PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

ESCUELA DE POSGRADO

Curso:

CONTROL ÓPTIMO

Tema:

Feedforward + Feedback + Observador

Sensor Fusión

Presentado por:

CONTRERAS MARTINEZ, DIMEL ARTURO

Docente:

DR. ANTONIO MORÁN

2016

1. Para la planta “Motor con tornillo sinfín” con control utilizando “feedforward + feedback”, se diseñará 2 tipos de observadores Optimal y Filtro de Kalman. Luego se compararán las 2 respuestas obtenidas.

Modelo de la planta Motor + tornillo sinfín:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ di/dt \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ b_3 \end{bmatrix} u$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ b_3 \end{bmatrix}$$

El sistema se representa::

$$\dot{X} = AX + Bu$$

$$y = CX$$

>>Script en Matlab

- Parametros del Sistema:

```
R = 1.1;
L = 0.001;
Kt = 0.0573;
Kb = 0.05665;
I = 4.326e-5;
p = 0.0075;
m = 0.5*1.00;
c = 10;
r = 0.015;
alfa = 45*pi/180;
d = m + 2*pi*I*tan(alfa)/(p*r);
a22 = -c/d;
a23 = Kt*tan(alfa)/(r*d);
a32 = -2*pi*Kb/(p*L);
a33 = -R/L;
b31 = 1/L;
w21 = -1/d;
A = [ 0   1   0
      0  a22 a23
      0  a32 a33 ];
B = [ 0
      0
      b31 ];
Wf = [ 0
       w21
       0 ];

r = 1*0.5;            % Posición deseada
voltmax = 1*24;       % Voltaje máximo
Fseca = 1.5*15;          % Fricción
```

Considerando la siguiente función de costo:

$$J = \int_0^\infty q_1 x^2 + q_2 \dot{x}^2 + q_3 i^2 + u^2$$

$$Q_i = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix}$$

Modelo Feedback + Feedforward

Modelo a seguir:

Se utiliza el modelo SKY HOOK DAMPER por poseer buena respuesta.

$$\dot{Z} = \bar{A}Z + \bar{B}r$$

$$y_m = \bar{C}Z$$

$w = 2\pi F$

$\xi = 0.9$

$Am = \begin{bmatrix} 0 & 1 \\ -w^2 & -2\xi w \end{bmatrix}$

$Bm = \begin{bmatrix} 0 \\ w^2 \end{bmatrix}$

$Cm = [1\ 0]$

Se considera el sistema de orden 2, con z_{2x1}.

$$Z = \begin{bmatrix} x_m \\ \dot{x_m} \end{bmatrix}$$

$$C = [1\ 0]$$

Siendo y la variable a controlar.

>>Script en Matlab

```
eta = 0.90;
f = 1.5*0.4;    % Cambiar factor para hacer más rápida
                % la respuesta del sistema controlado (Probar con 1 - 1.5)
w = 2*pi*f;
Am = [   0       1
       -w*w     -2*eta*w ];
Bm = [ 0
         w*w ];
Cm = [ 1   0 ];

dt = 0.001;     ti = 0;      tf = 10;
t = ti:dt:tf;    t = t';    nt = length(t);
rast = 0.3*ones(nt,1);
[Amk,Bmk] = c2d(Am,Bm,dt);
xm = [ 0
           0 ];
k = 1;
for tt = ti:dt:tf
   xm1(k,1) = xm(1,1);       xm2(k,1) = xm(2,1);
   xm = Amk*xm + Bmk*rast(k,1);
   k = k + 1;
end
figure(1);    plot(t,xm1);
title('Posicion - Modelo Deseado');
grid;
```

Seguimiento:

Para que el sistema siga al modelo a seguir, “y” debe ser igual a “ym”. Para ello se propone la función de costo:

$$J = \int_0^\infty (q(y - y_m)^2 + ru^2)dt$$

$$J = \int_0^\infty [X\ Z] \begin{bmatrix} C^T qC & -C^T q\bar{C} \\ -\bar{C}qC & -\bar{C}^T q\bar{C} \end{bmatrix} \begin{bmatrix} X \\ Z \end{bmatrix} + u^T Ru$$

El vector de estado de al unir el modelo y el sistema resulta:

$$\mathcal{X} = \begin{bmatrix} X \\ Z \end{bmatrix}$$

$$\mathcal{A} = \begin{bmatrix} A & 0 \\ 0 & \bar{A} \end{bmatrix}$$

$$\boldsymbol{J} = \int_0^\infty \mathcal{X}^T Q \mathcal{X} + u^T Ru$$

$$u = -\mathcal{K}\mathcal{X}$$

$$\mathcal{K} = r^{-1}\mathcal{B}^T\mathcal{P}$$

$$\mathcal{K} = r^{-1}[\mathrm{B}^T\ 0]\begin{bmatrix} P_{11} & P_{12} \\ P_{12}{}^T & P_{22} \end{bmatrix}$$

$P_{11}\ se\ resuelve\ mediante\ Riccati$:

$$A^T P_{11} + P_{11}A - P_{11}Br^{-1}B^T P_{11} + C^T qC = 0$$

$P_{12}\ se\ resuelve\ mediante\ Lyapunov$:

$$(A^T - P_{11}Br^{-1}B^T)P_{12} + P_{12}A - C^T q\bar{C} = 0$$

Luego las ganancias:

$$K_x = r^{-1}B^T P_{11}$$

$$K_z = r^{-1}B^T P_{12}$$

Entonces la señal de control es :

$$u = [K_x\ K_z]\ \mathcal{X}$$

$$u = [r^{-1}B^T P_{11} \quad r^{-1}B^T P_{12}]\begin{bmatrix} X \\ Z \end{bmatrix}$$

>>Script en Matlab

```
q = 1e9; %input('Peso del error [y-ym]   [1e9]: ');     % 4e8
Q = [ q ];
QQ = C'*Q*C;
RR = [ 1 ];
P11 = are(A,B*inv(RR)*B',QQ);
APR = A'-P11*B*inv(RR)*B';
CQCm = -C'*Q*Cm;
P12 = lyap(APR,Am,CQCm);
Kfb = inv(RR)*B'*P11;
Kff = inv(RR)*B'*P12;
```

Modelo de los ruidos

En la planta:

Se considerará la fricción seca Fseca, aunque no sea un ruido gaussiano.

En el sensor:

```
nmean = 0
nstd = 0.1
n = randn(nt,1)
n = (n-mean(n))*nstd/std(n) + nmean
R = sum((n-nmean).*(n-nmean))/(nt-1)
```

Observadores

Verificamos la observabilidad del sistema:

$$\text{OBS} = [\text{C} \quad \text{CA} \quad \text{CA}^2]$$

$$\det(OBS) \ \ debe\ ser\ diferente\ de\ 0$$

$$Qo = \begin{bmatrix} q1o & 0 & 0 \\ 0 & q2o & 0 \\ 0 & 0 & q3o \end{bmatrix}$$

$$Ro = I_{3x3}$$

Xo -> Estimación de los estados:

EL OBSERVADOR OPTIMAL:

$$So = are(A, Ro^{-1}, Qo)$$

Ganancia del observador:

$$L = Ro^{-1} So\, C^T (CC^T)^{-1}$$

Discretizando las matrices para el observador

Con tiempo de muestreo dt.

$$[Aok\,, Bok] = c2d(A - LC, B, dt)$$

$$[Aok\,, Lok] = c2d(A - LC, L, dt)$$

Se considera que la salida del sistema es x, osea solo se puede medir (hay sensor) la posición, sin embargo ésta salida también posee ruido:

$$y = x + ruido$$

El observador que se obtiene:

$$Xo = Aok * Xo + Bok * u + Lok * y$$

Condición inicial:

$$Xo = [0; 0; 0]$$

>>Script en Matlab

```
Obs = [ C;   C*A;   C*A*A ];
detObs = det(Obs)


q1o = input('Peso q1o : ');
q2o = input('Peso q2o : ');
q3o = input('Peso q3o : ');
Qo = diag([ q1o q2o q3o ]);
Ro = eye(3);
So = are(A,inv(Ro),Qo);
L = inv(Ro)*So*C'*inv(C*C');
[Aok Bok] = c2d(A-L*C,B,dt);
[Aok Lok] = c2d(A-L*C,L,dt);
[Aok Wfok] = c2d(A-L*C,Wf,dt);
 xo = [ 0;  0;   0 ];
```

Controlador y observador:

$$u = [r^{-1}B^T P_{11} \quad r^{-1}B^T P_{12}] \begin{bmatrix} X_o \\ Z \end{bmatrix}$$

Simulación :

```
umax = 24;
Pot = 200;
Fric = 0*10;
[Ak Bk] = c2d(A,B,dt);
[Ak Wk] = c2d(A,Wf,dt);
x   = [ 0.1;  -0.1;  0.5 ];% Vector de estado inicial
xm = [ 0;  0 ];
k = 1;
for tt = ti:dt:tf
    y = C*x + n(k,1);
    yy(k,1) = y;
    xm1(k,1) = xm(1,1);
    xm2(k,1) = xm(2,1);
    xx1(k,1) = x(1,1);       xx2(k,1) = x(2,1);   xx3(k,1) = x(3,1);
    xxo1(k,1) = xo(1,1);   xxo2(k,1) = xo(2,1);   xxo3(k,1) = xo(3,1);
    time(k,1) = tt;
    u = -Kfb*xo - Kff*xm
    if(u > umax)
       u = umax;
    elseif(u < -umax)
       u = -umax;
    end
    if(x(2,1) >= 0)
        F = Fric;
    elseif(x(2,1) < 0)
        F = -Fric;
    end
    uc(k,1) = u;
    pot(k,1) = u*x(3,1);
    x = Ak*x + Bk*u + Wk*F;
    xm = Amk*xm + Bmk*rast(k,1);

    xo = Aok*xo + Bok*u + Lok*y;
    k = k + 1;
end
```

Ploteo gráficas: Es la misma en ambos casos de observadores.

```
figure(1);
plot(t,xx1,'-r',t,xxo1,'-b',t,xm1,'--g');     grid;
title('Posición m');
legend('pos real','pos obs','pos modelo')
figure(2);
plot(t,xx2,'-r',t,xxo2,'-b');     grid;      title('Velocidad');
legend('vel real','vel obs')
figure(3);
plot(t,xx3,'-r',t,xxo3,'-b');     grid;      title('Corriente');
legend('I vel real','I obs')
```

***Pruebas**

Condiciones:

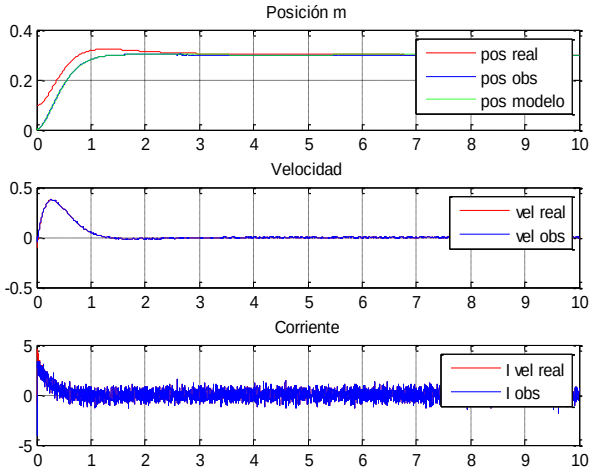
Ruido_sensor -> media = 0 , varianza = 0.01

Fricción seca = 0*10 (Ruido de planta)

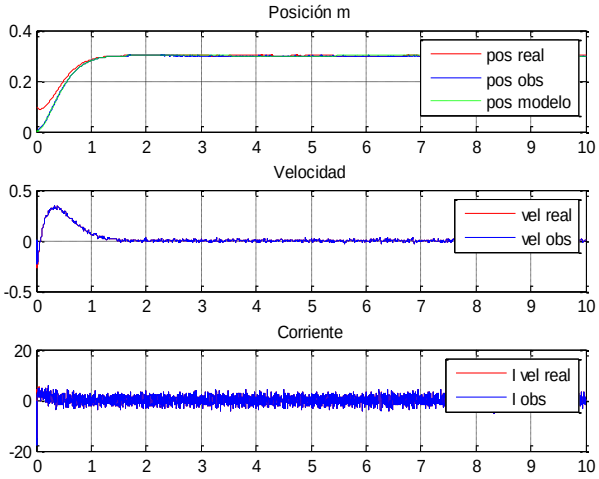
Referencia : x = 0.3

Variando pesos del observador

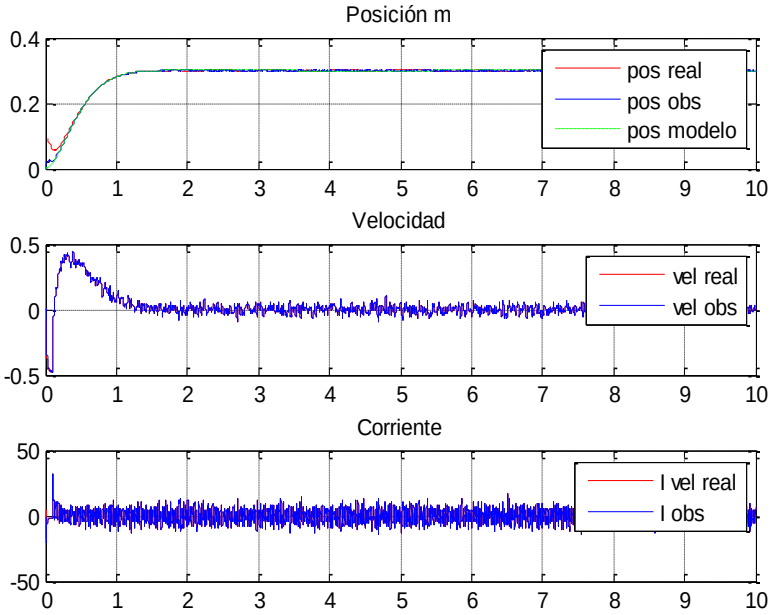
- Peso q1o = 1 , q2o = 0 , q3o = 0



- Peso q1o = 10 , q2o = 10 , q3o = 10



- Peso q1o = 100 , q2o = 100 , q3o = 100

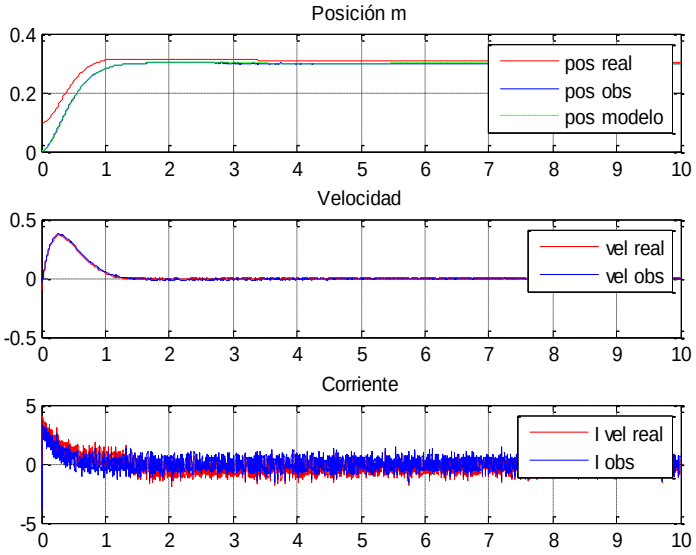


Condiciones:

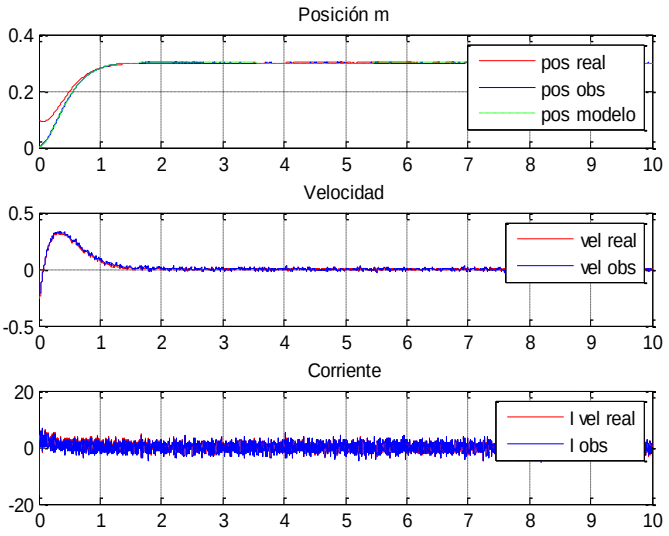
Ruido_planta-> media = 0 , varianza = 0.01

Fricción seca = 0.2*10 (Ruido de planta)

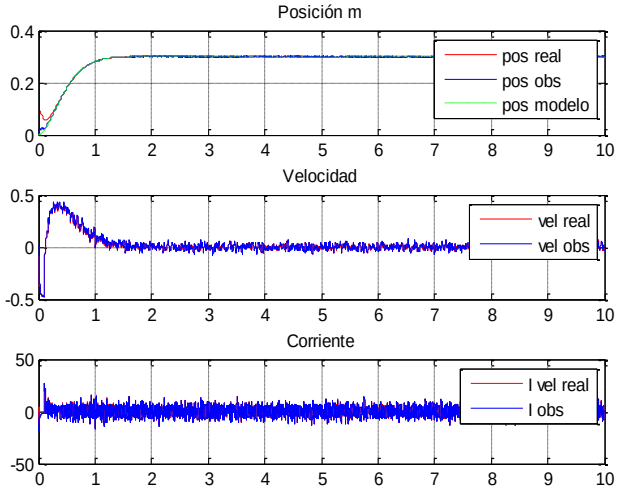
- Peso q1o = 1 , q2o = 0 , q3o = 0



- Peso q1o = 10 , q2o = 10 , q3o = 10



- Peso q1o = 100 , q2o = 100 , q3o = 100

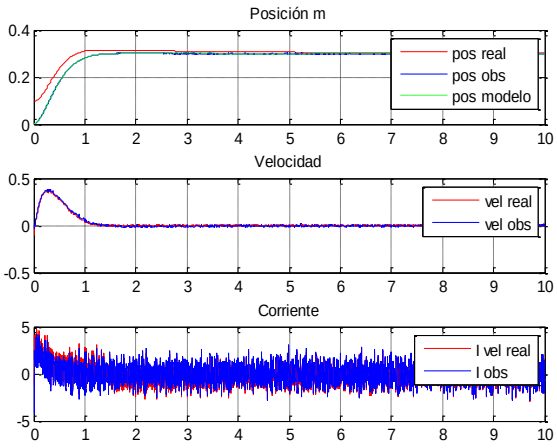


Condiciones:

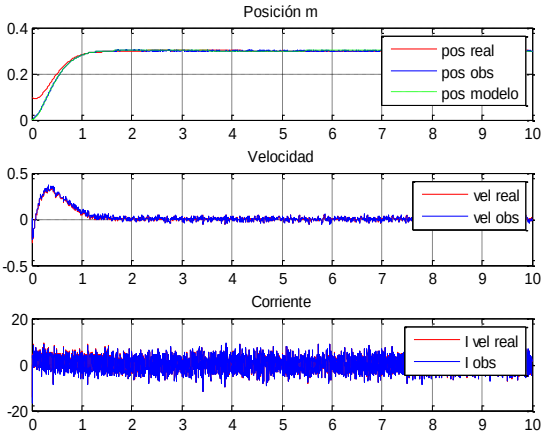
Ruido_planta-> media = 0 , varianza = 0.02

Fricción seca = 0.2*10 (Ruido de planta)

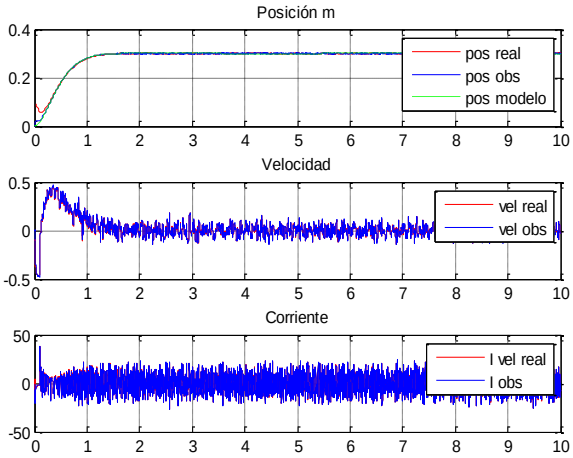
- Peso q1o = 1 , q2o = 0 , q3o = 0



- Peso q1o = 10 , q2o = 10 , q3o = 10



- Peso q1o = 100 , q2o = 100 , q3o = 100

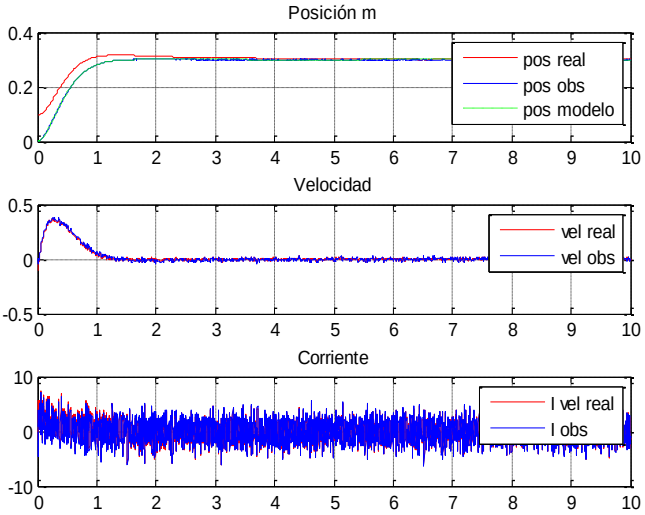


Condiciones:

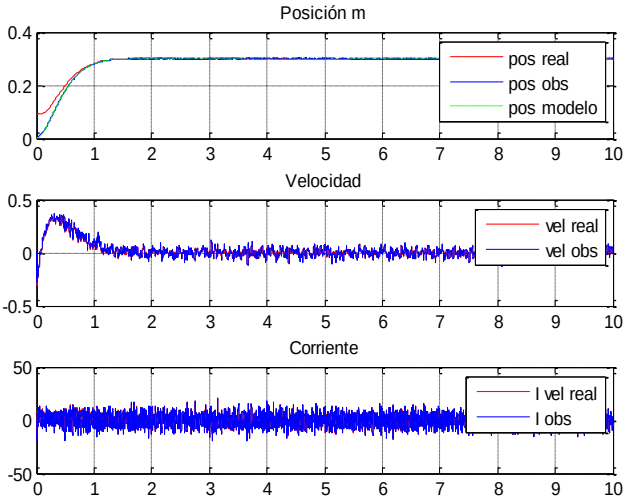
Ruido_planta-> media = 0 , varianza = 0.04

Fricción seca = 0.2*10 (Ruido de planta)

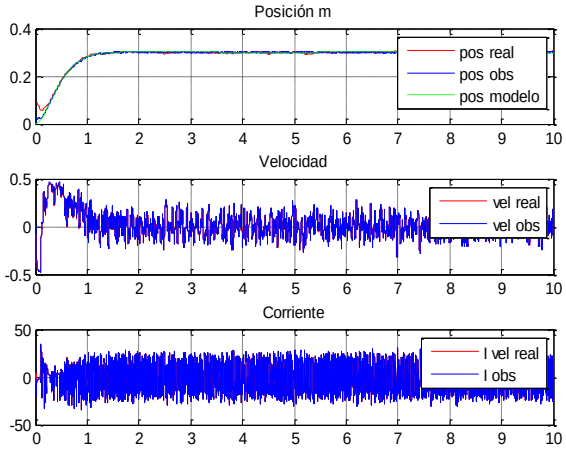
- Peso q1o = 1 , q2o = 0 , q3o = 0



- Peso q1o = 10 , q2o = 10 , q3o = 10



- Peso q1o = 100 , q2o = 100 , q3o = 100

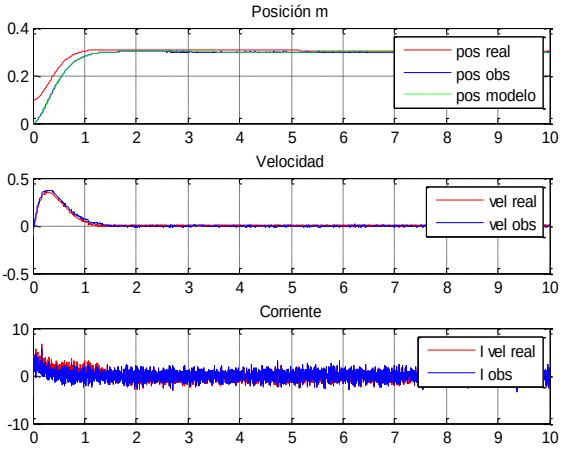


Condiciones:

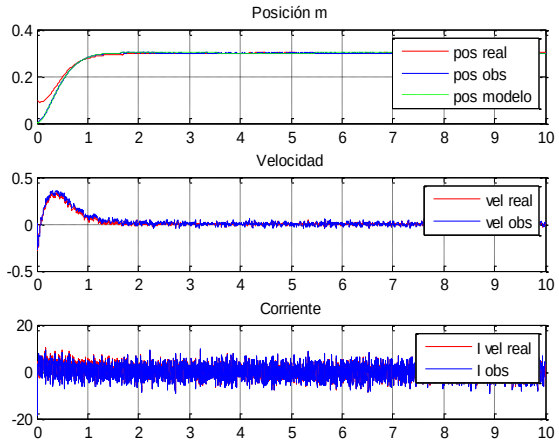
Ruido_planta-> media = 0 , varianza = 0.02

Fricción seca = 0.4*10 (Ruido de planta)

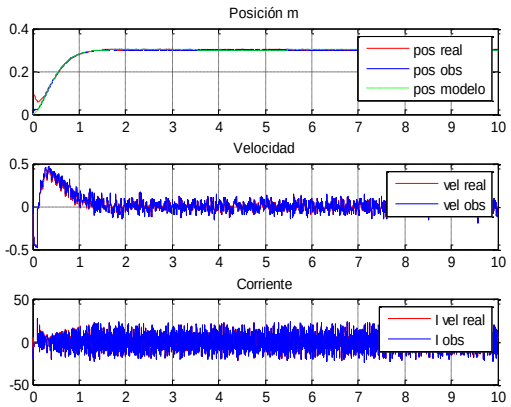
- Peso q1o = 1 , q2o = 0 , q3o = 0



- Peso q1o = 10 , q2o = 10 , q3o = 10



- Peso q1o = 100 , q2o = 100 , q3o = 100

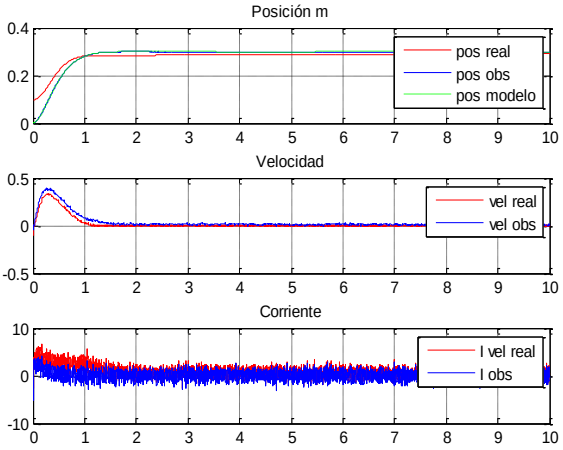


Condiciones:

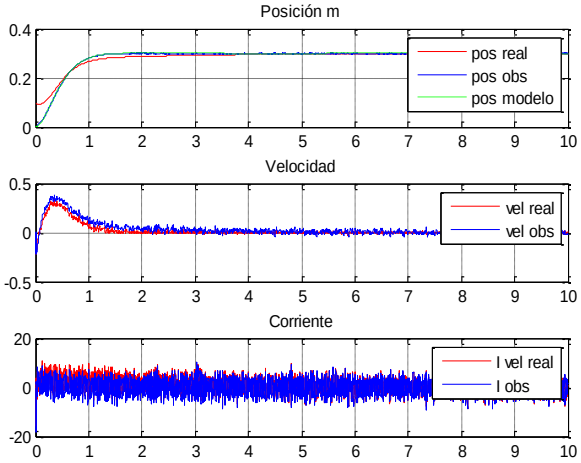
Ruido_planta-> media = 0 , varianza = 0.02

Fricción seca = 1*10 (Ruido de planta)

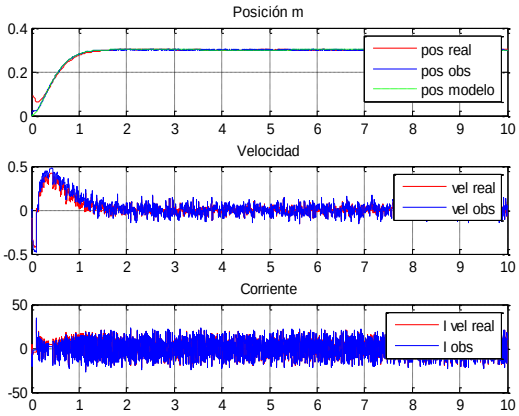
- Peso q1o = 1 , q2o = 0 , q3o = 0



- Peso q1o = 10 , q2o = 10 , q3o = 10



- Peso q1o = 100 , q2o = 100 , q3o = 100



OBSERVADOR FILTRO DE KALMAN

$$So = are(A', C^T Ro^{-1} C, WQoW')$$

Ganancia del observador:

$$L = S_o\, C^T Ro^{-1}$$

Discretizando las matrices para el observador

Con tiempo de muestreo dt.

$$[Aok\ , Bok] = c2d(A - LC, B, dt)$$

$$[Aok\ , Lok] = c2d(A - LC, L, dt)$$

$$[Aok\ , Wok] = c2d(A - LC, W, dt)$$

Si se toma a la fricción como ruido de planta, su valor medio es constante en cada sentido de movimiento.

$$[Aok\ , Wfok] = c2d(A - LC, Wf, dt)$$

Se considera que la salida del sistema es x, osea solo se puede medir (hay sensor) la posición, sin embargo ésta salida también posee ruido:

$$y = x + ruido_{sensor}$$

El observador que se obtiene:

$$Xo = Aok * Xo + Bok * u + Lok * y\ + Wfok * Fs$$

$$Xo = [0; 0; 0]$$

>>Script en Matlab

```
Q = input('Ingresar Q: ');

%% Observador
So = are(A',C'*inv(R)*C,W*Q*W');
L = So*C'*inv(R);

[Aok Bok] = c2d(A-L*C,B,dt);
[Aok Lok] = c2d(A-L*C,L,dt);
[Aok Wok] = c2d(A-L*C,W,dt);
[Aok Wfok] = c2d(A-L*C,Wf,dt);

xo = [ 0;  0;   0 ];
```

Simulación:

```
umax = 24;
Pot = 200;
Fric = 0.01*10;
[Ak Bk] = c2d(A,B,dt);
[Ak Wk] = c2d(A,Wf,dt);
x   = [ -0.1;  0.1;  0.1 ];
xm = [ 0;  0 ];
k = 1;
for tt = ti:dt:tf
    y = C*x + n(k,1);
    yy(k,1) = y;
    xm1(k,1) = xm(1,1);
    xm2(k,1) = xm(2,1);
    xx1(k,1) = x(1,1);  xx2(k,1) = x(2,1);   xx3(k,1) = x(3,1);
    xxo1(k,1) = xo(1,1);xxo2(k,1) = xo(2,1); xxo3(k,1) = xo(3,1);
    time(k,1) = tt;
    u = -Kfb*xo - Kff*xm;
    if(u > umax)
       u = umax;
    elseif(u < -umax)
       u = -umax;
    end
    if(x(2,1) >= 0)
        F = Fric;
    elseif(x(2,1) < 0)
        F = -Fric;
    end
    uc(k,1) = u;
    pot(k,1) = u*x(3,1);
    x = Ak*x + Bk*u + Wk*F;
    xm = Amk*xm + Bmk*rast(k,1);

    xo = Aok*xo + Bok*u + Lok*y + Wok*wmean;
    k = k + 1;
end
```

***Pruebas**

Condiciones:

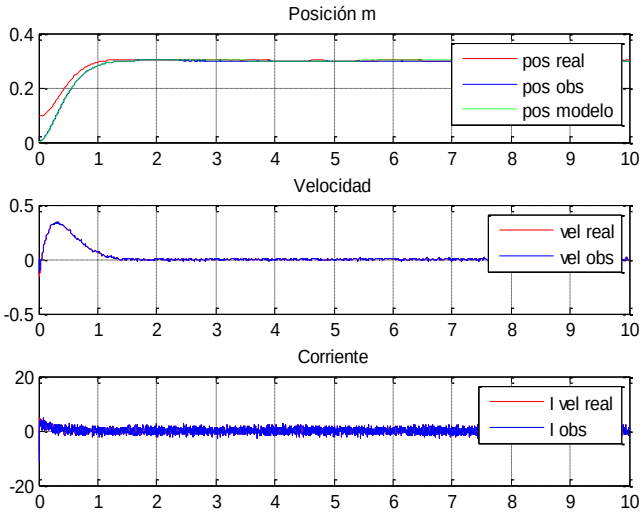
Ruido_sensor -> media = 0 , varianza = 0.01

Fricción seca = 0*10 (Ruido de planta)

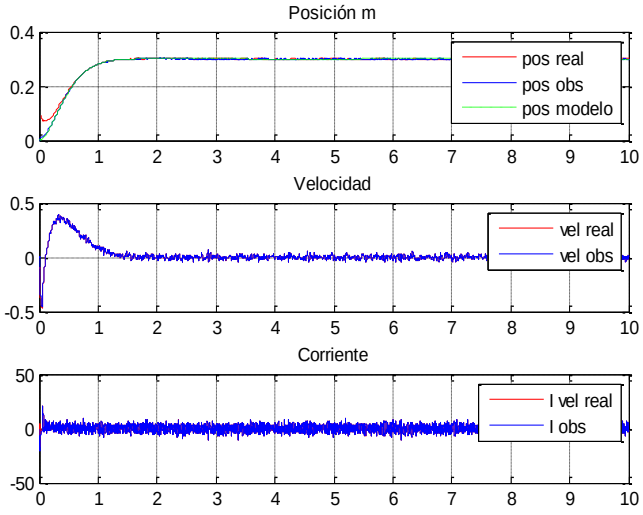
Referencia : x = 0.3

Variando pesos del observador

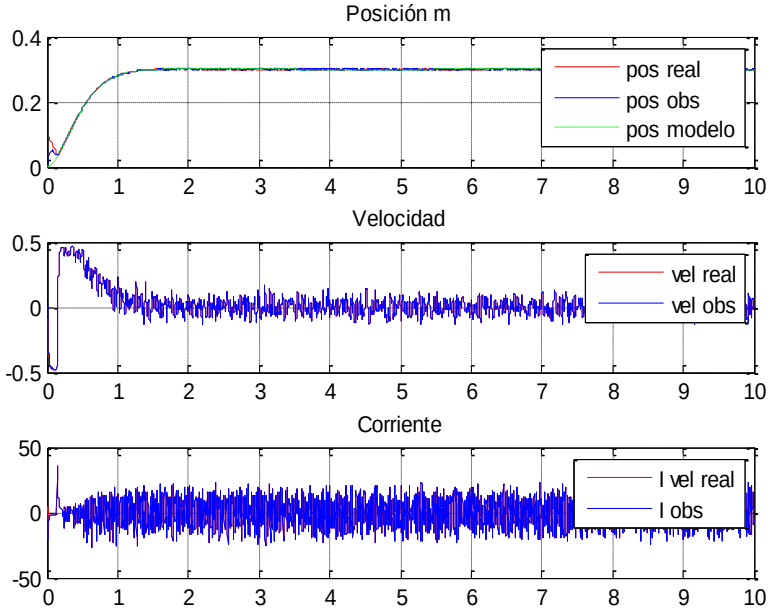
- Peso Q = 0.01



- Peso Q = 0.1



- Peso Q = 1

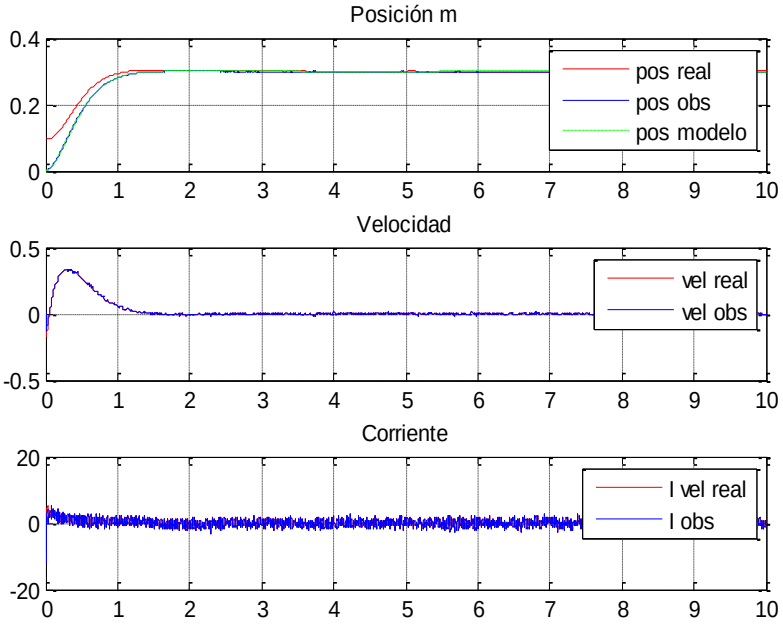


Condiciones:

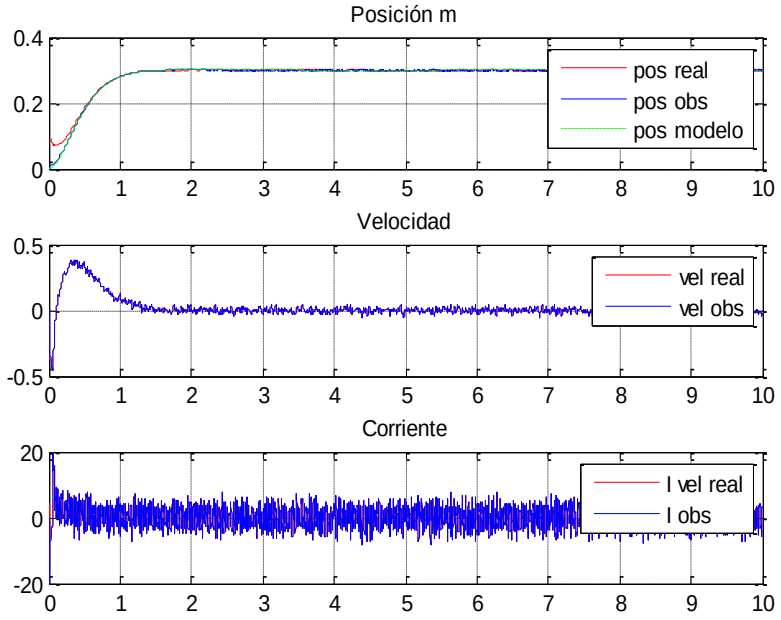
Ruido_planta-> media = 0 , varianza = 0.01

Fricción seca = 0.2*10 (Ruido de planta)

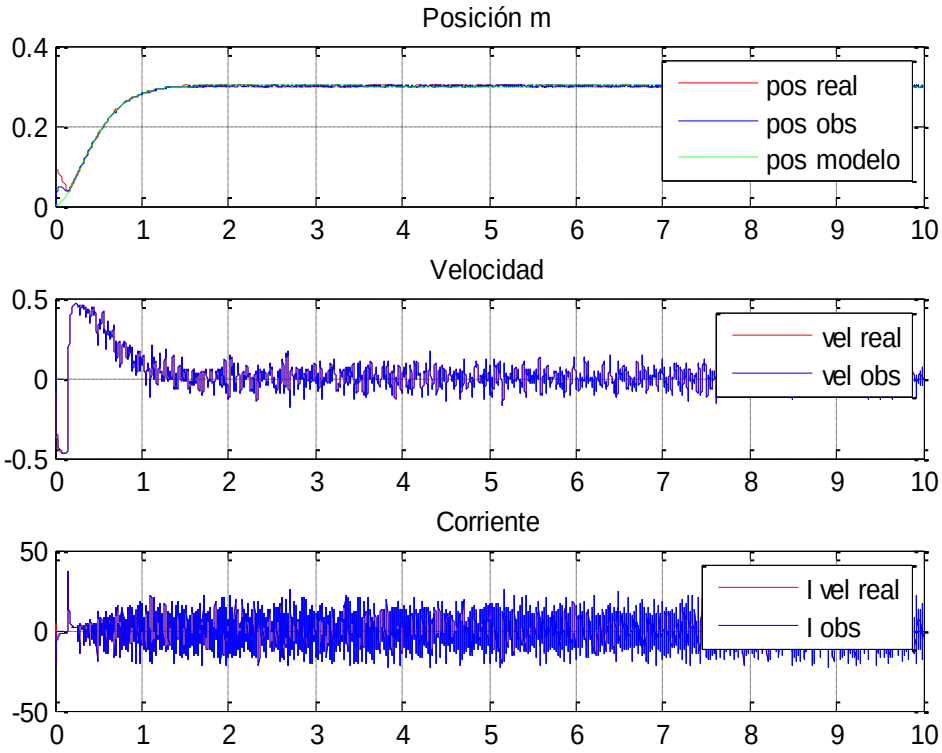
- Peso Q = 0.01



- Peso Q = 0.1



- Peso Q = 1

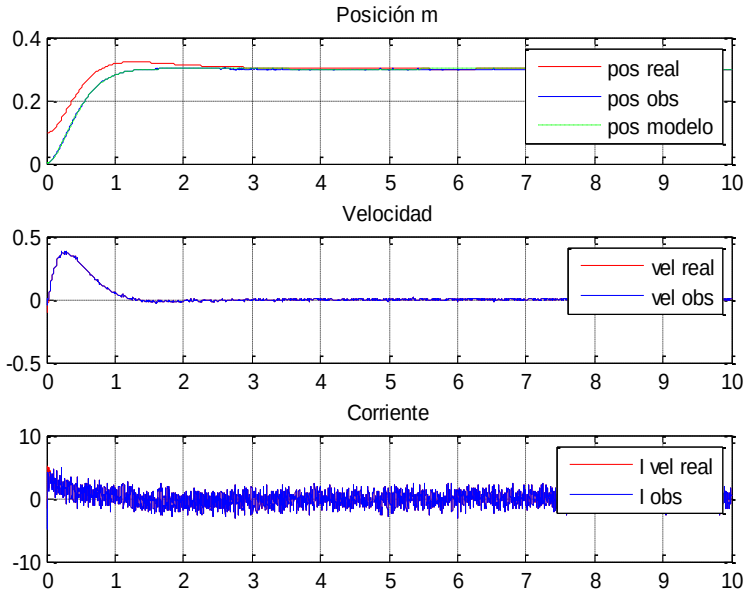


Condiciones:

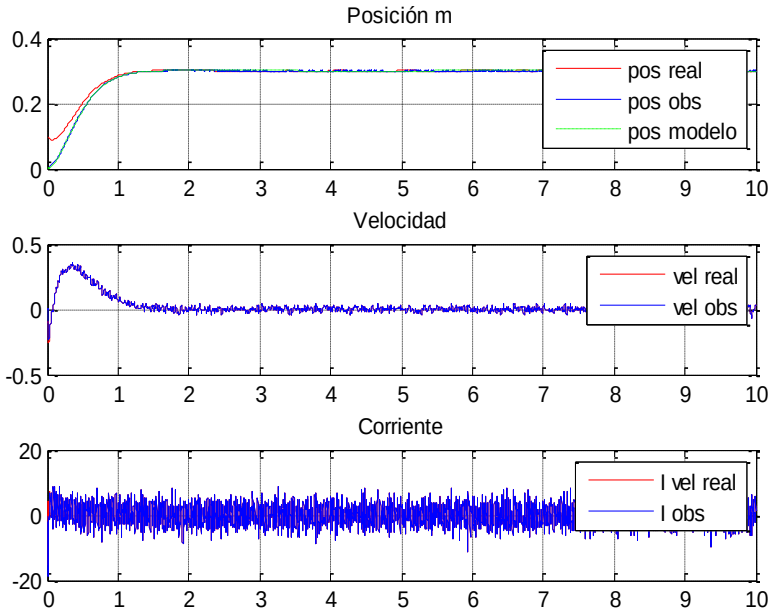
Ruido_planta-> media = 0 , varianza = 0.02

Fricción seca = 0.2*10 (Ruido de planta)

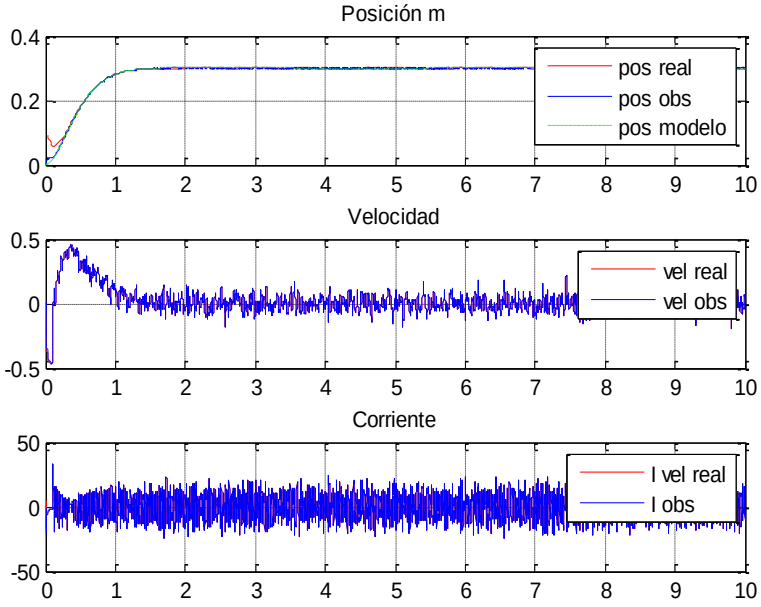
- Peso Q = 0.01



- Peso Q = 0.1



- Peso Q = 1

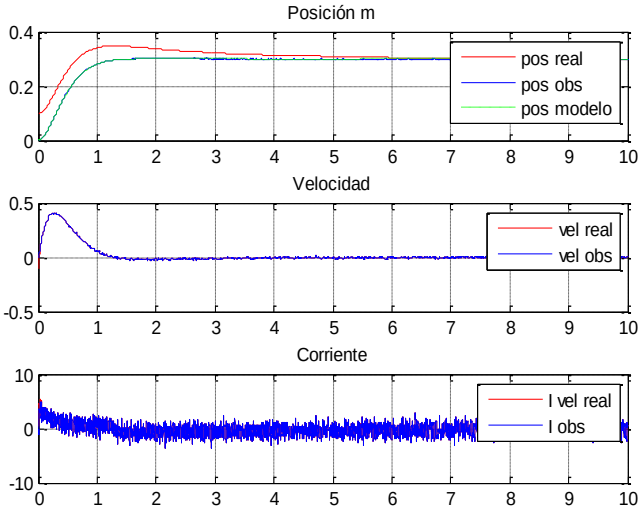


Condiciones:

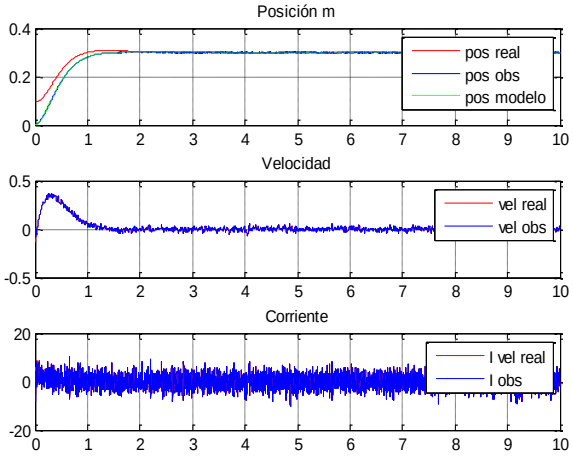
Ruido_planta-> media = 0 , varianza = 0.04

Fricción seca = 0.2*10 (Ruido de planta)

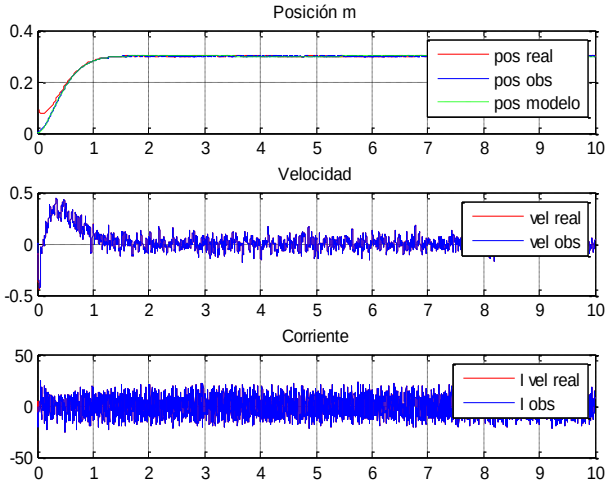
- Peso Q = 0.01



- Peso Q = 0.1



- Peso Q = 1

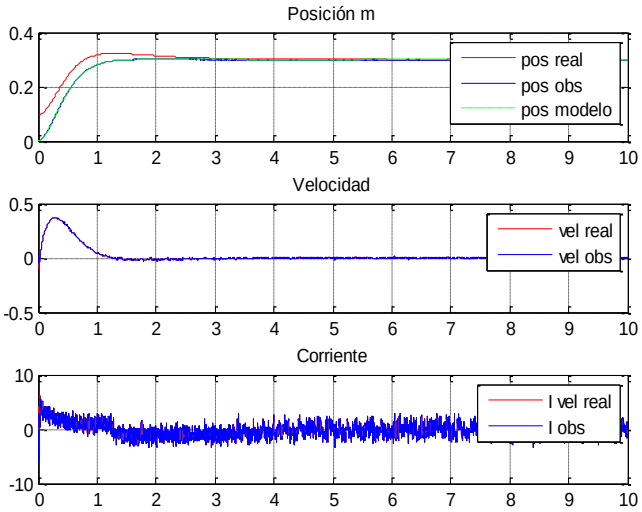


Condiciones:

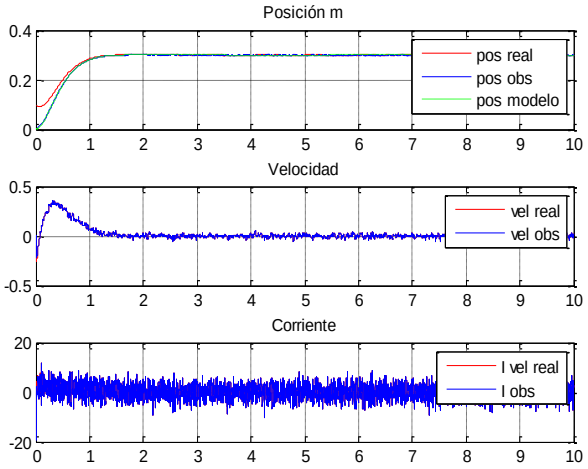
Ruido_planta-> media = 0 , varianza = 0.02

Fricción seca = 0.4*10 (Ruido de planta)

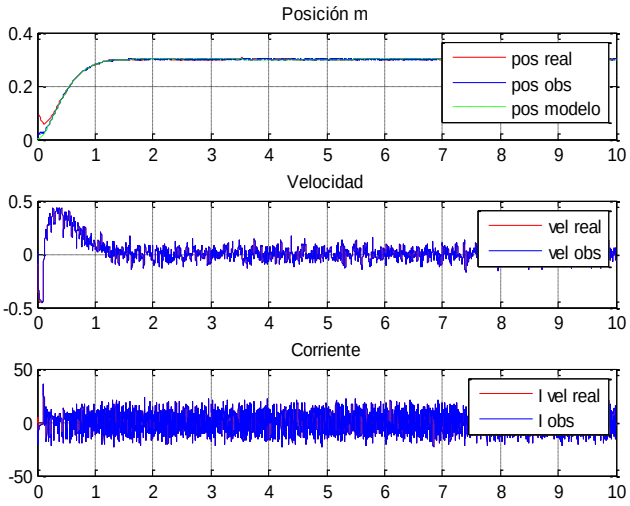
- Peso Q = 0.01



- Peso Q = 0.1



- Peso Q = 1

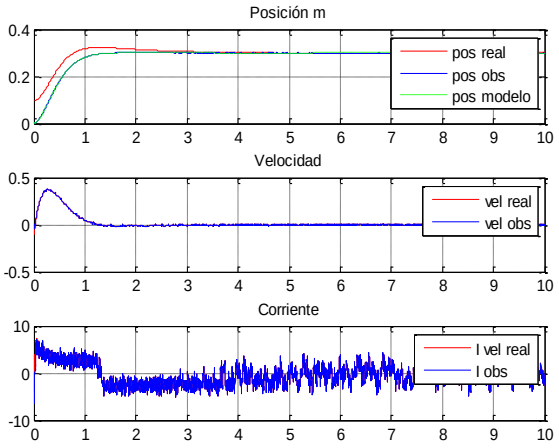


Condiciones:

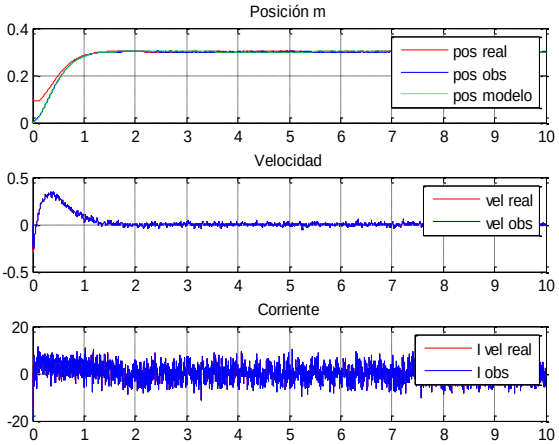
Ruido_planta-> media = 0 , varianza = 0.02

Fricción seca = 1*10 (Ruido de planta)

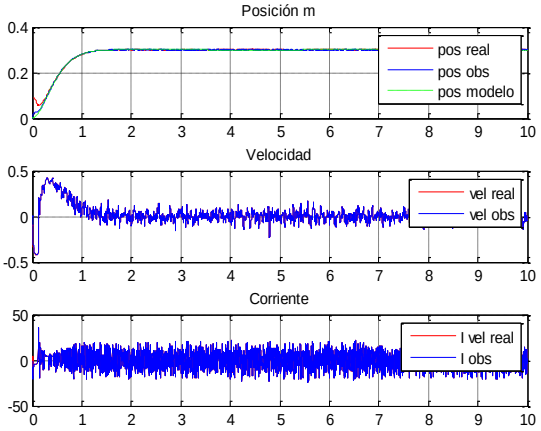
- Peso Q = 0.01



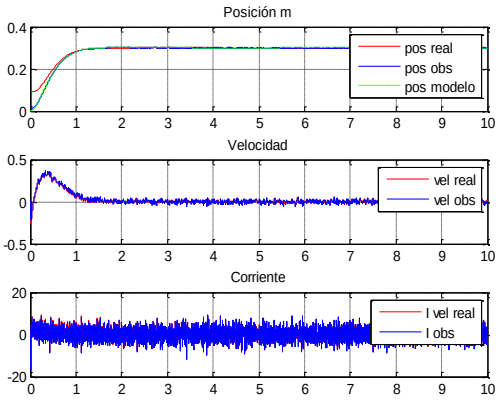
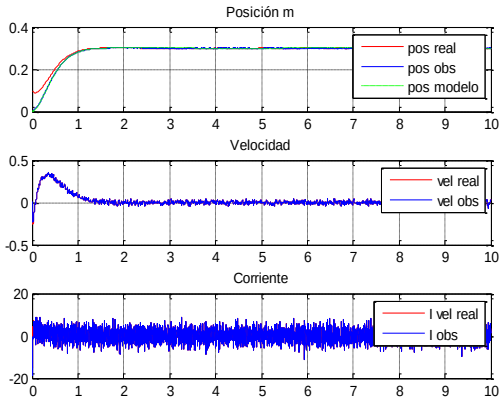
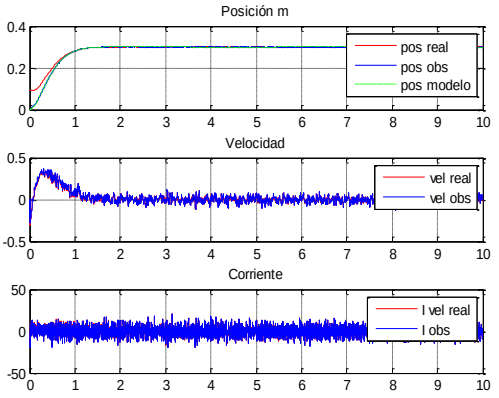
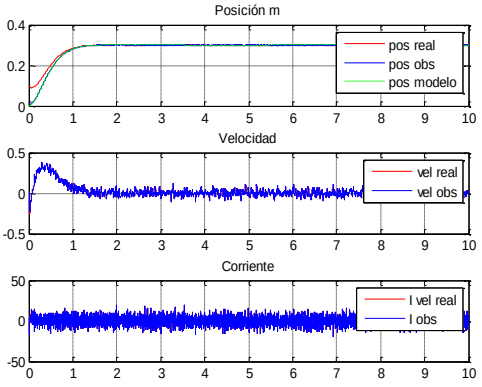
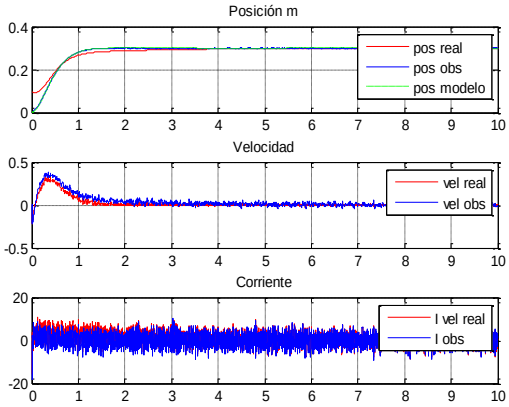
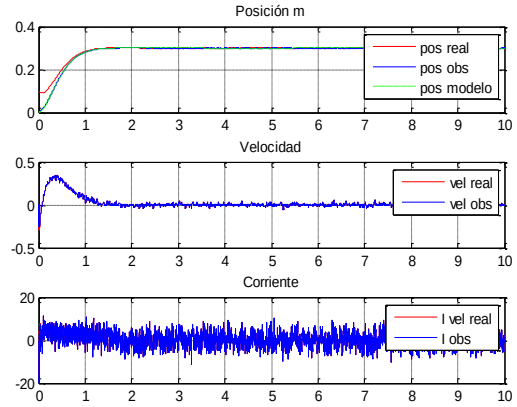
- Peso Q = 0.1



- Peso Q = 1



RESUMEN COMPARATIVO:

OPTIMAL	FILTRO DE KALMAN
q1o = 10 , q2o = 10 , q3o = 10	Q=0.1
Ruido_planta-> media = 0 , varianza = 0.02	**Fricción seca = 0.2*10**
	
Ruido_planta-> media = 0 , varianza = 0.04	**Fricción seca = 0.2*10**
	
Ruido_planta-> media = 0 , varianza = 0.02	**Fricción seca = 1*10**
	

Se puede apreciar que cuando el ruido de sensor se incrementa de varianza 0.02 a 0.04 el filtro de Kalman estima sólo muy poco mejor y reduce levemente el ruido en los estados observados. Pero si se incrementa el ruido en planta (en éste caso la fricción seca) el control utilizando los estados observados por el Filtro de Kalman es mucho mejor que el usado con el observador Optimal.

CONCLUSIONES:

1. Para el diseño del observador con la estructura de Feedforward + Feedback se utilizó solo la planta y la acción de control “u” generada con dicha estructura de control.

2. Tanto el observador Optimal como el filtro de Kalman estiman los estados correctamente al utilizar la estructura de Feedforward + Feedback.

3. Se apreció que los estados observados por los dos tipos de observadores resultan un poco más ruidosos con la estructura de Feedforward + Feedback, que respecto a la estructura de solo Feedback (realizada en el informe anterior).

4. Al incrementar el ruido en el sensor el filtro Kalman estima levemente mejor que al utilizar el observador Optimal.

5. Al incrementar el ruido en la planta el filtro de Kalman estima mucho mejor que el observador Optimal, haciendo así que la respuesta del controlador sea menos ruidosa y más rápida.

2. Probar el algoritmo de sensor fusion basado en el filtro de Kalman, variando la cantidad de sensores ruidosos.

Integración de sensores usando filtro de Kalman

El proceso se modela de la siguiente manera:

Sensado de temperatura T, utilizando "n" sensores:

$$\dot{T} = [0]T + [1]w$$

$$y = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} T + \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_n \end{bmatrix}$$

Dónde:

w: ruido del proceso

$\eta_1, \eta_2, \dots, \eta_n : ruido\ de\ los\ sensores$

Sensores de temperatura:

Sensor	ruido	varianza
Sensor 1	η_1	$\boldsymbol{\sigma_1}$
Sensor 2	η_2	$\boldsymbol{\sigma_2}$
……	….	….
Sensor n	η_n	$\boldsymbol{\sigma_n}$

El algoritmo de estimación:

Utilizando el filtro de Kalman, sólo utilizando las señales medidas por los "n" sensores podemos estimar la temperatura con menor error.

$$\hat{\dot{T}} = AT + L(y - C\hat{T})$$

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{\sigma_1}^2 & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\sigma_2}^2 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\sigma_3}^2 & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{\sigma_n}^2 \end{bmatrix}$$

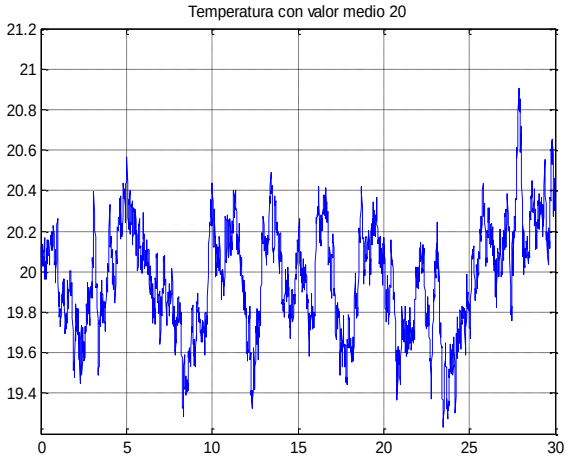
$$\boldsymbol{S} = \boldsymbol{are}(A^T, C^T R^{-1} C, Q)$$

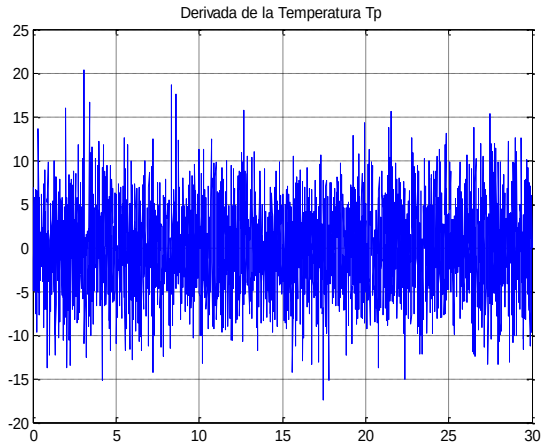
$$L = SC^T R^{-1}$$

>>Script en Matlab

Generación del ruido en la variable de temperatura.

```
ti = 0;   dt = 0.01;   tf = 30;
t = ti:dt:tf;   t = t';
nt = length(t);
% Generacion de ruido (filtrado) de variacion de T
sigmarT = 0.5;
rT = sigmarT*randn(nt,1);
a = 1;   % Filtro pasa bajoe
rfilt = 0;
k = 1;
for tt = 1:nt
   rf(k,1) = rfilt;
   rfilt = (1-a*dt)*rfilt + a*rT(k,1)*dt;
   k = k + 1;
end

rf = 10*rf;
figure(1);
plot(t,rT,t,rf);   grid;
title('Ruido filtrado con valor medio cero');

rf = 10*rf;
figure(1);
plot(t,rT,t,rf);   grid;
title('Ruido filtrado con valor medio cero');

x = 20;    % Temperatura media
T = x + rf;
figure(2);
plot(t,T); grid;
title('Temperatura con valor medio 20');

T(nt+1,1) = T(nt);    % Punto adicional para que Tp tenga nt puntos
Tp = diff(T)/dt;
figure(3);
plot(t,Tp);   grid;
title('Derivada de la Temperatura Tp');
w = Tp-mean(Tp);
sigmaw = std(Tp);% Desviacion estandar de la derivada de T
```





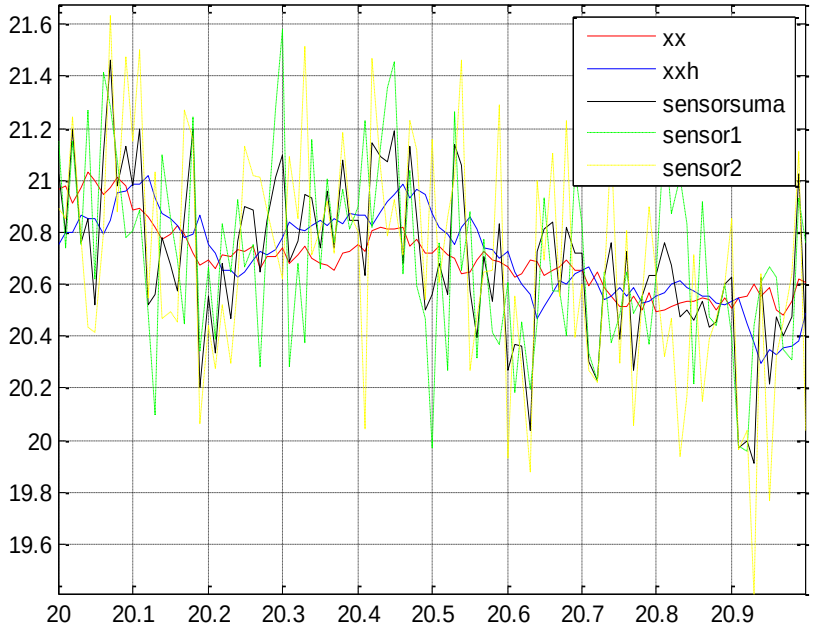
Algoritmo para 2 sensores

```
A = [ 0 ];
C = [ 1; 1];
sigmaw = sigmaw;
Q = [ sigmaw*sigmaw ];
sensigma1 = 0.4;
sensigma2 = 0.4;
R = diag([sensigma1^2  sensigma2^2]);
S = are(A',C'*inv(R)*C,Q);      % Riccati Equation
L = S*C'*inv(R);

w = w;     % Tp generado anteriormente
ruido1 = sensigma1*randn(nt,1);
ruido2 = sensigma2*randn(nt,1);

xh = x + (ruido1(1,1) + ruido2(1,1))/2;

k = 1;
for tt = ti:dt:tf
    xx(k,1) = x;
    xxh(k,1) = xh;
    y = [ x + ruido1(k,1)
            x + ruido2(k,1)];
    sensor1(k,1) = y(1,1);
    sensor2(k,1) = y(2,1);
    sensorsuma(k,1) = (y(1,1)+y(2,1))/2;
    xhp = A*xh + L*(y - C*xh);
    xh = xh + xhp*dt;
    xp = A*x + w(k,1);
    x = x + xp*dt;
    k = k + 1;
end


figure(4);
plot(t,xx,'-r',t,xxh,'-b',t,sensorsuma,'-k',t,sensor1,'--g',t,sensor2,'--y');
grid;
title('Comparación de Temperaturas');

% Error de estimacion
ekalman = xxh-xx;
esuma = sensorsuma - xx;
esensor1 = sensor1 - xx;
figure(4);
plot(t,ekalman,'-b',t,esuma,'--k',t,esensor1,'--g');
grid;
title('Comparación de Errores de Estimación');

stdkalman = std(ekalman);
stdsuma = std(esuma);
stdsensor1 = std(esensor1);
disp(['STD error Kalman :  ',num2str(stdkalman)]);
disp(['STD error Suma    :  ',num2str(stdsuma)]);
disp(['STD error Sensor1:  ',num2str(stdsensor1)]);
```

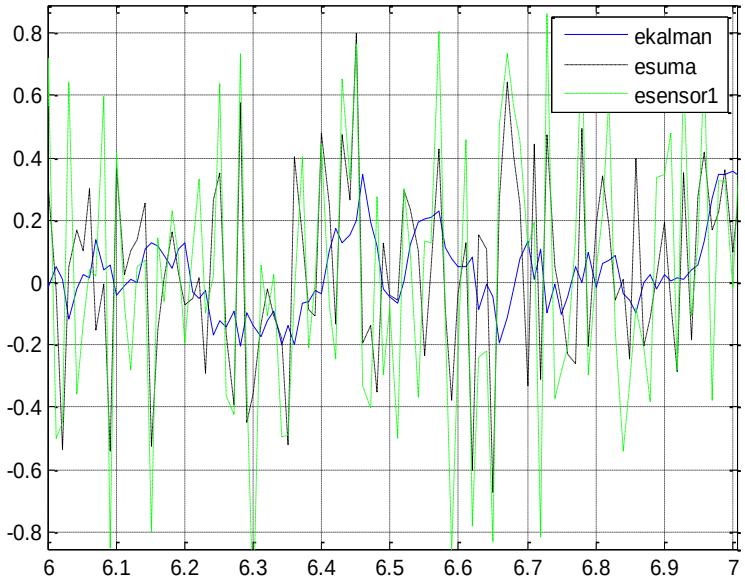
Pruebas:

Sensor	media	varianza
Sensor 1	0	$\mathbf{0.4}$
Sensor 2	0	$\mathbf{0.4}$

Comparación de temperaturas:



Comparación de errores de estimación:



Comparamos las desviaciones estándar de los errores de estimación

Estimación	Kalman	Promedio	Sensor1
STD	0.12746	0.28743	0.40257

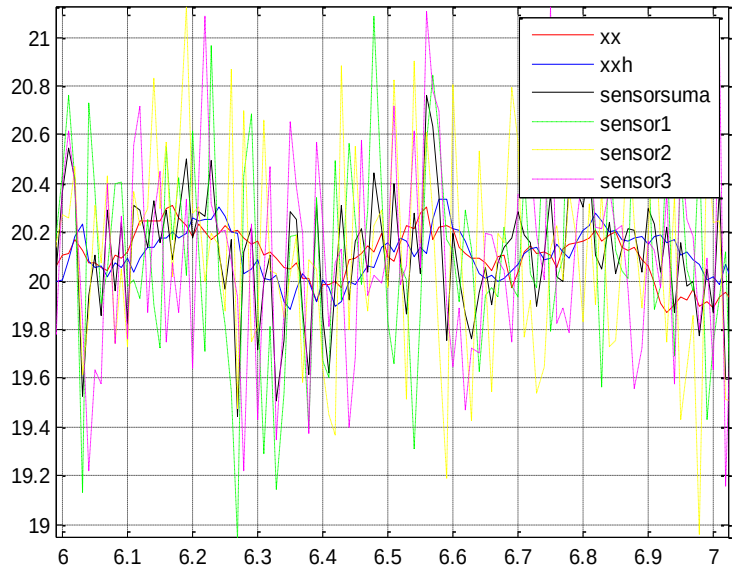
Algoritmo para 3 sensores

```
A = [ 0 ];
C = [ 1; 1; 1];
sigmaw = sigmaw;
Q = [ sigmaw*sigmaw ];
sensigma1 = 0.4;
sensigma2 = 0.4;
sensigma3 = 0.4;
R = diag([sensigma1^2  sensigma2^2  sensigma3^2 ]);
S = are(A',C'*inv(R)*C,Q);      % Riccati Equation
L = S*C'*inv(R);

w = w;     % Tp generado anteriormente
ruido1 = sensigma1*randn(nt,1);
ruido2 = sensigma2*randn(nt,1);
ruido3 = sensigma3*randn(nt,1);

xh = x + (ruido1(1,1) + ruido2(1,1) + ruido3(1,1))/3;

k = 1;
for tt = ti:dt:tf
    xx(k,1) = x;
    xxh(k,1) = xh;
    y = [ x + ruido1(k,1)
            x + ruido2(k,1)
            x + ruido3(k,1) ];
    sensor1(k,1) = y(1,1);
    sensor2(k,1) = y(2,1);
    sensor3(k,1) = y(3,1);
    sensorsuma(k,1) = (y(1,1)+y(2,1)+y(3,1))/3;
    xhp = A*xh + L*(y - C*xh);
    xh = xh + xhp*dt;
    xp = A*x + w(k,1);
    x = x + xp*dt;
    k = k + 1;
end

figure(4);
plot(t,xx,'-r',t,xxh,'-b',t,sensorsuma,'-k',t,sensor1,'--g',t,sensor2,'--y',
t,sensor3,'--m');
grid;
title('Comparación de Temperaturas');

% Error de estimacion
ekalman = xxh-xx;
esuma = sensorsuma - xx;
esensor1 = sensor1 - xx;
figure(4);
plot(t,ekalman,'-b',t,esuma,'--k',t,esensor1,'--g');
grid;
title('Comparación de Errores de Estimación');

stdkalman = std(ekalman);
stdsuma = std(esuma);
stdsensor1 = std(esensor1);
disp(['STD error Kalman :  ',num2str(stdkalman)]);
disp(['STD error Suma    :  ',num2str(stdsuma)]);
disp(['STD error Sensor1:  ',num2str(stdsensor1)]);
```

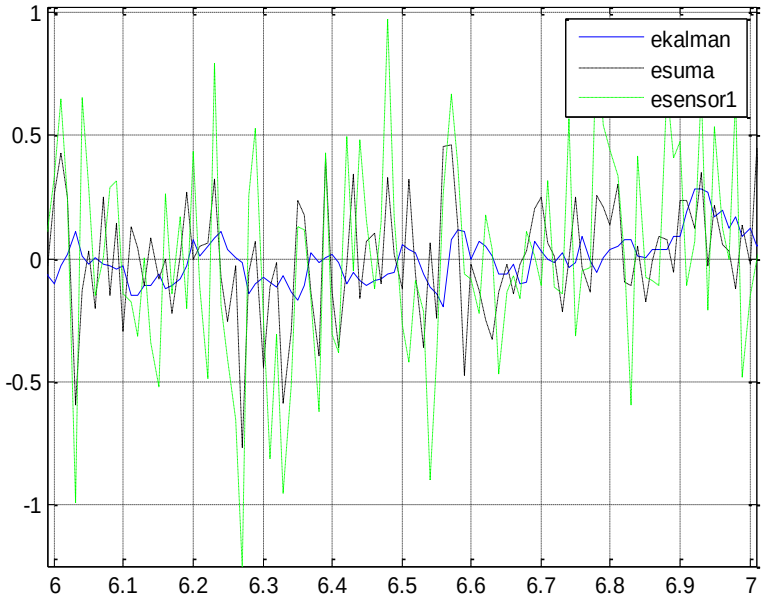
Pruebas:

Sensor	media	varianza
Sensor 1	0	$\mathbf{0.4}$
Sensor 2	0	$\mathbf{0.4}$
Sensor 3	0	$\mathbf{0.4}$

Comparación de temperaturas:



Comparación de errores de estimación:



Comparamos las desviaciones estándar de los errores de estimación

Estimación	Kalman	Promedio	Sensor1
STD	0.11451	0.24036	0.40756

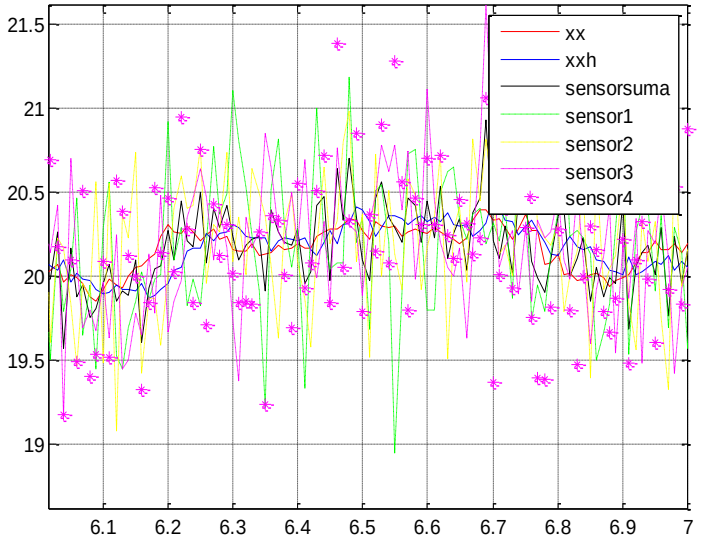
Algoritmo para 4 sensores

```
A = [ 0 ];
C = [ 1; 1; 1;1];
sigmaw = sigmaw;
Q = [ sigmaw*sigmaw ];
sensigma1 = 0.4;
sensigma2 = 0.4;
sensigma3 = 0.4;
sensigma4 = 0.4;
R = diag([sensigma1^2  sensigma2^2  sensigma3^2  sensigma4^2]);
S = are(A',C'*inv(R)*C,Q);      % Riccati Equation
L = S*C'*inv(R);

w = w;     % Tp generado anteriormente
ruido1 = sensigma1*randn(nt,1);
ruido2 = sensigma2*randn(nt,1);
ruido3 = sensigma3*randn(nt,1);
ruido4 = sensigma4*randn(nt,1);
xh = x + (ruido1(1,1) + ruido2(1,1) + ruido3(1,1) + ruido4(1,1))/4;

k = 1;
for tt = ti:dt:tf
    xx(k,1) = x;
    xxh(k,1) = xh;
    y = [ x + ruido1(k,1)
            x + ruido2(k,1)
            x + ruido3(k,1)
            x + ruido4(k,1)];
    sensor1(k,1) = y(1,1);
    sensor2(k,1) = y(2,1);
    sensor3(k,1) = y(3,1);
    sensor4(k,1) = y(4,1);
    sensorsuma(k,1) = (y(1,1)+y(2,1)+y(3,1) +y(4,1))/4;
    xhp = A*xh + L*(y - C*xh);
    xh = xh + xhp*dt;
    xp = A*x + w(k,1);
    x = x + xp*dt;
    k = k + 1;
end

figure(4);
plot(t,xx,'-r',t,xxh,'-b',t,sensorsuma,'-k',t,sensor1,'--g',t,sensor2,'--
y',t,sensor3,'--m',t,sensor4,'ob');
grid;
title('Comparación de Temperaturas');

% Error de estimacion
ekalman = xxh-xx;
esuma = sensorsuma - xx;
esensor1 = sensor1 - xx;
figure(5);
plot(t,ekalman,'-b',t,esuma,'--k',t,esensor1,'--g');
grid;
title('Comparación de Errores de Estimación');

stdkalman = std(ekalman);
stdsuma = std(esuma);
stdsensor1 = std(esensor1);
disp(['STD error Kalman :  ',num2str(stdkalman)]);
disp(['STD error Suma    :  ',num2str(stdsuma)]);
disp(['STD error Sensor1:  ',num2str(stdsensor1)]);
```

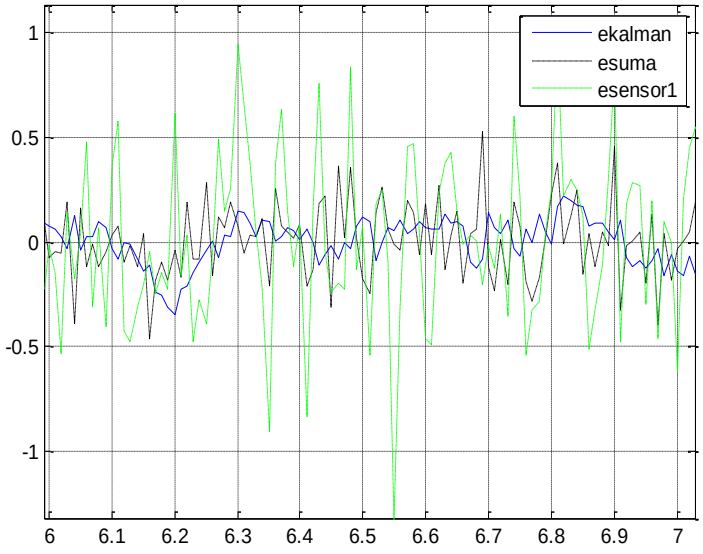
Pruebas:

Sensor	media	varianza
Sensor 1	0	$\mathbf{0.4}$
Sensor 2	0	$\mathbf{0.4}$
Sensor 3	0	$\mathbf{0.4}$
Sensor 4	0	$\mathbf{0.4}$

Comparación de temperaturas:



Comparación de errores de estimación:



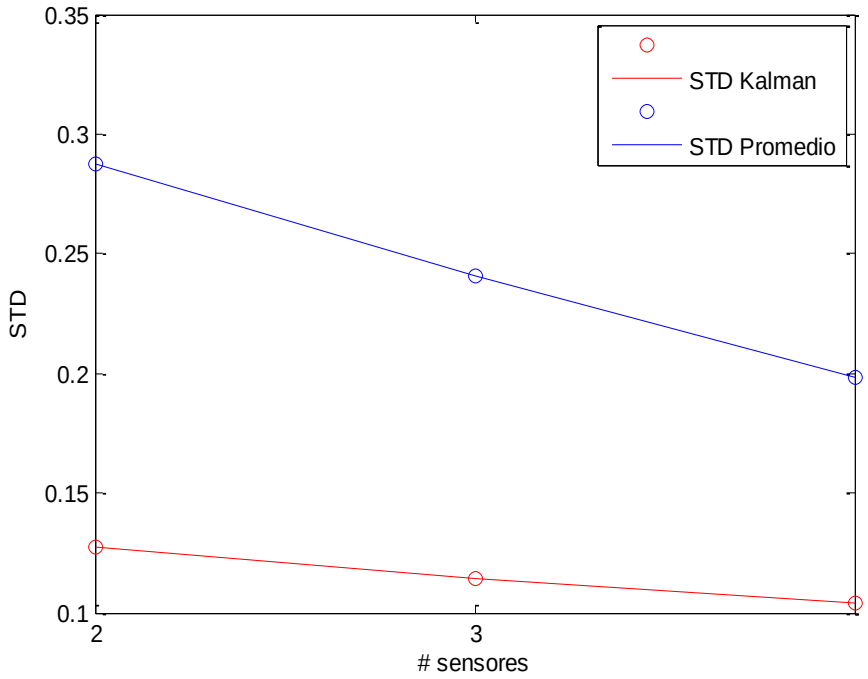
Comparamos las desviaciones estándar de los errores de estimación

Estimación	Kalman	Promedio	Sensor1
STD	0.10413	0.198	0.40848

Resumen de comparación de mejora en la estimación aumentando sensores:

Con fines de comparar, se utilizó la misma media (0) desviación estándar en cada sensor (0.4), con ello se obtuvieron los siguientes resultados:

Estimación	STD Kalman	STD Promedio	STD Sensor1
2 sensores	0.12746	0.28743	0.40257
3 sensores	0.11451	0.24036	0.40756
4 sensores	0.10413	0.198	0.40848



3. Probar el algoritmo de sensor fusion en la integración de encoder con acelerómetro.

Fusión del encoder + acelerómetro:

Seguir la posición: (f=0.5)

$$x(t) = 3\sin(2\pi f t + \varphi)$$

>Derivando:

La velocidad sería:

$$\dot{x}(t) = 6\pi f \cos(2\pi f t + \varphi)$$

La aceleración sería:

$$\ddot{x}(t) = -12(\pi f)^2 \sin(2\pi f t + \varphi)$$

Encoder:

P pulsos x vuelta

$$d\theta = \frac{2\pi}{P}$$

Acelerómetro:

$$\ddot{x} \; es \; la \; medida \; de \; la \; aceleración$$

$$x_{accel} = \ddot{x} + ruido$$

$$\dot{x} = \dot{x_o} + x_{accel} * dt$$

Modelo del sistema de sensores

$$\dot{X} = AX + Wa$$

$$y = CX$$

$$X = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \ddot{x}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \; ; \qquad W \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

El observador es:

>Sólo encoder:

$$\dot{\hat{X}} = (A - LC)\hat{X} + Ly$$

>Encoder + acelerómetro:

$$\dot{\hat{X}} = (A - LC)\hat{X} + Wa + Ly$$

Siendo y: medición del encoder

- Pesos:

$$Q = q$$

$$R = r$$

$$\boldsymbol{S} = \boldsymbol{are}(A^T, C^T R^{-1} C, Q)$$

$$L = SC^T R^{-1}$$

- Discretización:

$$\begin{gathered} Ao = A - LC \\ Wo = W \\ Lo = L \\ [Aok\ Wok] = c2d(Ao, Wo, dt) \\ [Aok\ Lok] = c2d(Ao, Lo, dt) \end{gathered}$$

>Script en Matlab:

Determinación del ángulo leído por el encoder:

```
k = 1;
x_old = 0;
for tt = ti:dt:tf
    xk = x(k,1);
    if(xk >= 0)
         x_enc(k,1) = dang*floor(xk/dang);
    elseif(xk < 0)
         x_enc(k,1) = dang*ceil(xk/dang);
    end
    xvel_enc(k,1) = (x_enc(k,1) - x_old)/dt;
    x_old = x_enc(k,1);
    k= k + 1;
end
```

Estimador:

```
A = [ 0 1
      0 0 ];
W = [ 0
      1 ];
C = [ 1 0 ];
% q y r se probarán como si fuesen pesos
q = input('Peso q [50]: ');
r = input('Peso r [1]: ');
Q = W*[q]*W';
R = [r];
S = are(A',C'*inv(R)*C,Q);
L = S*C'*inv(R);
Ao = A-L*C;
Wo = W;
Lo = L;
[Aok Wok] = c2d(Ao,Wo,dt);
[Aok Lok] = c2d(Ao,Lo,dt);
```

Simulación: (Se va a comparar Estimación con sólo encoder y encoder+acelerómetro)

```
y = x_enc;
xh = [ x_enc(1,1); 0 ];
xha = [ x_enc(1,1); 0 ];
k = 1;
acel_int = 0;
for tt = ti:dt:tf
    xxh(k,1) = xh(1,1);
    xxvelh(k,1) = xh(2,1);
    xxha(k,1) = xha(1,1);
    xxvelha(k,1) = xha(2,1);
    xh = Aok*xh + 0*Wok*xacelruido(k,1) + Lok*y(k,1);
    xha = Aok*xha + 1*Wok*xacelruido(k,1) + Lok*y(k,1);
    acel_int = acel_int + xacel(k,1)*dt;
    vel_acel(k,1) = acel_int;
    k = k + 1;
end
```

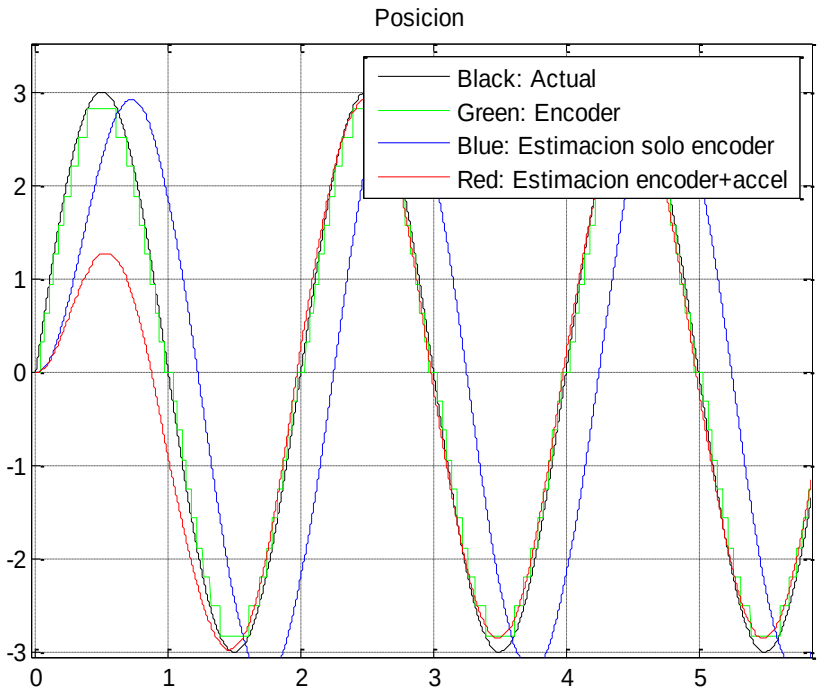
Ploteo:

```
figure(1);
plot(t,x,'-k',t,x_enc,'-g',t,xxh,'-b',t,xxha,'-r');
legend('Black: Actual','Green: Encoder','Blue: Estimacion
solo encoder','Red: Estimacion encoder+accel');
title('Posicion');
grid;
figure(2);
plot(t,xvel,'-k',t,xvel_enc,'-g',t,xxvelh,'-b',t,xxvelha,'-
r',t,vel_acel,'-y');
legend('Black: Actual','Green: Encoder','Blue: Estimacion
solo encoder','Red: Estimacion encoder+accel','Yellow:
Integra accel');
title('Velocidad');
grid;
```

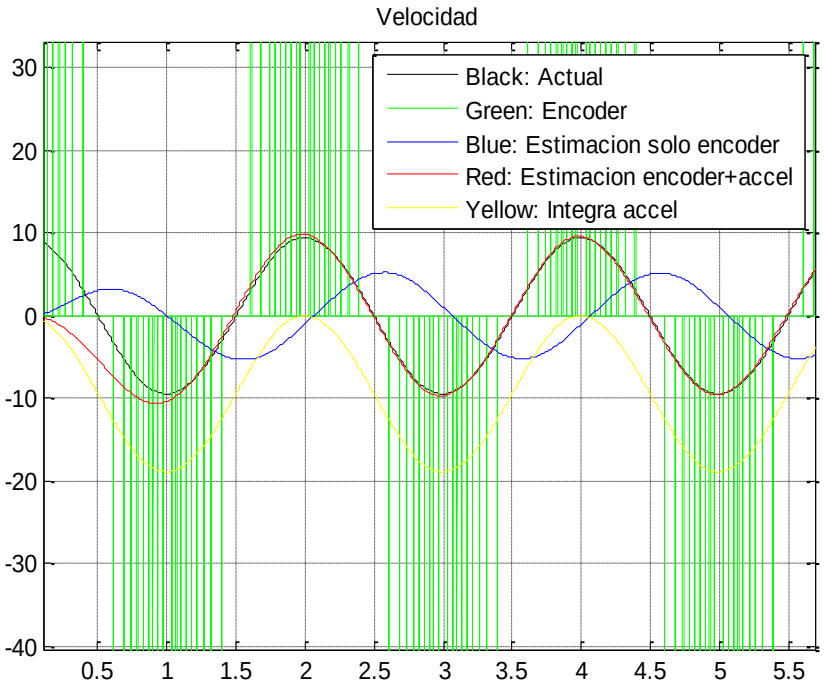
Pruebas:

- Variando los pulsos

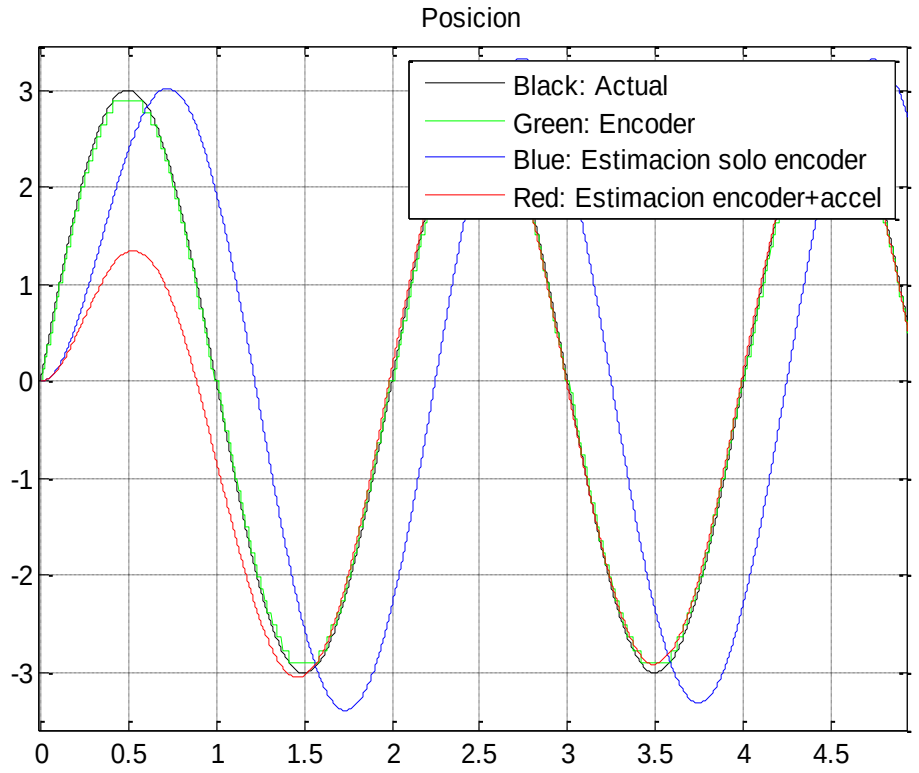
a. Considerando : q = 50 , r =1 , pulsos= 20 , $\varphi = 0$

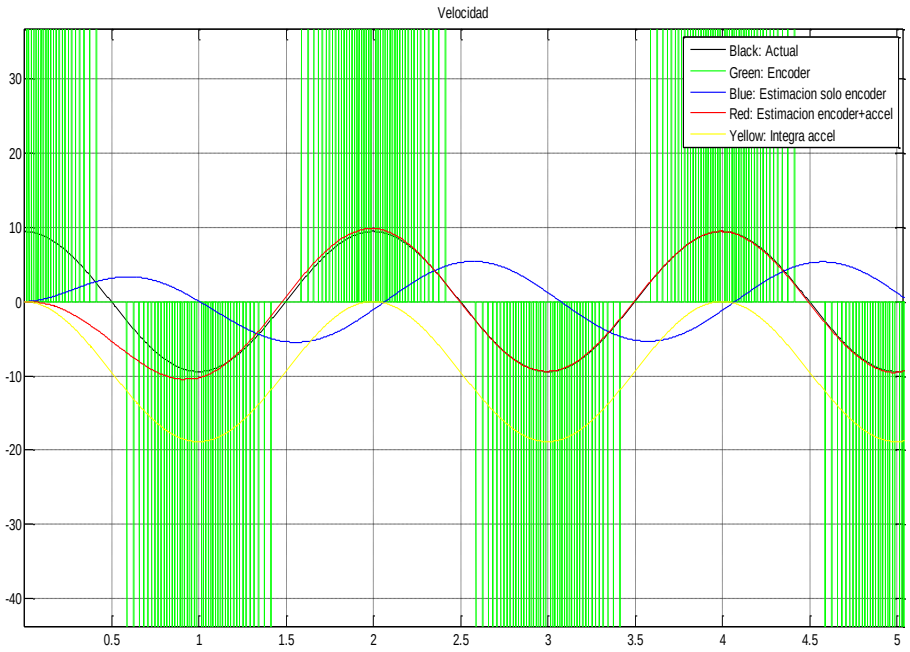


Se aprecia que en los picos y valles no se logra seguir muy bien.

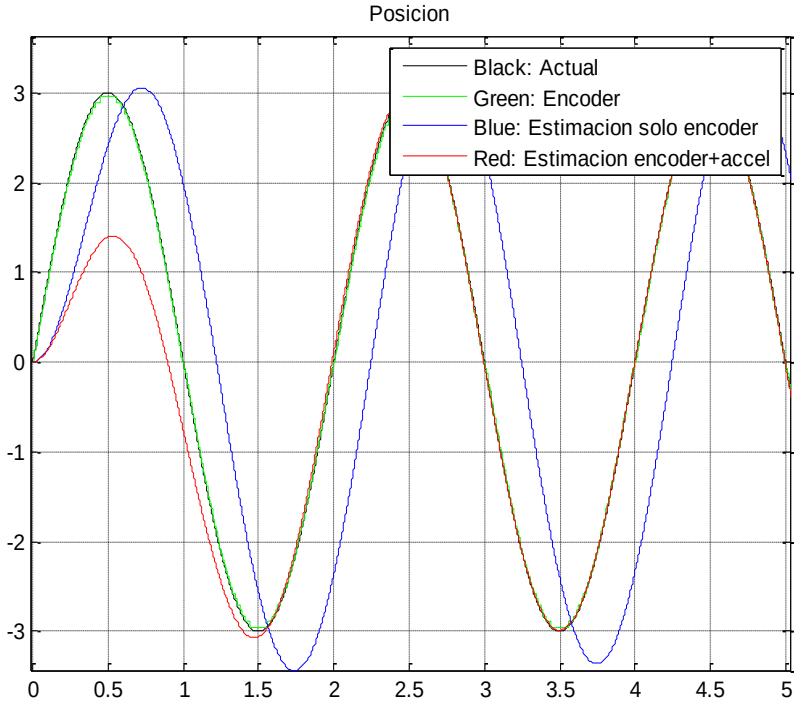


b. Considerando : q = 50 , r =1 , pulsos= 50 , $\varphi = 0$

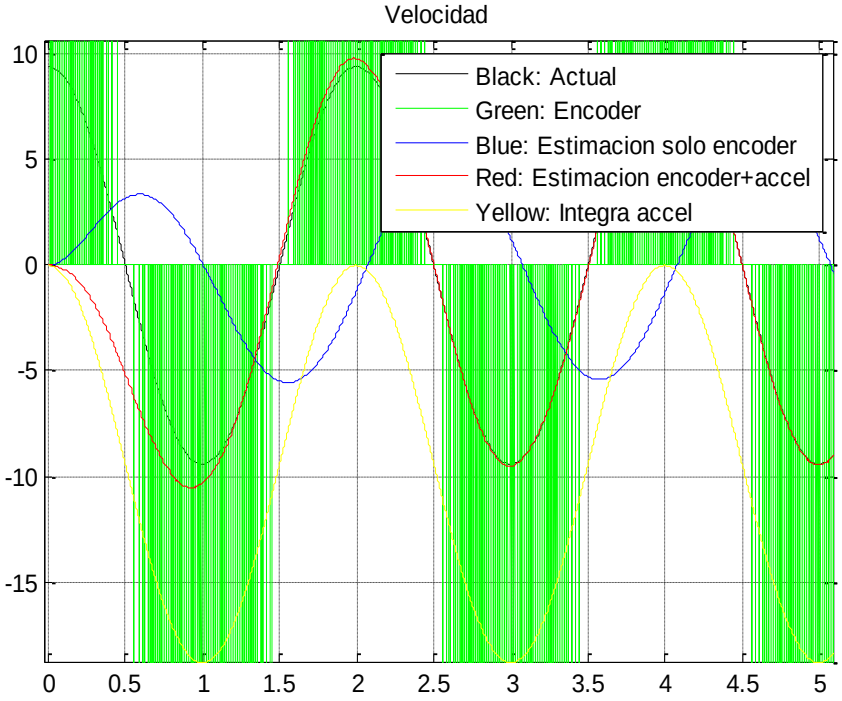




c. Considerando : q = 50 , r =1 , pulsos= 100 , $\varphi = 0$

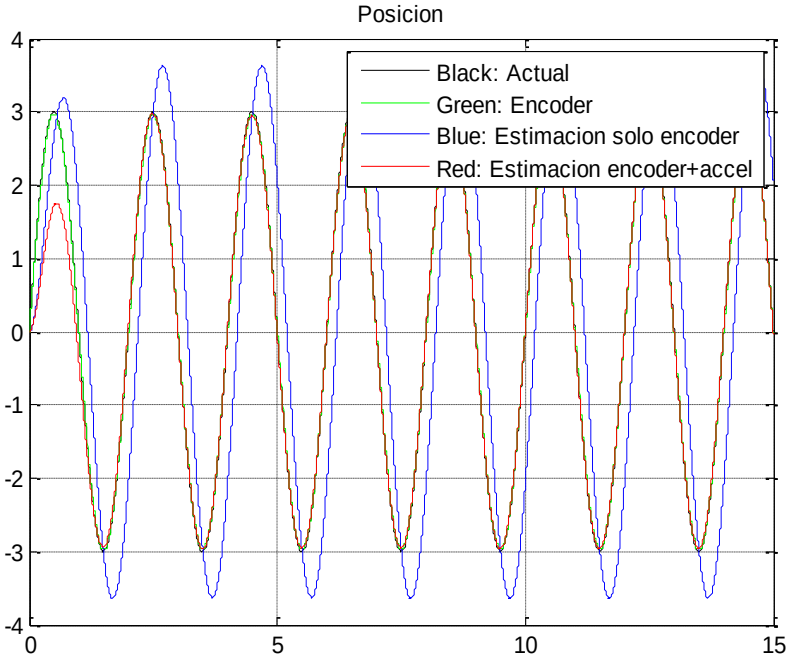


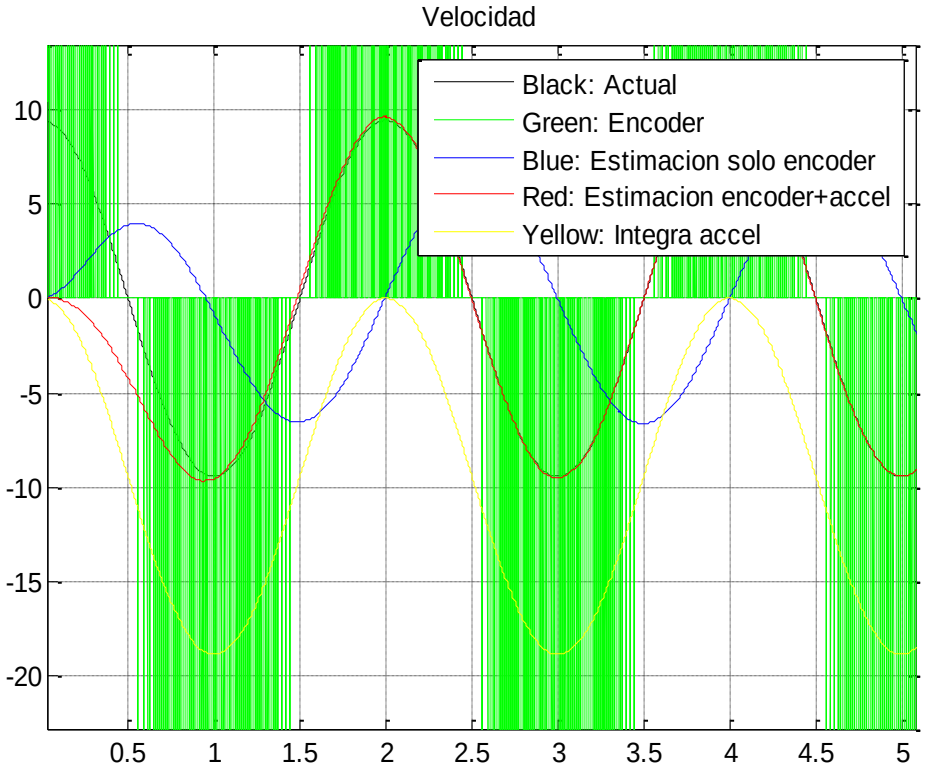
Se aprecia que en los picos y valles se mejoró el seguimiento de la estimación “encoder + acelerómetro” respecto al “valor actual”.



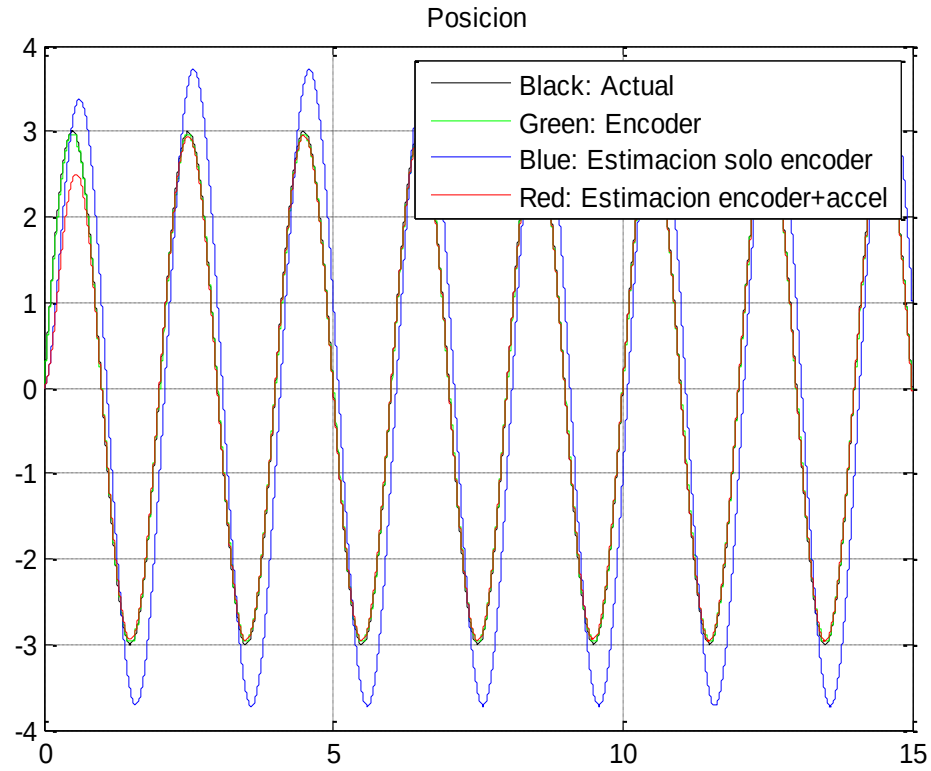
- Variando el peso “q”

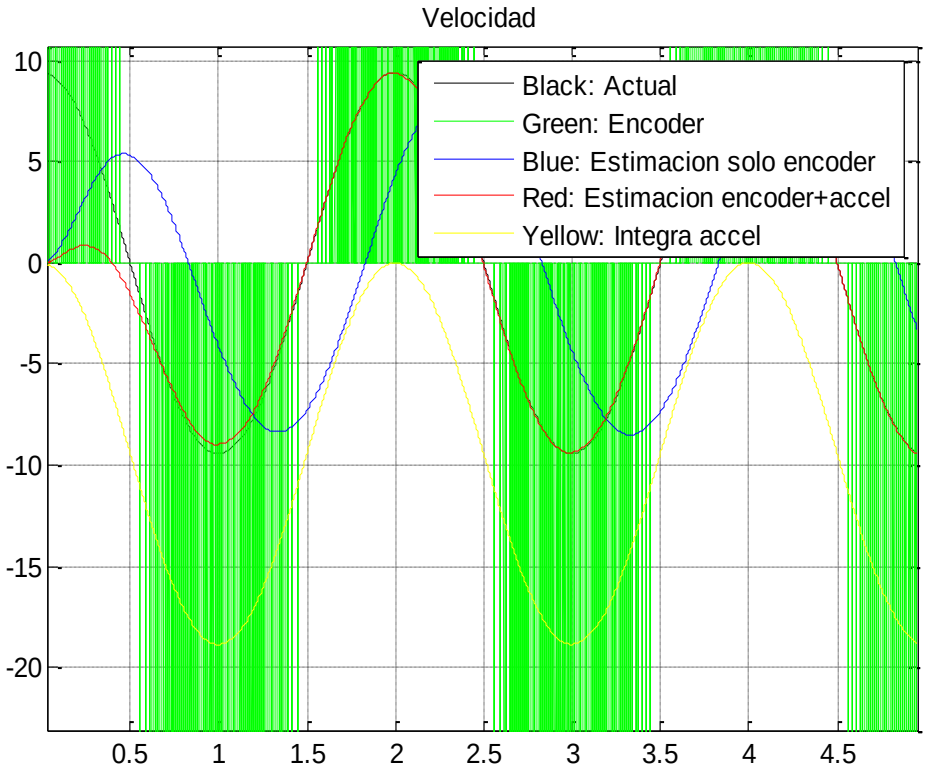
d. Considerando : q = 100 , r =1 , pulsos= 100 , $\varphi = 0$



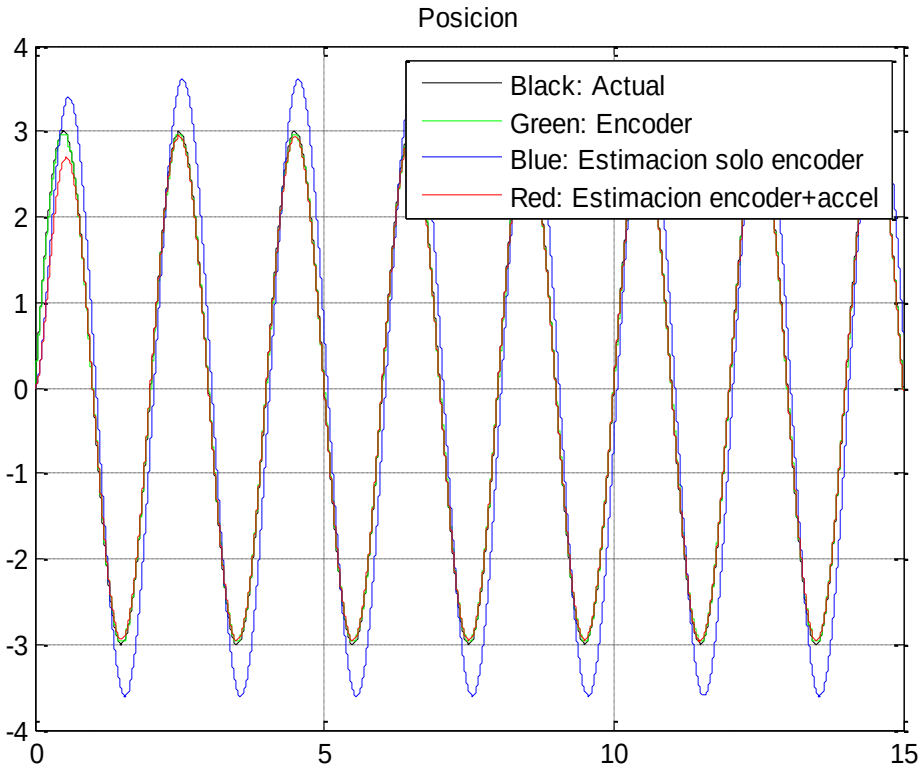


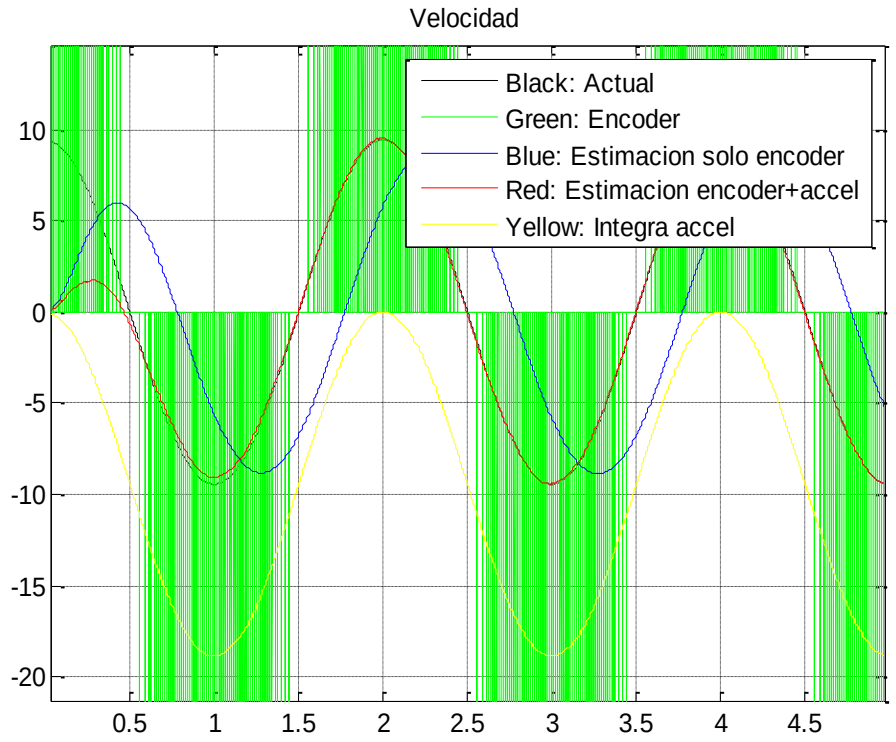
e. Considerando : q = 500 , r =1 , pulsos= 100 , $\varphi = 0$



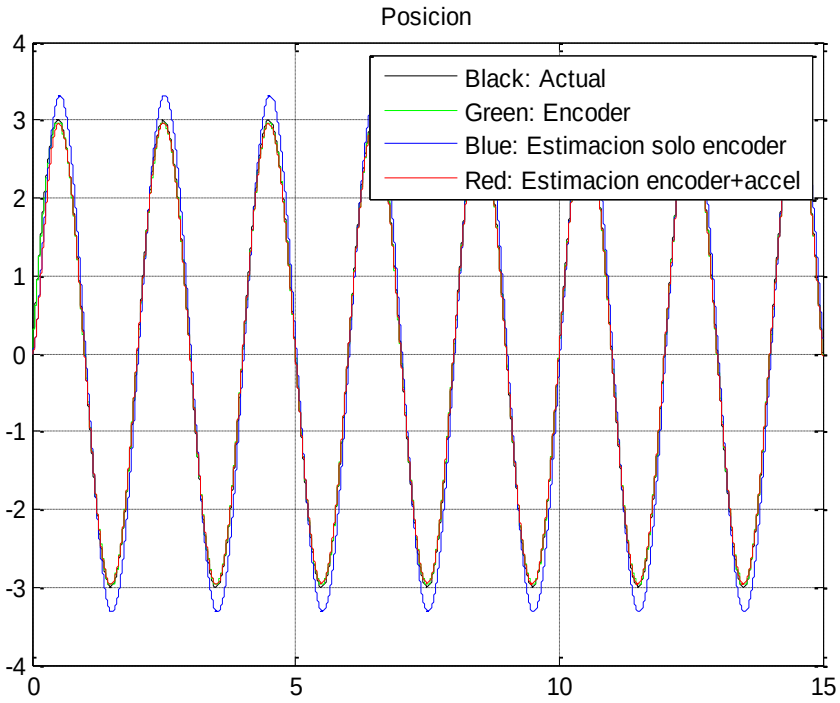


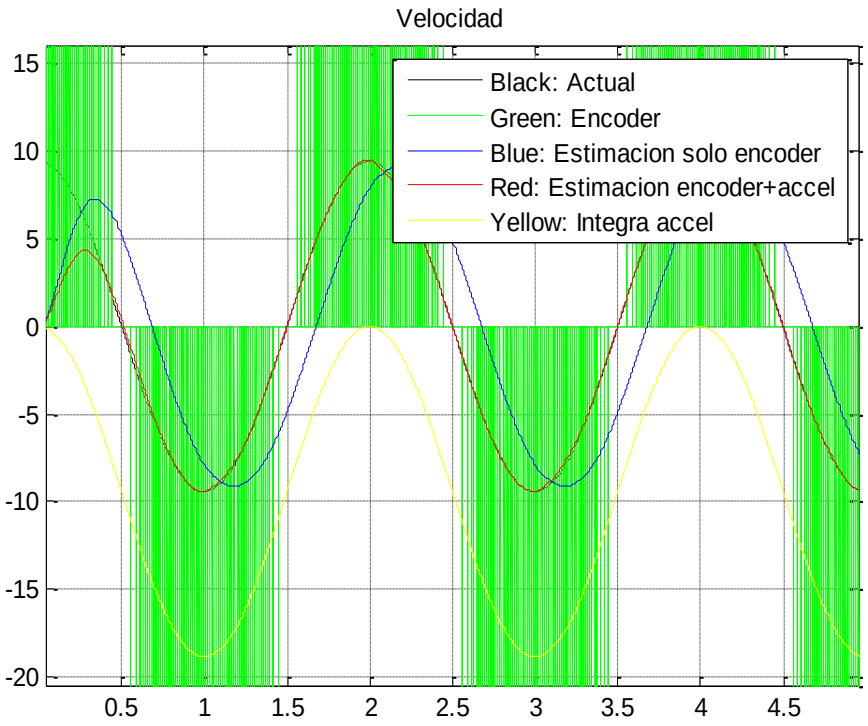
f. Considerando : q = 1000 , r =1 , pulsos= 100 , $\varphi = 0$





g. Considerando : q = 5000 , r =1 , pulsos= 100 , $\varphi = 0$



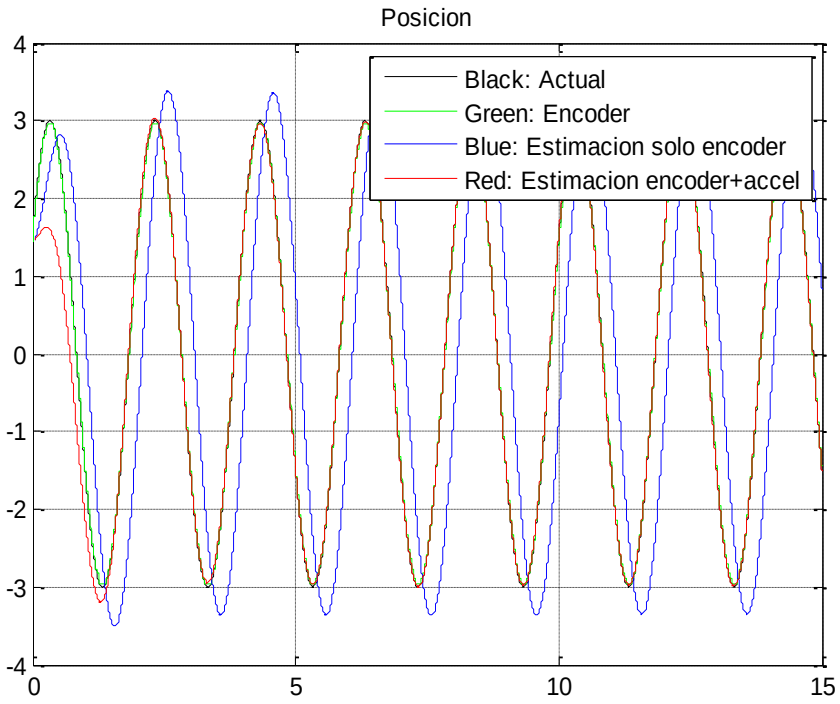


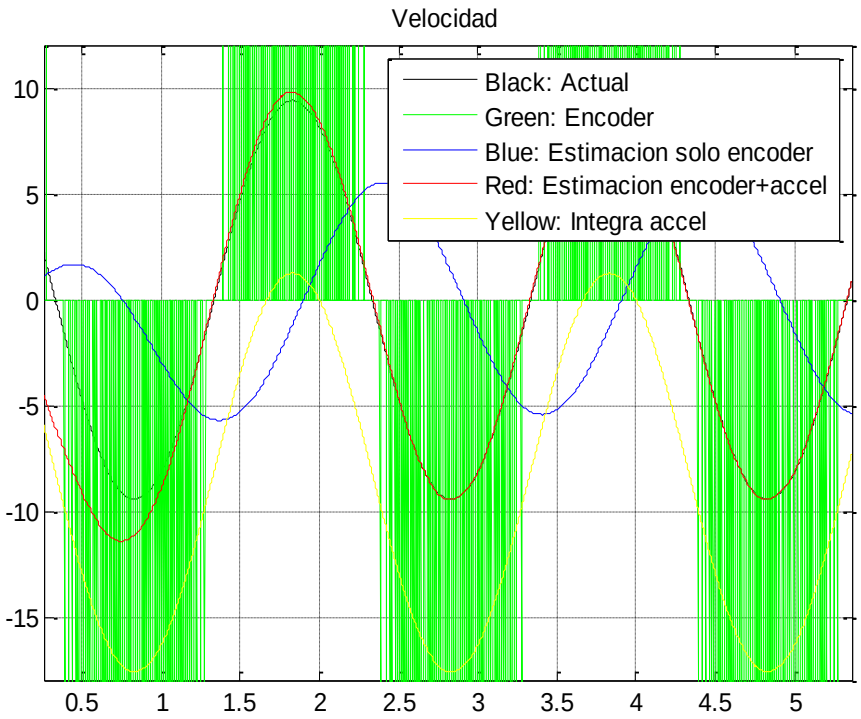
Se aprecia que al aumentar el peso “q” la convergencia de la estimación al valor actual es más rápida. Incluso la estimación de “solo encoder” también mejora y se acerca al valor actual.

Se probó que cuando el valor de “q” es muy grande la estimación se pierde en los picos y valles, además cuando es demasiado grande Riccati ya no tiene solución.

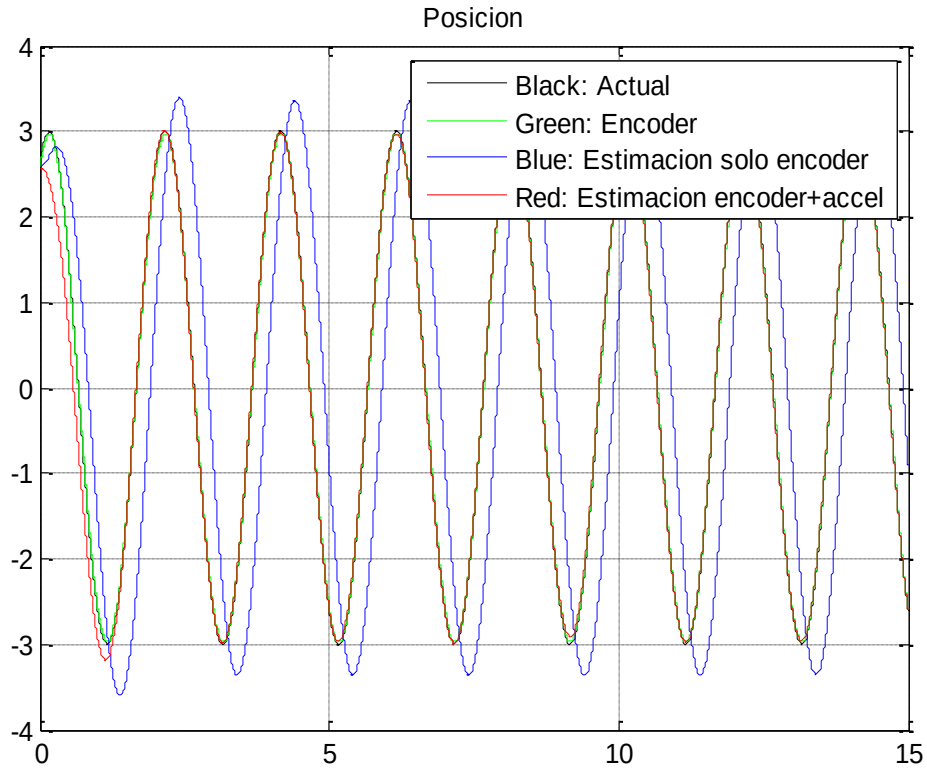
Variando la fase φ

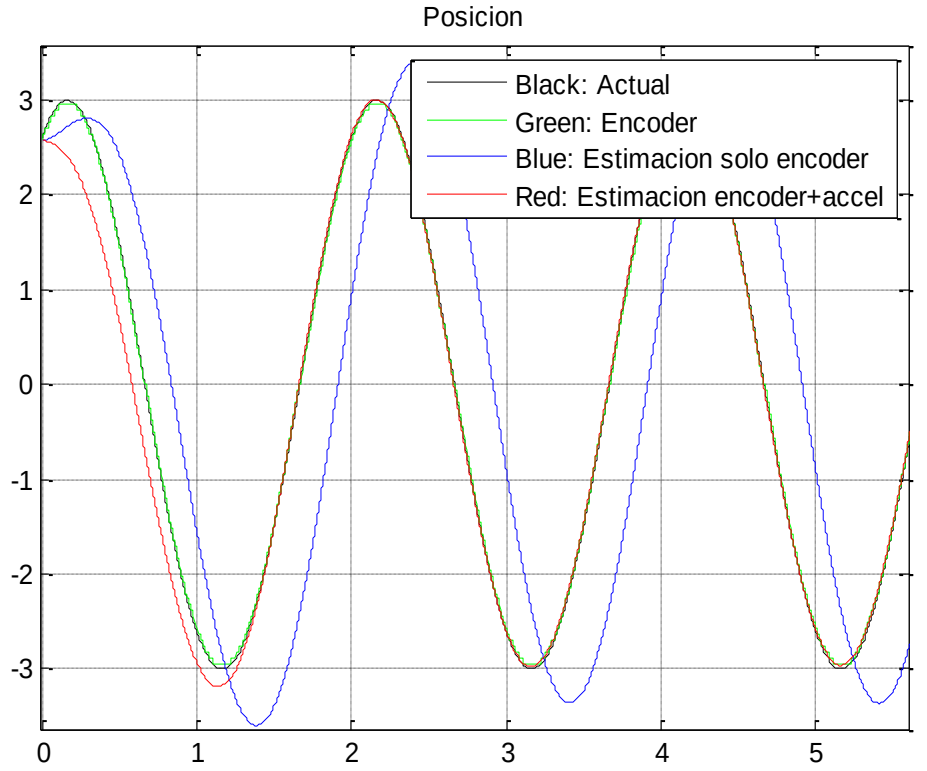
h. Considerando : q = 50 , r =1 , pulsos= 100 , φ = pi/6



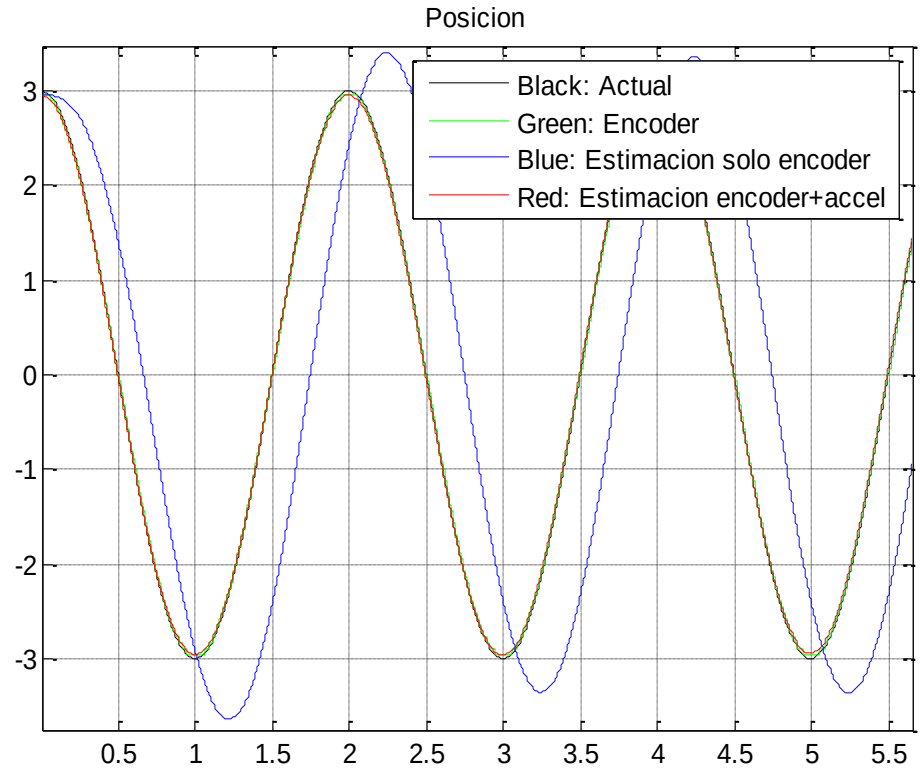


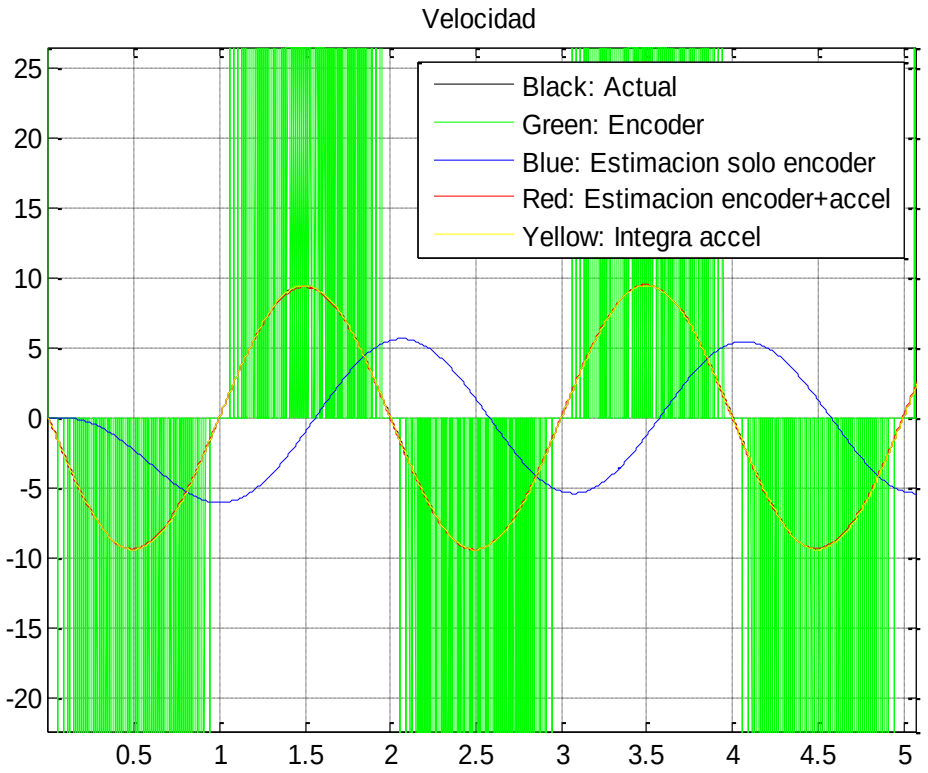
i. Considerando : q = 50 , r =1 , pulsos= 100 , φ = pi/3



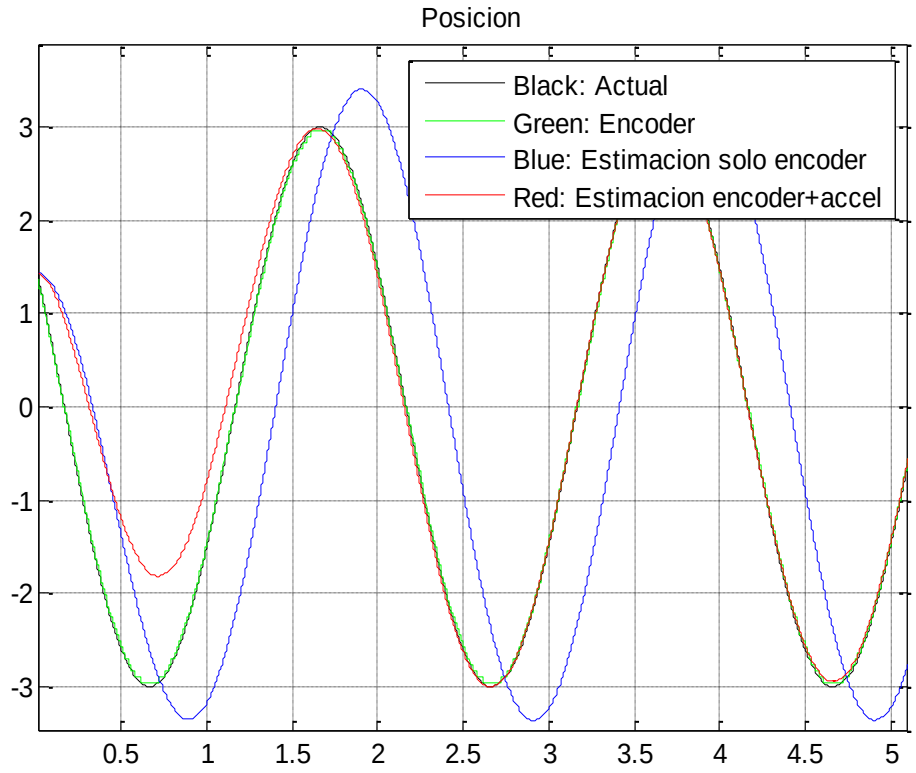


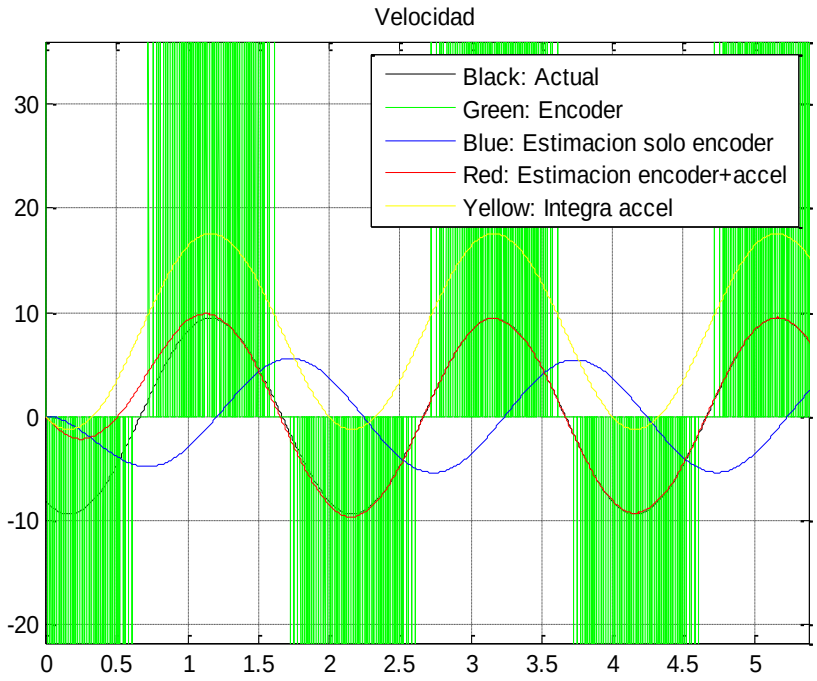
j. Considerando : q = 50 , r =1 , pulsos= 100 , φ = pi/2





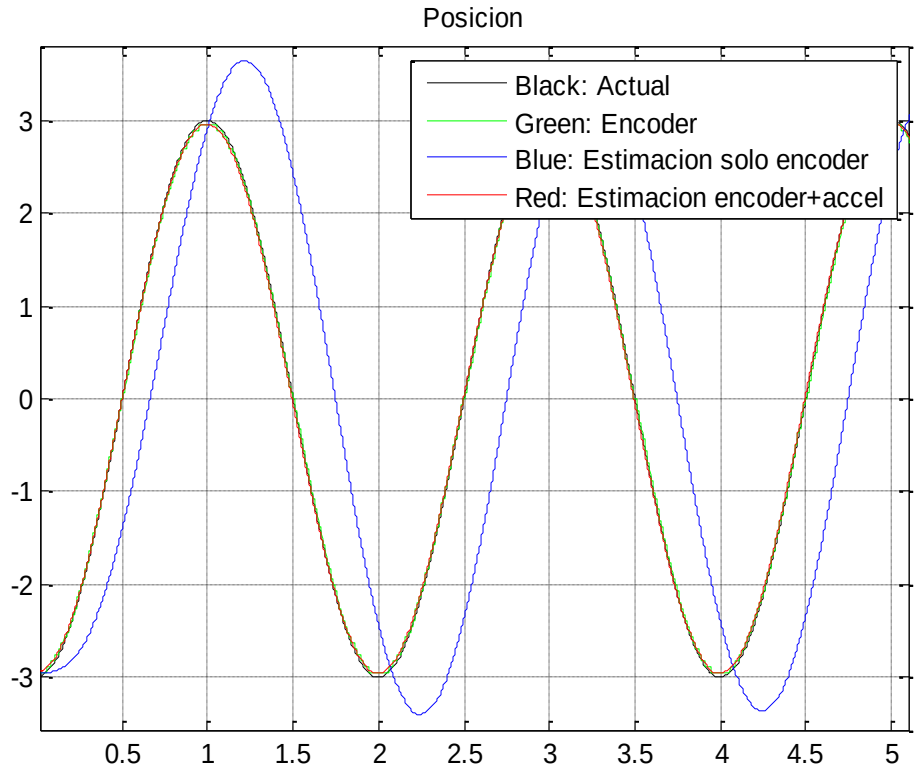
k. Considerando : q = 50 , r =1 , pulsos= 100 , φ = 5pi/6

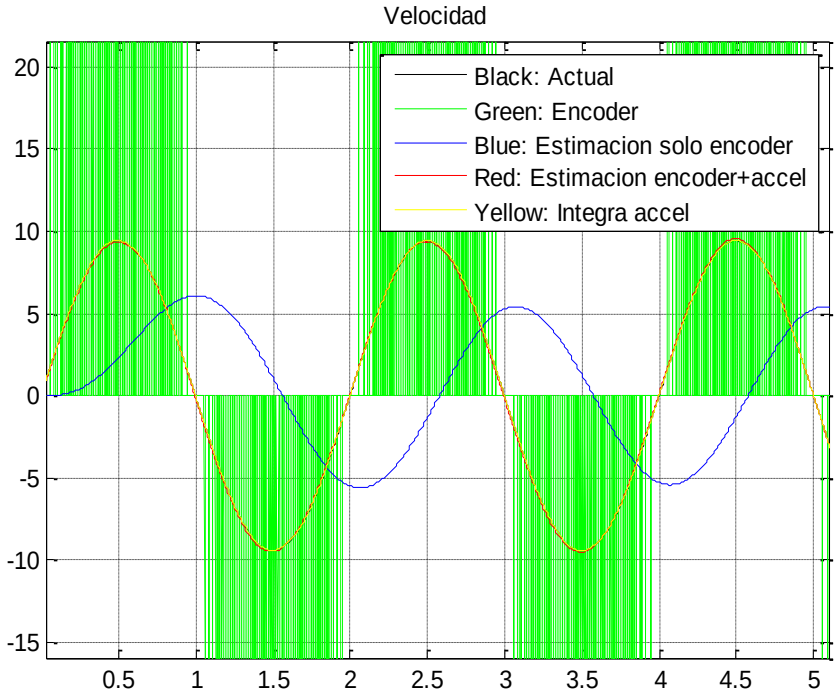




Para éste desfase, se demora más el estimador en converger.

I. Considerando : q = 50 , r =1 , pulsos= 100 , φ = 3pi/2





Cuando el desfase es múltiplo impar de pi/2 , (2k+1)*pi/2, la estimación converge desde el inicio.

Además se podría decir a partir de las pruebas realizadas que el tiempo de convergencia depende del desfase.

Conclusiones

Medición de temperatura.

1. Al incrementar la cantidad de sensores ruidosos utilizando el filtro de Kalman para la fusión de sensores, se logra mejorar la estimación de la variable medida.

2. No es necesario incrementar una gran cantidad de sensores para observar mejoras ya que a partir de 3 o 4 sensores ya no varía mucho el beneficio (La implementación de sensores tiene un costo considerable).

3. Cuando los sensores son ruidosos conviene utilizar el filtro de Kalman respecto al método del promedio, sin embargo si el ruido es pequeño no siempre se cumple.

Encoder + acelerómetro.

4. Al aumentar la cantidad de pulsos por vuelta del encoder, se mejora la estimación tanto de “solo encoder” como de “encoder + acelerómetro”.

5. Al incrementar el valor del peso “q” la estimación converge más rápido. Incluso la estimación con solo encoder mejora considerablemente en posición pero sigue el desfase en la estimación de velocidad.

6. Al variar la fase φ de la posición, afecta al tiempo de convergencia. Se observó que para fases que son múltiplos impares de pi/2 se logra convergencia desde el inicio y para los otros valores depende de su valor.