Identificación y observación simultáneas (causal) mediante filtro de Kalman Extendido

Presentación en vídeo:

http://personales.upv.es/asala/YT/V/ekfid1.html , http://personales.upv.es/asala/YT/V/ekfid2.html .

Este código ejecutó correctamente con `Matlab` **`R2020a`**

Objetivos: Comprender como un filtro de Kalman no-lineal (en este caso, el filtro extendido) puede ser usado para simultáneamente observar el estado de un sistema e identificar parámetros físicos del modelo del mismo, incluso cuando dichos parámetros no son constantes.

Tabla de Contenidos

Planteamiento de la idea básica

En una regresión $y = \theta x + v$, estimar θ supuesto constante, con x conocido lo escribimos como:

Ec. de estado: $\theta_{k+1} = \theta_k$; Ec. de salida: $y_k = \theta_k x_k + v_k$

La solución son los mínimos cuadrados recursivos... con olvido si cambiamos a $\theta_{k+1} = \theta_k + w_k$... entonces podemos hacer un filtro de Kalman (mínimos cuadrados recursivos sin olvido cuando $E(w_k w_k^T) \to 0$).

Un sistema lineal con unos ciertos parámetros físicos a identificar θ podría expresarse como:

$$\begin{pmatrix}\theta_{k+1}\\ x_{k+1}\end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & A(\theta_k)\end{pmatrix}\begin{pmatrix}\theta_k\\ x_k\end{pmatrix} + \begin{pmatrix}0\\ B(\theta_k)\end{pmatrix}u + \begin{pmatrix}w_{k,\theta}\\ w_{k,proceso}\end{pmatrix}$$

o, en contínuo:

$$\frac{d}{dt}\begin{pmatrix}\theta\\ x\end{pmatrix}=\begin{pmatrix}0 & 0\\ 0 & A(\theta)\end{pmatrix}\begin{pmatrix}\theta\\ x\end{pmatrix}+\begin{pmatrix}0\\ B(\theta)\end{pmatrix}u+ruidos$$

con ecuación de salida $y=(0\ \ C(\theta))\widetilde{x}+D(\theta)u$.

Se trata de un sistema no-lineal con un estado "extendido" $\widetilde{x}:=\begin{pmatrix}\theta\\ x\end{pmatrix}$ ... una vez entramos en el dominio "no lineal", podemos generalizar a:

Ec. estado: $\frac{d\theta}{dt}=0,\quad \frac{dx}{dt}=f(\theta,x,u)$, ... o sea $\frac{d}{dt}\begin{pmatrix}\theta\\ x\end{pmatrix}=\begin{pmatrix}0\\ f(\theta,x,u)\end{pmatrix}$

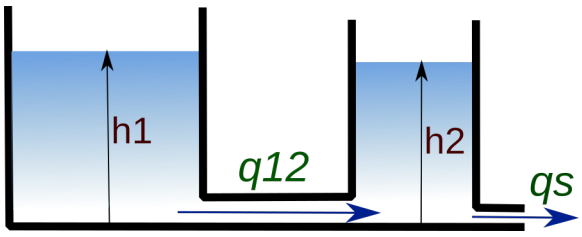
Ec: salida: $y=g(\theta,x,u)$

donde realmente θ es un estado más de un proceso no lineal, siendo el "estado" completo según la definición del mismo en teoría de sistemas el conjunto de θ y x, esto es, $\widetilde{x}$.

Denominando $\bar{A}(x,\theta,u)=jacobian(f)$, $\bar{C}(x,\theta,u)=jacobian(g)$, podemos diseñar un observador para dicha linealización, motivando esto el filtro de Kalman **extendido**, cuyos detalles no son objeto de este material, en el que sólo consideraremos la implementación: en el filtro extendido se linealiza ante trayectorias, por lo que los modelos linealizados son variantes en el tiempo; se debe usar el filtro de Kalman *no estacionario*, el filtro estacionario **dlqe NO vale**.

La idea es que si los parámetros son conocidos, dicho filtro es un "observador del estado x", si las trayectorias de estados x son conocidas entonces es un "estimador de parámetros (identificación)" y si "ni θ ni x son conocidos" entonces es un problema combinado de observación e identificación: estimar el estado extendido $\begin{pmatrix}\theta\\ x\end{pmatrix}$.

Modelado de un sistema de 2 depósitos (tiempo contínuo)




```
syms h1 h2 b12 b_s dh1 dh2 q_e q_s q12
S1=0.5;S2=1;
estados=[h1;h2]; params=[b12;b_s];
```

```
Modelo=[ dh1==(q_e-q12)/S1; dh2==(q12-q_s)/S2;...
   q12==b12*sqrt(h1-h2);q_s==b_s*sqrt(h2)]
```

```
Modelo =
```

$$\begin{pmatrix} \mathrm{dh}_1 = 2\,q_e - 2\,q_{12} \\ \mathrm{dh}_2 = q_{12} - q_s \\ q_{12} = b_{12}\,\sqrt{h_1 - h_2} \\ q_s = b_s\,\sqrt{h_2} \end{pmatrix}$$

```
sol=solve(Modelo,{dh1,dh2,q12,q_s});
EcEstado=[sol.dh1;sol.dh2]
```

```
EcEstado =
```

$$\begin{pmatrix} 2\,q_e - 2\,b_{12}\,\sqrt{h_1 - h_2} \\ b_{12}\,\sqrt{h_1 - h_2} - b_s\,\sqrt{h_2} \end{pmatrix}$$

Los sensores disponibles serán:

```
EcSalida_sym=[h1;h2;sol.q12]
```

```
EcSalida_sym =
```

$$\begin{pmatrix} h_1 \\ h_2 \\ b_{12}\,\sqrt{h_1 - h_2} \end{pmatrix}$$

```
EcEstado_num=matlabFunction(EcEstado)
```

```
EcEstado_num = function_handle with value:

    @(b12,b_s,h1,h2,q_e)[q_e.*2.0-b12.*sqrt(h1-h2).*2.0;-b_s.*sqrt(h2)+b12.*sqrt(h1-h2)]
```

```
EcSalida_num=matlabFunction(EcSalida_sym)
```

```
EcSalida_num = function_handle with value:

    @(b12,h1,h2)[h1;h2;b12.*sqrt(h1-h2)]
```

Discretización (Euler)

Integración Euler, aproximaremos x_+=x+Ts*dxdt

```
Ts=0.1;
EcAlturasDiscreto=@(b_12,b_s,h1,h2,q_e) ...
   max([h1;h2]+Ts*EcEstado_num(b_12,b_s,h1,h2,q_e),0);
```

Modelo extendido

El modelo "extendido" completo será que los parámetros son "más o menos constantes" (derivada = 0) y las alturas evolucionan conforme al modelo no lineal anterior:

```
EstadoExtendido=[params;estados]
```

```
EstadoExtendido =
```

$$\begin{pmatrix} b_{12} \\ b_s \\ h_1 \\ h_2 \end{pmatrix}$$

```
DerEstadoExtendido=[0;0;EcEstado]
```

```
DerEstadoExtendido =
```

$$\begin{pmatrix} 0 \\ 0 \\ 2\,q_e - 2\,b_{12}\,\sqrt{h_1 - h_2} \\ b_{12}\,\sqrt{h_1 - h_2} - b_s\,\sqrt{h_2} \end{pmatrix}$$

Linealización

```
A1_sym=jacobian(DerEstadoExtendido,EstadoExtendido)
```

```
A1_sym =
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -2\,\sqrt{h_1 - h_2} & 0 & -\frac{b_{12}}{\sqrt{h_1 - h_2}} & \frac{b_{12}}{\sqrt{h_1 - h_2}} \\ \sqrt{h_1 - h_2} & -\sqrt{h_2} & \frac{b_{12}}{2\,\sqrt{h_1 - h_2}} & -\frac{b_s}{2\,\sqrt{h_2}} - \frac{b_{12}}{2\,\sqrt{h_1 - h_2}} \end{pmatrix}$$

```
A1_num=matlabFunction(A1_sym)
```

```
A1_num = function_handle with value:

    @(b12,b_s,h1,h2)reshape([0.0,0.0,sqrt(h1-h2).*-2.0,sqrt(h1-h2),0.0,0.0,0.0,-sqrt(h2),0.0,0.0,-b12.*1.0
```

```
Bext=eval(jacobian(DerEstadoExtendido,q_e)) %modelo lineal en la entrada
```

```
Bext = 4×1
     0
     0
     2
     0
```

```
C_sym=jacobian(EcSalida_sym,EstadoExtendido)
```

```
C_sym =
```

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \sqrt{h_1 - h_2} & 0 & \frac{b_{12}}{2\,\sqrt{h_1 - h_2}} & -\frac{b_{12}}{2\,\sqrt{h_1 - h_2}} \end{pmatrix}$$

```
C_num=matlabFunction(C_sym)
```

```
C_num = function_handle with value:

    @(b12,h1,h2)reshape([0.0,0.0,sqrt(h1-h2),0.0,0.0,0.0,1.0,0.0,(b12.*1.0./sqrt(h1-h2))./2.0,0.0,1.0,b12.
```

Si fuéramos a linealizar en un punto de equilibrio "fijo, constante" calcularíamos el punto de funcionamiento para sustituirlo en los jacobianos:

```
q_epf=1.5; bs_pf=1.2; b12_pf=1;
%ptoeq=solve([Modelo; dh1==0; dh2==0; q_e==q_epf;bs==bs_pf;b12==b12_pf]);
%eval(ptoeq.h1)
%eval(ptoeq.h2)
```

Pero en filtro de Kalman "extendido" **no lo haremos,** dado que se linealiza alrededor de **trayectorias.**

Simulación del observador

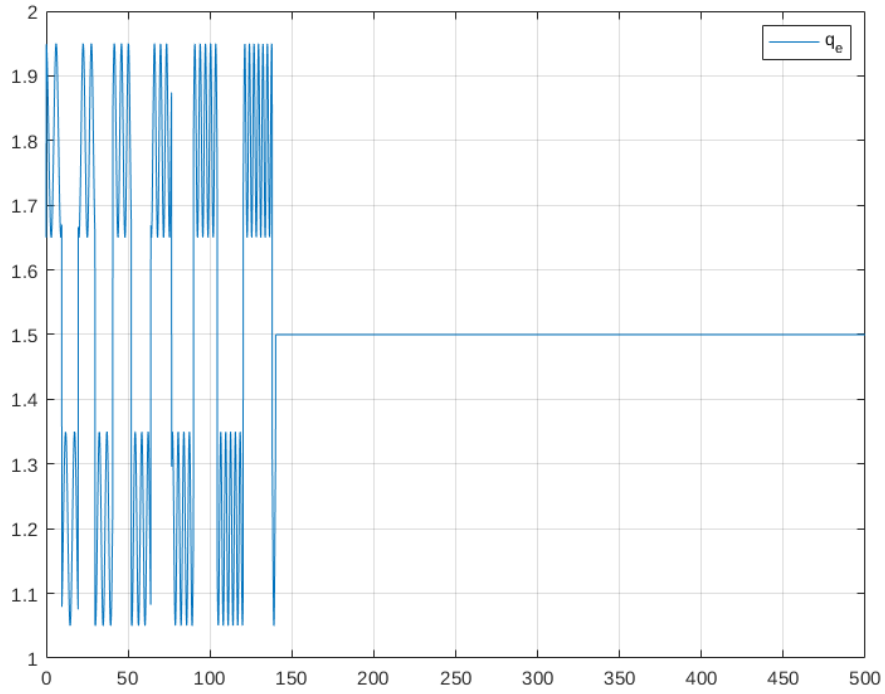
Importante: las raíces cuadradas de las ecuaciones teóricas sólo darán un resultado correcto si $h_1 \geq h_2$. Si las trayectorias o condiciones iniciales hicieran que en algún momento se verificara $h_1 < h_2$ entonces habría que hacer modificaciones al modelo... la ecuación "correcta" sería $q_{12} = sign(h_1 - h_2) \cdot \sqrt{|h_1 - h_2|}$, y la linealización sería:

$$\Delta q_{12} = \frac{sign(h_1 - h_2)}{2\sqrt{|h_1 - h_2|}}(\Delta h_1 - \Delta h_2).$$

```
TiemposSimulacion=0:Ts:500;
Nmuestras=length(TiemposSimulacion);
```

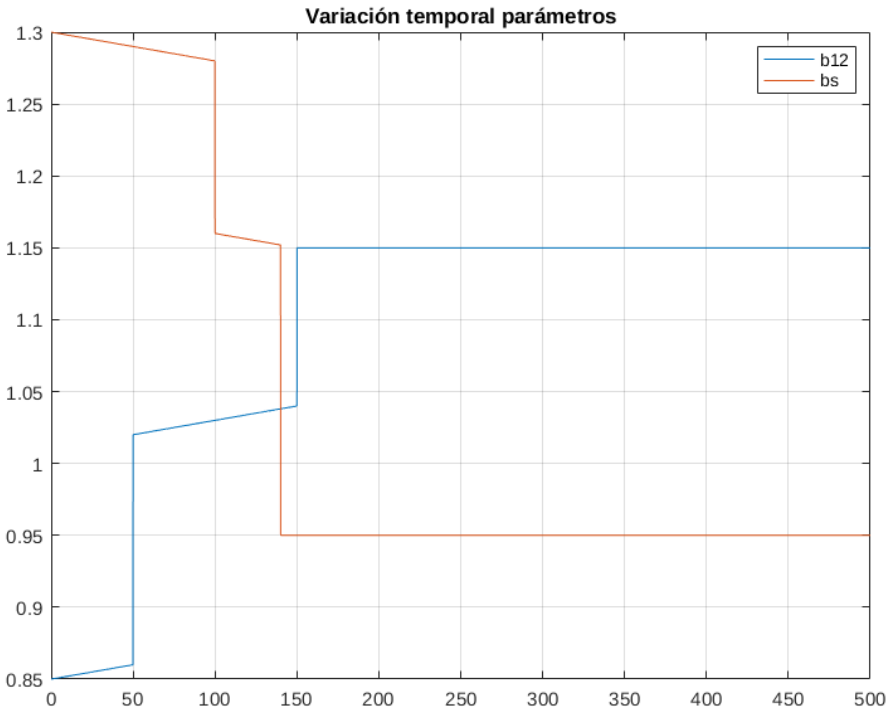
El caudal de entrada será:

```
q_e=q_epf+0.3*sign(sin(TiemposSimulacion/3-0.0006*TiemposSimulacion.^2)) ...
   +0.15*(cos(TiemposSimulacion+0.005*TiemposSimulacion.^2));
q_e(TiemposSimulacion>140)=q_epf;
plot(TiemposSimulacion,q_e'), grid on, legend('q_e')
```



Y los valores de los parámetros variarán en el tiempo:

```
b12=.65+0.2*linspace(1,1.5,Nmuestras)+0.16*(TiemposSimulacion>=50);
b12(TiemposSimulacion>150)=1.15;
bs=0.8+0.5*linspace(1,0.8,Nmuestras)-0.12*(TiemposSimulacion>=100);
bs(TiemposSimulacion>140)=0.95;
plot(TiemposSimulacion,[b12;bs]'), grid on, legend('b12','bs'), title("Variación tempor
```



Los parámetros del ruido del filtro serán:

```
VCEstadoEx=(8e-1)^2*eye(4);%inicial
VCEstado_antes_medir=VCEstadoEx;
desvTipRuidoMed=diag([0.7e2 0.6 0.3]);
VCRuidoMed=desvTipRuidoMed*desvTipRuidoMed';
Nsens=size(desvTipRuidoMed,1);
Bruido=[[1 0;0 1;0 0;0 0] Bext*Ts]
```

```
Bruido = 4×3
    1.0000         0         0
         0    1.0000         0
         0         0    0.2000
         0         0         0
```

```
VCRuidoParams=diag([0.004;0.004])^2;
DesvTip_Caudal=0.005;
VCRuidoProc=blkdiag(VCRuidoParams,DesvTip_Caudal^2);
```

Las condiciones iniciales del proceso real, en la simulación, serán:

```
h1=2;h2=1;
alturas_real=[h1;h2];
```

Y las del proceso estimado:

```
h1_est=3.6; h2_est=1.5;
```

Siendo los valores iniciales en el estimador de parámetros:

```
b12_est=0.9;bs_est=1.2;
```

El vector de estado "extendido" estimado del observador inicial será:

```
Xestim=[b12_est;bs_est;h1_est;h2_est];
```

Guardaremos resultados en gráficas

```
grafH=[];grafH_est=[];grafb_est=[];
grafmedidas=[];grafdesvtip=[];
```

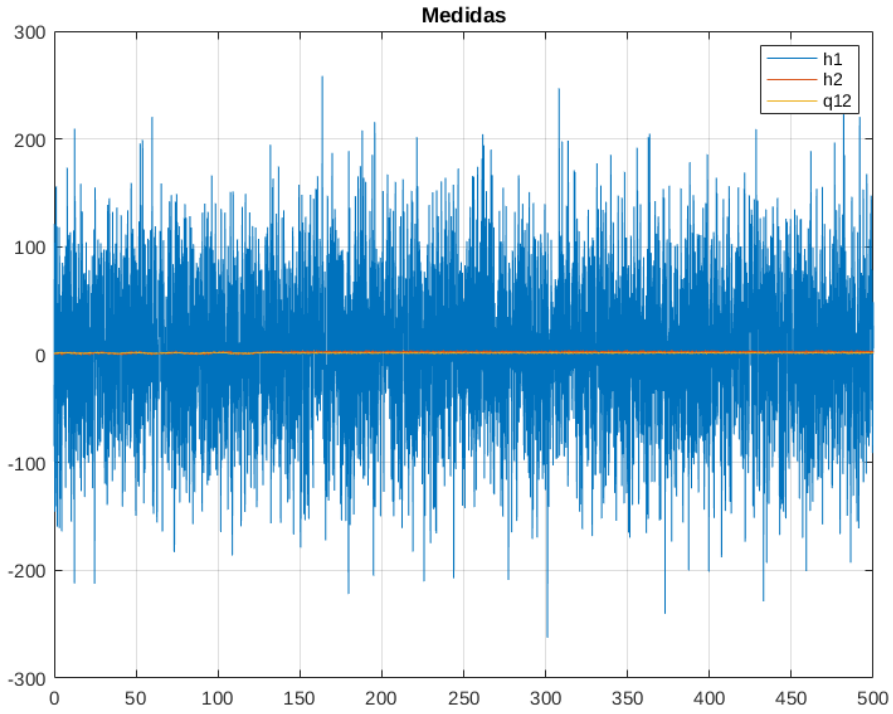
Simulamos...

```
for k=1:Nmuestras
   %Leemos sensores
   medidas=EcSalida_num(b12(k),h1,h2)+desvTipRuidoMed*randn(Nsens,1);


   %Linealizamos la ecuación de salida
   Csens=C_num(b12_est,h1_est,h2_est);
   %Calculamos ganancia del observador
   GananciaKalman= ...
      VCEstado_antes_medir*Csens'/(Csens*VCEstado_antes_medir*Csens'+VCRuidoMed);
   %corrector
   Xestim=Xestim+GananciaKalman*(medidas-EcSalida_num(b12_est,h1_est,h2_est));
   %Varianza a posteriori, después de la medida:
   VCEstadoEx=VCEstado_antes_medir-GananciaKalman*Csens*VCEstado_antes_medir;
   %hacemos simétrica si no lo es por redondeo:
   VCEstadoEx=(VCEstadoEx+VCEstadoEx')/2;

   %esta es la salida "en media" del filtro de Kalman extendido:
   b12_est=Xestim(1); bs_est=Xestim(2);  %parámetros
   h1_est=Xestim(3); h2_est=Xestim(4);   %alturas

   %Hacemos gráficas de un montón de cosas:
   grafmedidas=[grafmedidas medidas];
   grafH=[grafH alturas_real];
   grafH_est=[grafH_est [h1_est;h2_est]];
   grafb_est=[grafb_est [b12_est;bs_est]];
   grafdesvtip=[grafdesvtip sqrt(diag(VCEstadoEx))];

   %predictor:
   %simulamos el modelo estimado para siguiente muestreo...
   %los parámetros son simplemente "constantes", no cambian en media, no hace
   %falta simular ninguna ecuación de estado...
   Xestim(1:2)=Xestim(1:2); %esto sobra, pero lo pongo "conceptualmente"
   %las alturas, simulamos el modelo:
```

```
    Xestim(3:4)=EcAlturasDiscreto(b12_est,bs_est,h1_est,h2_est,q_e(k));
    %sin ruido de proceso, claro...
    %esta es la salida "en media" del filtro de Kalman extendido:
    b12_est=Xestim(1); bs_est=Xestim(2);  %parámetros
    h1_est=Xestim(3); h2_est=Xestim(4);   %alturas

    %Linealizamos sobre la trayectoria estimada
    % (evaluando numéricamente los jacobianos)
    Alinealizado=A1_num(b12_est,bs_est,h1_est,h2_est);
    %discretizamos la ecuación linealizada continua
    Alinealz_disc=eye(4)+Alinealizado*Ts;
    %actualizamos la varianza del estado antes de medir (siguiente estado)
    VCEstado_antes_medir= ...
       Alinealz_disc*VCEstadoEx*Alinealz_disc'+Bruido*VCRuidoProc*Bruido';



    %proceso real: escribir actuadores y esperar un período de muestreo
    alturas_real=EcAlturasDiscreto(b12(k),bs(k),h1,h2,q_e(k)+randn()*DesvTip_Caudal);
    h1=alturas_real(1);h2=alturas_real(2);

end
```

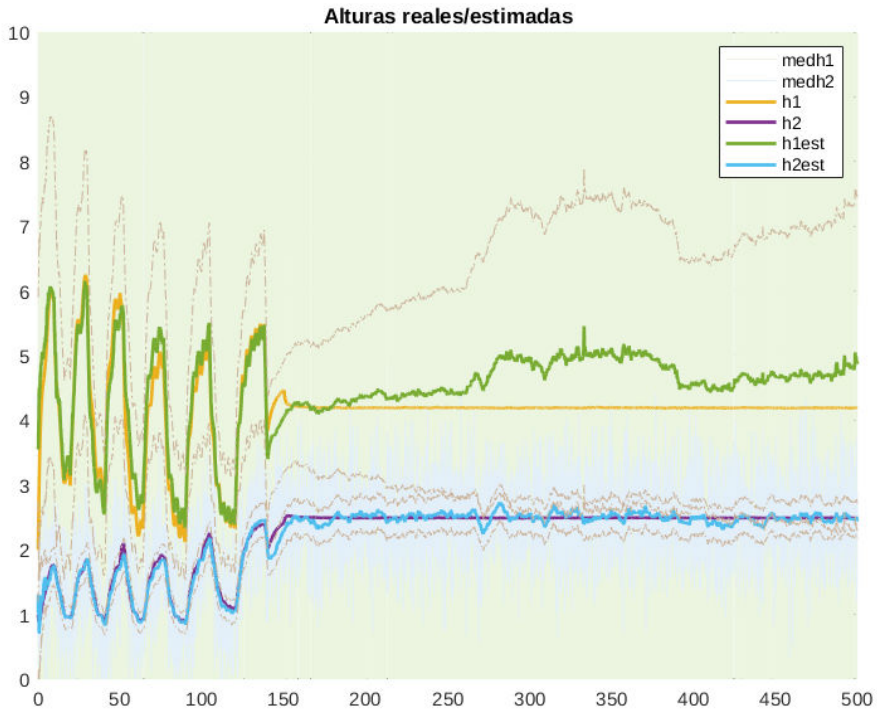
Gráficas de resultados

```
plot(TiemposSimulacion,grafmedidas'), title("Medidas"),legend('h1','h2','q12'), grid or
```

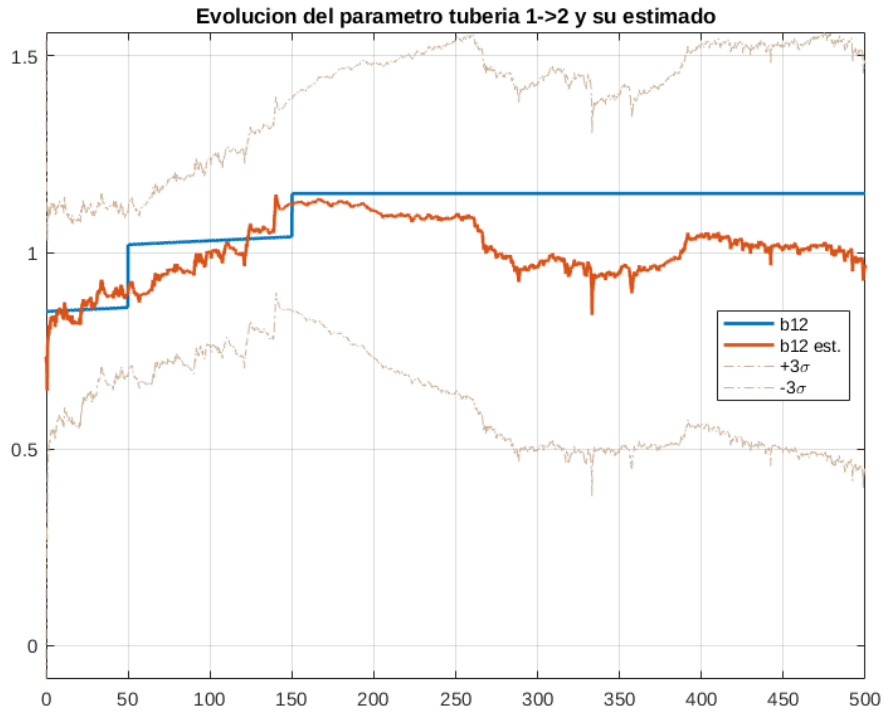


Dibujamos el comportamiento "en media" y añadimos intervalos de confianza $\pm 3\sigma$:

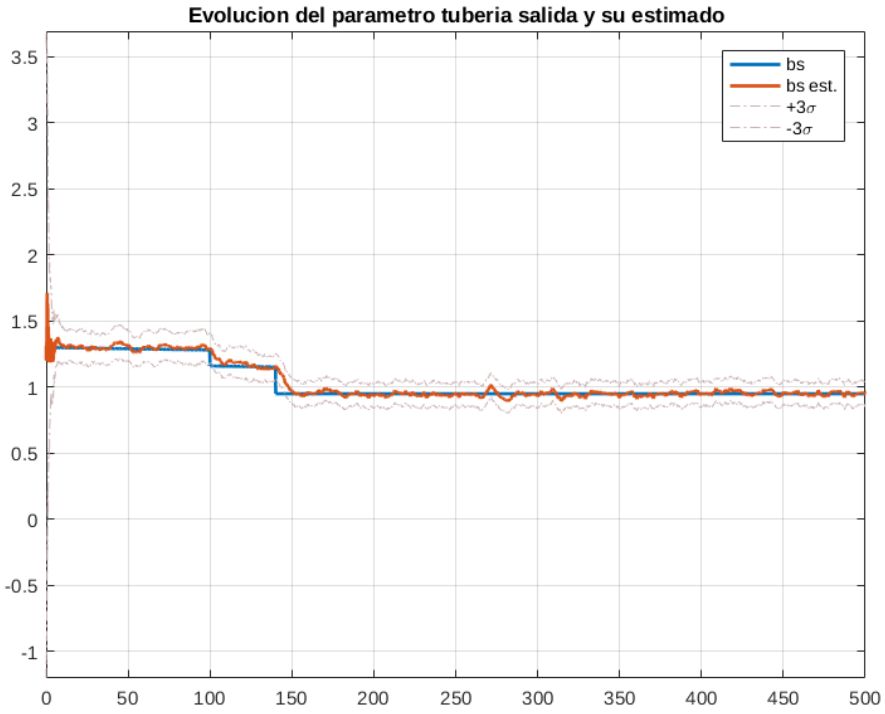
```
plot(TiemposSimulacion,grafmedidas(1,:)','Color',[.92 .96 .88])
hold on
plot(TiemposSimulacion,grafmedidas(2,:)','Color',[.89 .94 .98])
plot(TiemposSimulacion,[grafH;grafH_est]','LineWidth',2 ), title("Alturas reales/estima
grid on, hold on
plot(TiemposSimulacion,[grafH_est+grafdesvtip(3:4,:)*3; grafH_est-grafdesvtip(3:4,:)*3]
hold off, legend('medh1','medh2','h1','h2','h1est','h2est'), axis tight
ylim([0,10])
```




```
plot(TiemposSimulacion,[b12;grafb_est(1,:)],'LineWidth',2); hold on
plot(TiemposSimulacion,[grafb_est(1,:)+grafdesvtip(1,:)*3; grafb_est(1,:)-grafdesvtip(1
legend('b12','b12 est.','+3\sigma','-3\sigma','Location','best'), hold off
title("Evolucion del parametro tuberia 1->2 y su estimado"), axis tight
```




```
plot(TiemposSimulacion,[bs;grafb_est(2,:)],'LineWidth',2); hold on
plot(TiemposSimulacion,[grafb_est(2,:)+grafdesvtip(2,:)*3; grafb_est(2,:)-grafdesvtip(2
legend('bs','bs est.','+3\sigma','-3\sigma','Location','best'), hold off
title("Evolucion del parametro tuberia salida y su estimado"), axis tight
```



Discusión y conclusiones

El filtro de Kalman extendido presentado permite:

- Observar "estados" e identificar parametros *simultáneamente*, incluso en procesos no lineales.
- Seguir trayectorias de parámetros que *varían* en el tiempo (como los minimos cuadrados recursivos con olvido)
- Estimar parámetros "*físicos*", y no coeficientes sin significado físico claro como en los modelos polinomiales $y_{k+2} = a_1 y_{k+1} + a_0 y_k + b_0 u_k$ o subespacio (a partir de SVD).

Aunque sus inconvenientes son:

- Está basado en linealización, pero estamos observando un modelo no lineal... No hay garantía de convergencia a estados/parámetros correctos, sobre todo si las condiciones iniciales del filtro están lejos de las reales. Necesaria "simulación exhaustiva"...
- Más sensible al ruido que la identificación o la observación lineales por separado: estimar x y θ simultáneamente en $y = \theta x + v$ es imposible... hay que ayudarse de la dinámica, etc... conveniente tener buenos sensores en el sitio adecuado.
- Problema de la "excitación": si todo está en "equilibrio" las matrices de varianzas-covarianzas podrían crecer... supervisión del olvido o dinámica con "decay" de los parámetros.
- Aunque varianza del "ruido de medida" podría suponerse conocida, estimar la varianza del ruido de proceso, que esta vez incluye a los posibles "cambios en los parámetros" no es fácil: se deben elegir las "varianzas de ruido de parámetros" y de "estados" de modo que se alcance un compromiso entre sensibilidad a ruido, capacidad de detección de cambios "bruscos" en parámetros, rapidez de polos del observador,...
- ¿Que pasa con errores de modelado, dinamica adicional no modelado, ruido no distribucion normal?