) * P1e_pred;
177
178     xe1_hist=[xe1_hist; [t x1e']];
179     xe2_hist=[xe2_hist; [t x2e']];
180     xe3_hist=[xe3_hist; [t x3e']];
181     %P1e_hist = [P1e_hist, P1];
182
183
184 end
185 figure(1)
186 hold on
187 plot(x1_hist(:,1),x1_hist(:,2),'-b')
188 plot(xe1_hist(:,1),xe1_hist(:,2),'--b')
189
190 plot(x1_hist(:,1),x1_hist(:,3),'-g')
191 plot(xe2_hist(:,1),xe2_hist(:,2),'--g')
192
193 plot(x1_hist(:,1),x1_hist(:,4),'-r')
194 plot(xe3_hist(:,1),xe3_hist(:,2),'--r')
195
196 grid on
197 legend('posicion x','estimacion posicion x','posicion ...
        y','estimacion posicion y');
198 title('x11');
199
200 return
201 figure (2)
202 plot(x2_hist(:,1),x2_hist(:,2),'-b')
203 plot(x2_hist(:,1),x2_hist(:,3),'-r')
204 hold on
205 grid on
206 legend('velocidad angular derecha','velocidad angular ...
        izquierda');
207 title('x2');
208
209 figure (3)
210 plot(x3_hist(:,1),x3_hist(:,2),'-b')
211 hold on
212 plot(x3_hist(:,1),x3_hist(:,3),'-r')
213 hold on
214 grid on

```

```
215 legend('corriente derecha','corriente izquierda');  
216 title('x3');  
217 % 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 el 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.