Identificación de respuestas ante impulso regularizadas por un kernel de covarianza: ejemplo Matlab

© 2020, Antonio Sala Piqueras, Universitat Politècnica de València. Todos los derechos reservados.

Este código funcionó correctamente con Matlab **R2020b**

Objetivo: la identificación de respuestas ante impulso de sistemas lineales con pocos datos tiene gran varianza. Si suponemos que la respuesta impulsional es una función incierta, pero con una cierta "suavidad", podemos regularizar el problema.

Bibliografía:

Ljung *et. al.*, A shift in paradigm for system identification https://doi.org/10.1080/00207179.2019.1578407

Pilloneto & Nicolao, A new kernel-based approach for linear system identification https://doi.org/10.1016/j.automatica.2009.10.031

Tabla de Contenidos

0.- Revisión Teórica

Fórmula de convolución

Si la respuesta ante $\{1,0,0,\dots\}$ es $\{h_0,h_1,h_2,\dots\}$, denominada *respuesta impulsional*, entonces,

por linealidad (superposición), la respuesta ante $\{u_0,u_1,u_2,u_3,\dots\}$ y condiciones iniciales nulas es

$\{y_0=u_0h_0, y_1=u_0h_1+u_1h_0, y_2=u_0h_2+u_1h_1+u_2h_0, y_3=u_0h_3+u_1h_2+u_2h_1+u_3h_0,\dots\}$, que se denomina **fórmula de convolución discreta**.

Escribiendo la última expresión en forma de matriz:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} u_0 & 0 & 0 & \dots & 0 \\ u_1 & u_0 & 0 & \dots & 0 \\ u_2 & u_1 & u_0 & \dots & 0 \\ \vdots & & \vdots & & \vdots \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \end{pmatrix} = T \cdot \mathbf{h}$$

ello permite estimar $\mathbf{h}$ dados datos de entrada $\{u_0, u_1, u_2, u_3, \dots\}$ y salida $\{y_0, y_1, y_2, y_3, \dots\}$ por mínimos cuadrados. Como en un sistema estable los elementos de la respuesta impulsional tienden a cero, se limita el número de elementos de $\mathbf{h}$ (y de columnas de T) a un tamaño "razonable" N.

Mínimos cuadrados recursivos

Si se tiene información *a priori* de $\mathbf{h}$, se puede incorporar, con fórmula de **mínimos cuadrados recursivos**. Si creemos que $\mathbf{h}$ es $\mathcal{N}(0, \Sigma)$, entonces, a partir de medidas de y con ruido de medida de varianza V (usualmente diagonal) podemos refinar la estimación, teniendo un estimado *a posteriori* dado por:

$$\widehat{\theta} = \Sigma T^T (T \Sigma T^T + V)^{-1} y$$

y la varianza *a posteriori* se reduce con la nueva información:

$$\Sigma_1 = \Sigma - \Sigma T^T (T \Sigma T^T + V)^{-1} T \Sigma$$

Kernel de covarianza de un proceso gaussiano (función aleatoria)

La respuesta impulsional no es un conjunto de números "al azar"... es una función del tiempo (discreto en este caso) más o menos "suave", de modo que valores cercanos deben estar "relacionados" (si el período de muestreo es suficientemente alto, no debería haber componentes de "alta frecuencia").

La correlación estadística (bueno, en rigor, varianza y covarianza) entre distintas muestras de $\mathbf{h}$ podría establecerse como una cierta función de covarianza κ de modo que la matriz Σ se pudiera construir como:

$\Sigma_{ij} = E(h_i h_j) = \kappa(i, j)$

expresando dicha *"suavidad a priori"* de $\mathbf{h}$.

H es una función del tiempo, desde 0 (h_0) hasta el número de parámetros ("orden" en este fichero) menos 1.

Probaremos distintas opciones de κ, construyendo un estimado a priori de la covarianza de los parámetros a estimar como $\Sigma = K(0 : (orden - 1), 0 : (orden - 1))$ donde la matriz simétrica positiva-definida K se suele conocer como matriz "***kernel***", construida con $\kappa(i, j)$. Con ello, usando las fórmulas de mínimos cuadrados recursivos podremos "regularizar" el problema de estimación dando más probabilidad a priori a determinadas "formas de onda" de $\mathbf{h}$ que a otras.

Opciones para $\kappa(i, j)$:

- Suponer $\mathbf{h}$ un proceso **estacionario**, con correlación dependiente de la diferencia de tiempos $\kappa(i, j) = \overline{\kappa}(|j - i|)$, usualmente decreciente. Es el enfoque más popular en estimación/suavizado (Kriging), pero posiblemente no es el mejor para este caso concreto.

- Suponer $\mathbf{h}$ un proceso **no estacionario**, de modo que la varianza en función del tiempo *no* sea constante: en procesos asintóticamente estables tiene que decrecer hacia cero con una cierta tasa (polo dominante). Existen propuestas en la literatura usando la transformación exponencial $\kappa(i,j) = \overline{\kappa}(e^{-i/\tau}, e^{-j/\tau})$, para ciertas $\overline{\kappa}$. **Estabilidad a priori:** cuando $i, j \to \infty$ verifique $\kappa(i,j) \to 0$. Con la transf. exponencial, $\overline{\kappa}(0,0) = 0$.

1.- Modelado y simulación para conseguir datos de entrenamiento/ validación

Modelo:

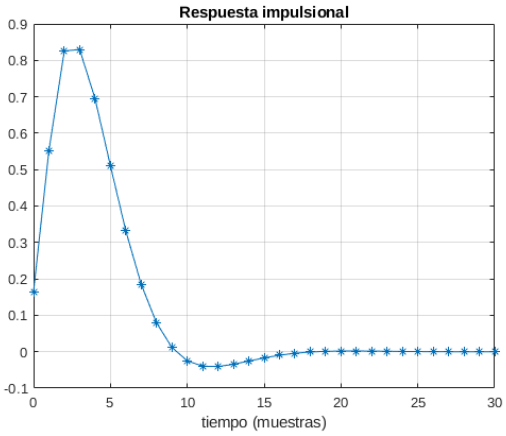
```
[b,a]=butter(2,.15);
G=tf(b*4,a,1)
```

```
G =

  0.165 z^2 + 0.33 z + 0.165
  --------------------------
    z^2 - 1.349 z + 0.514

Sample time: 1 seconds
Discrete-time transfer function.
```

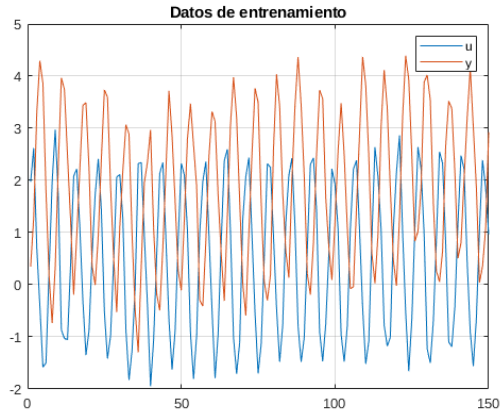
```
orden=31;ejeTh=0:(orden-1);
thperfecto=impulse(G,orden-1);
plot(ejeTh,thperfecto,'Marker','*'), grid on, title("Respuesta impulsional"), xlabel("tiempo (m
```



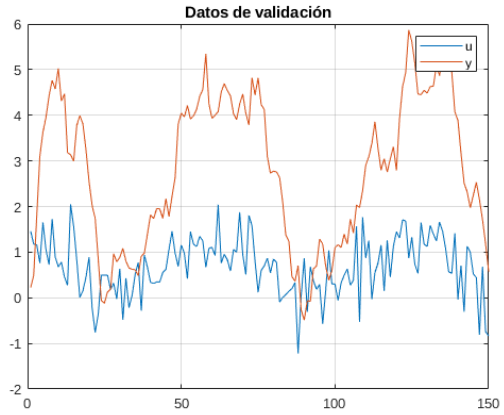
Generación de datos de entrenamiento y validación:

```
N=150;
u=2*sin(0.9*(1:N))'+0.2*randn(N,1)+0.3*(1:N)'/N+0.3;
uvalid=0.5*cos(0.1*(1:N))'+0.5*randn(N,1)+0.7;
dtruido=0.25;
y=lsim(G,u)+dtruido*randn(N,1);
yvalid=lsim(G,uvalid)+dtruido*randn(N,1);
```

```
plot([u y]), legend('u','y'), grid on, title("Datos de entrenamiento")
```




```
plot([uvalid yvalid]), legend('u','y'), grid on, title("Datos de validación")
```



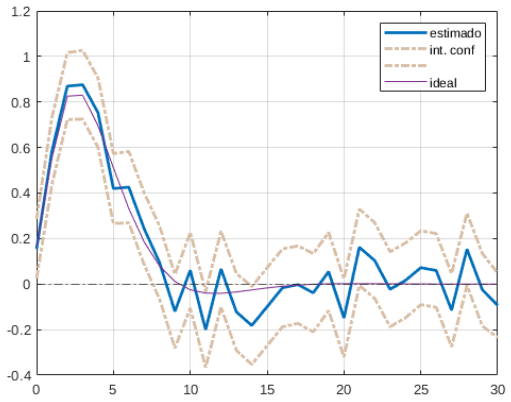
2.- Identificación de respuesta impulsional por mínimos cuadrados sin regularizar

```
T=toeplitz(u,[u(1) zeros(1,orden-1)])
```

```
T = 150×31
    1.9762         0         0         0         0         0         0         0 ···
    2.6185    1.9762         0         0         0         0         0         0
    0.7090    2.6185    1.9762         0         0         0         0         0
   -0.4046    0.7090    2.6185    1.9762         0         0         0         0
   -1.5813   -0.4046    0.7090    2.6185    1.9762         0         0         0
   -1.4951   -1.5813   -0.4046    0.7090    2.6185    1.9762         0         0
    0.2609   -1.4951   -1.5813   -0.4046    0.7090    2.6185    1.9762         0
    1.9719    0.2609   -1.4951   -1.5813   -0.4046    0.7090    2.6185    1.9762
    2.9735    1.9719    0.2609   -1.4951   -1.5813   -0.4046    0.7090    2.6185
    1.6981    2.9735    1.9719    0.2609   -1.4951   -1.5813   -0.4046    0.7090
      ⋮
```

```
%mínimos cuadrados estándard
th_noreg=pinv(T)*y;
varth=dtruido^2*inv(T'*T); %si los datos estuvieran generados por el modelo FIR+ruido blanco
```

```
plotIncierto(ejeTh,th_noreg,sqrt(diag(varth)))
hold on
plot(ejeTh,thperfecto)
hold off, yline(0,'-.')
legend('estimado','int. conf','','ideal')
```



El error sobre los datos de validación será:

```
Tvalid=toeplitz(uvalid,[uvalid(1) zeros(1,orden-1)]);
error_cuadr_noreg=sum((yvalid-Tvalid*th_noreg).^2)
```

```
error_cuadr_noreg = 18.8443
```

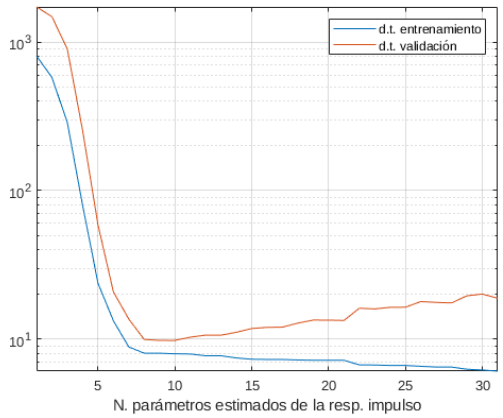
3a.- Regularización por truncación (selección de número de elementos de la respuesta a estimar)

***Por error sobre datos de validación**

Hay muchos elementos de la respuesta ante impulso cuyo intervalo de confianza toca el cero: puede que sean "verdaderamente cero" o al menos "no hay suficientes datos para asegurar que no sean cero". Si estimamos menos parámetros, tendremos menos varianza (pero más "sesgo").

Limitaremos el número de parámetros a estimar, viendo el ajuste sobre datos de validación:

```
ord_max=31;
error_test=zeros(1,ord_max);
error_val=zeros(1,ord_max);
for orden=1:ord_max
    %hacemos regresión
    T=toeplitz(u,[u(1) zeros(1,orden-1)]);
    Tvalid=toeplitz(uvalid,[uvalid(1) zeros(1,orden-1)]);
    thv=pinv(T)*y;
    %determinamos estadísticos del ajuste
    error_test(orden)=sum((y-T*thv).^2);
    error_val(orden)=sum((yvalid-Tvalid*thv).^2);
end
semilogy(1:ord_max,[error_test;error_val]), grid on, axis tight
legend("d.t. entrenamiento","d.t. validación")
xlabel("N. parámetros estimados de la resp. impulso")
```

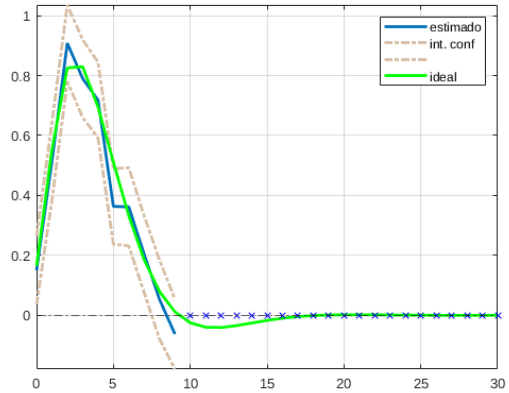



```
[mejor_err_val,orden_trunc]=min(error_val)
```

```
mejor_err_val = 9.8047
orden_trunc = 10
```

Repetimos la regresión:

```
T=toeplitz(u,[u(1) zeros(1,orden_trunc-1)]);
Tvalid=toeplitz(uvalid,[uvalid(1) zeros(1,orden_trunc-1)]);
th_reg=pinv(T)*y;
error_test=sum((y-T*th_reg).^2);
varth=error_test/(N-orden_trunc)*inv(T'*T);
std_th=sqrt(diag(varth));
plotIncierto(0:(orden_trunc-1),th_reg,std_th)
hold on
L=length(thperfecto);
plot(ejeTh,thperfecto,'g','LineWidth',2)
plot(orden_trunc:(L-1),zeros(L-orden_trunc,1),'xb')
hold off, yline(0,'-.'), axis tight
legend('estimado','int. conf','','ideal')
```



3b.- Mínimos cuadrados regularizados con Kernel de covarianza

Optimización de Hiperparámetros de la matriz de varianzas-covarianzas

```
orden=31; %volvemos a estimar "todos" los parámetros de h_0 a h_{orden-1}
```

```
T=toeplitz(u,[u(1) zeros(1,orden-1)]);
Tvalid=toeplitz(uvalid,[uvalid(1) zeros(1,orden-1)]);
kappa_que_probamos=@kappaIW;
tic
if(1)
    desvTrange=0.35:0.05:8;taurange=2:0.025:9.5;
    minEV=1e5;
    for des=desvTrange
        for tau=taurange
            hyperparams.desvt=des;hyperparams.tau=tau;
            K=Kernel(0:(orden-1),0:(orden-1),hyperparams,kappa_que_probamos);
            th2b=K*T'*((dtruido^2*eye(N)+T*K*T')\y);
            errK_valid=sum((yvalid-Tvalid*th2b).^2);
            if(errK_valid<minEV)
                hyperparamsOptimos=hyperparams;
                minEV=errK_valid;
            end
        end
    end
end
toc
```

```
Elapsed time is 46.111608 seconds.
```

```
minEV %la mejor desviación típica estimada sobre datos de validación.
```

```
minEV = 7.9825
```

```
hyperparamsOptimos
```

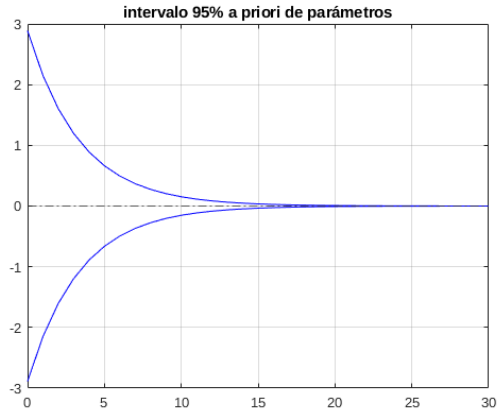
```
hyperparamsOptimos = struct with fields:
    desvt: 2.5000
      tau: 5.1000
```

```
K=Kernel(0:(orden-1),0:(orden-1),hyperparamsOptimos,kappa_que_probamos);
```

Análisis del modelo a priori

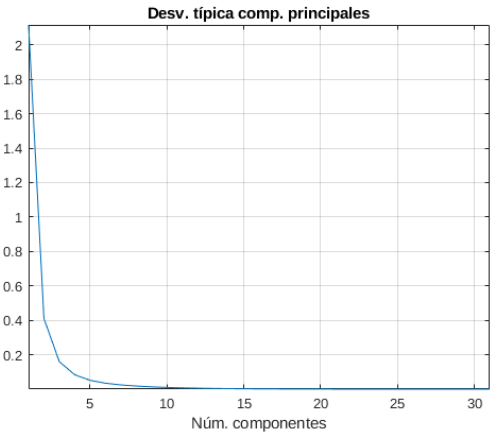
Intervalo de confianza

```
stdthba=sqrt(diag(K));
plot(ejeTh,[-stdthba*2 stdthba*2],'b'), grid on,
title('intervalo 95% a priori de parámetros'), yline(0,'-.')
```



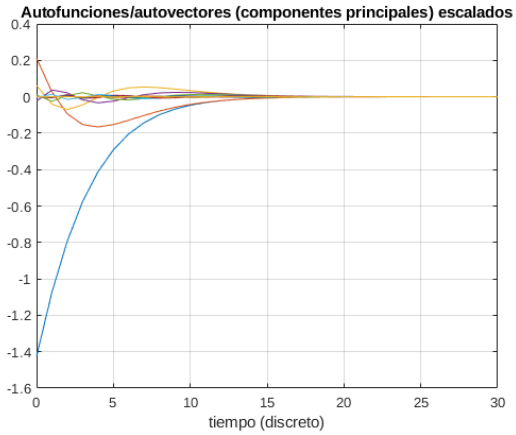
Componentes principales (Karhunen-Loève) que generan la respuesta a identificar

```
[V,D]=eig(K);
[Ds,Index]=sort(diag(D+1e-9),'descend');
plot(sqrt(Ds)), grid on, title("Desv. típica comp. principales"),
xlabel("Núm. componentes"), axis tight
```

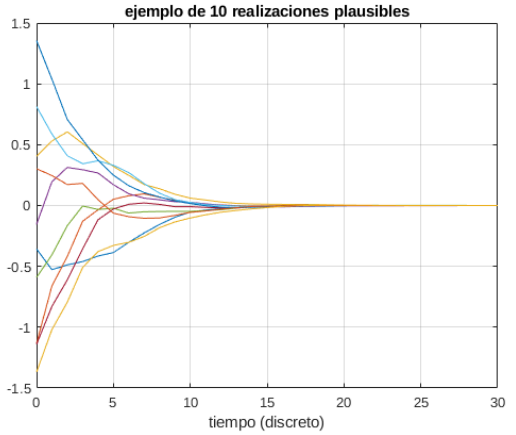


Grosso modo, si supiera ≈ 5 parámetros de la resp. impulsional podría interpolar toda la respuesta... Esa es la regularización... estamos estimando aprox. cinco "parámetros efectivos".

```
V2=V(:,Index);DT2=sqrt(diag(Ds));
%plot(ejeTh,V2(:,1:5)), grid on, xlabel("tiempo (discreto)")
%title("Autofunciones/autovectores (componentes principales) NO escalados")
plot(ejeTh,V2(:,1:end)*DT2(1:end,1:end)), grid on
xlabel("tiempo (discreto)")
title("Autofunciones/autovectores (componentes principales) escalados")
```




```
plot(ejeTh,V2(:,1:end)*DT2(1:end,1:end)*randn(orden,10)), xlabel("tiempo (discreto)")
grid on, title("ejemplo de 10 realizaciones plausibles")
```



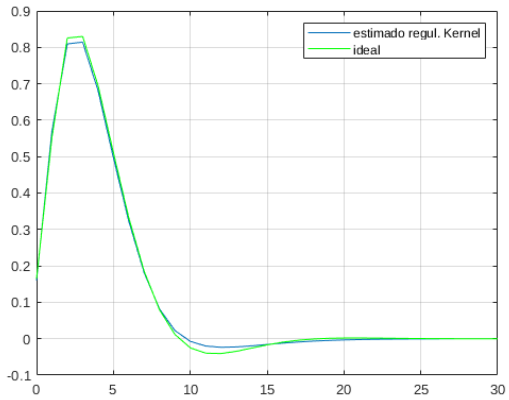
Regresión regularizada final

Hagamos, pues, la regresión con los (hiper)parámetros óptimos encontrados:

```
th2b=K*T'*((dtruido^2*eye(N)+T*K*T')\y);
varthbc=K-K*T'*((dtruido^2*eye(N)+T*K*T')\T*K);
dtthbc=sqrt(diag(varthbc));
```

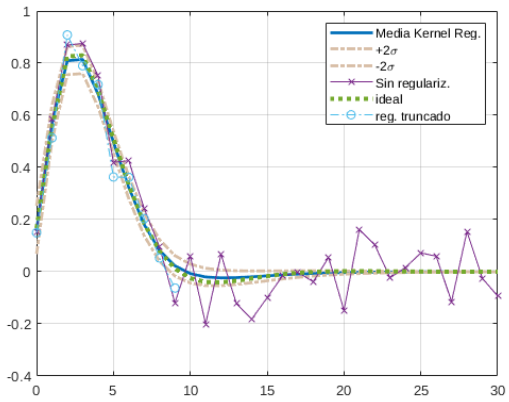
Y representemos los resultados

```
plot(ejeTh,th2b), hold on
plot(ejeTh,thperfecto,'g')
hold off, grid on, legend("estimado regul. Kernel","ideal")
```

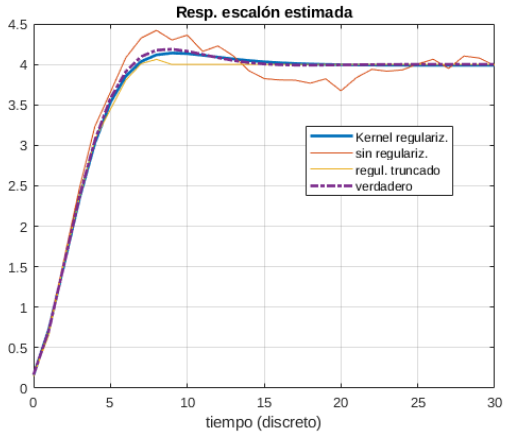


Con intervalos de confianza y comparando con otras opciones:

```
plotIncierto(ejeTh,th2b,dtthbc)
hold on
plot(ejeTh,th_noreg,'Marker','x')
plot(ejeTh,thperfecto,':','LineWidth',3)
plot(0:(orden_trunc-1),th_reg,'-.','Marker','o')
hold off
legend('Media Kernel Reg.','+2\sigma','-2\sigma','Sin regulariz.','ideal','reg. truncado')
```




```
cst=cumsum(th_reg);
plot(ejeTh, cumsum(th2b),'LineWidth',2)
hold on
plot(ejeTh, cumsum(th_noreg), [0:(orden_trunc-1) 30],[cst;cst(end)])
plot(ejeTh,cumsum(thperfecto),'-.','LineWidth',2)
hold off
legend('Kernel regulariz.','sin regulariz.','regul. truncado','verdadero','Location',"best"), g
title("Resp. escalón estimada"),xlabel("tiempo (discreto)")
```



Resumamos los ajustes sobre datos de validación:

```
error_cuadr_noreg %sin regularizar
```

```
error_cuadr_noreg = 18.8443
```

```
minEV %con Kernel regularisation
```

```
minEV = 7.9825
```

```
mejor_err_val %con truncación
```

```
mejor_err_val = 9.8047
```

Conclusiones

Se ha comparado la regularización con distintos Kernel que indican la estructura (covarianza) del prior sobre los elementos de la respuesta impulsional de un sistema a ser identificada. Se ha comparado con otras opciones de regularización (truncación).

Apéndice: funciones auxiliares

```
function K=Kernel(X1,X2,hyperparams,kappa)
n1=length(X1);n2=length(X2);
K=zeros(n1,n2);
for i=1:n1
    for j=1:n2
        K(i,j)=kappa(X1(i),X2(j),hyperparams);
    end
end
end


function k=kappaG(x1,x2,hyperparams)
```

```
M=hyperparams.desvt^2;
tau=hyperparams.tau;
k=exp(-(x1-x2)^2/2/tau^2)*M; %filtro orden infinito "square exponential", estacionario
end

function k=kappaW(x1,x2,hyperparams)
M=hyperparams.desvt^2;
tau=hyperparams.tau;
s=exp((-x1)/tau);t=exp((-x2)/tau);
k=min(s,t)*M; %Wiener Process, NO estacionario
end

function k=kappaIW(x1,x2,hyperparams)
M=hyperparams.desvt^2;
tau=hyperparams.tau;
x1b=exp((-x1)/tau);x2b=exp((-x2)/tau);
s=min(x1b,x2b); t=max(x1b,x2b);
k=s^2/2*(t-s/3)*M; %Integrated Wiener Process, NO estacionario
end



function plotIncierto(X,Y,DesvTip)
plot(X,Y,'LineWidth',2)
hold on
plot(X,Y+2*DesvTip,'-.','Color',[.85 .75 .65],'LineWidth',2)
plot(X,Y-2*DesvTip,'-.','Color',[.85 .75 .65],'LineWidth',2)
hold off, grid on
end
```