Filtro de Kalman EKF, UKF y **PF**: ejemplo péndulo

Presentación en vídeo: http://personales.upv.es/asala/YT/V/pfml.html

Este código ejecutó correctamente en Matlab ***R2019a***

Objetivo: comprender la implementación del filtro de partículas en la Control System Toolbox (con la mayoría de opciones por defecto). Comparar con EKF/UKF en un modelo no lineal (péndulo).

1.- Modelado

Modelo continuo de péndulo con dos puntos de equilibrio (estable $x_1=0$, inestable $x_1=\pi$). Las ecuaciones de estado y salida son:

```
%ecuación de estado en tiempo continuo
dxdt=@(x,u) [x(2); ... %deriv. posicion = velocidad
             -sin(x(1))-0.2*x(2)-0.25*x(2)^3+u];%deriv. velocidad = aceleracion
global EcSalida
EcSalida=@(x,u) x(1); %medimos la posición
```

Discretizamos la ec. de estado:

```
Ts=0.45;
global EcEstadoDiscreto
%EcEstadoDiscreto=@(x,u) x+Ts*dxdt(x,u);  %Euler
EcEstadoDiscreto=@(x,u) x+Ts*dxdt(x+Ts/2*dxdt(x,u),u); %Midpoint integration, ZOH u
```

La ecuación de salida, evidentemente, no se discretiza porque no hay dinámica.

1.1- Linealización

Obtengamos jacobianos y modelos linealizados mediante Symbolic toolbox:

```
x=sym('x',[2,1]);u=sym('u');
EcEstado_simbolico=EcEstadoDiscreto(x,u)
```

```
EcEstado_simbolico =
```

$$\begin{pmatrix} -\frac{81\,x_2^3}{3200}+\frac{1719\,x_2}{4000}+\frac{81\,u}{800}+x_1-\frac{81\sin(x_1)}{800} \\ \frac{1719\,u}{4000}+\frac{18281\,x_2}{20000}-\frac{9\sin\left(x_1+\frac{9\,x_2}{40}\right)}{20}+\frac{81\sin(x_1)}{4000}+\frac{81\,x_2^3}{16000}-\frac{9\left(-\frac{9\,x_2^3}{160}+\frac{191\,x_2}{200}+\frac{9\,u}{40}-\frac{9\sin(x_1)}{40}\right)^3}{80} \end{pmatrix}$$

```
LinealizacionA_symbolic=simplify(jacobian(EcEstado_simbolico,x));
LinealizacionA_Compilada=matlabFunction(LinealizacionA_symbolic,'Vars',{x,u});
LinealizacionB=(jacobian(EcEstado_simbolico,u)) %no es realmente necesaria
```

```
LinealizacionB =
```

$$\begin{pmatrix} \frac{81}{800} \\ \frac{1719}{4000} - \frac{243\left(-\frac{9x_2^3}{160}+\frac{191x_2}{200}+\frac{9u}{40}-\frac{9\sin(x_1)}{40}\right)^2}{3200} \end{pmatrix}$$

```
LinealizacionC=eval(jacobian(EcSalida(x,u),x)) %no depende de x, hacemos eval y punto,
```

```
LinealizacionC = 1×2
     1     0
```

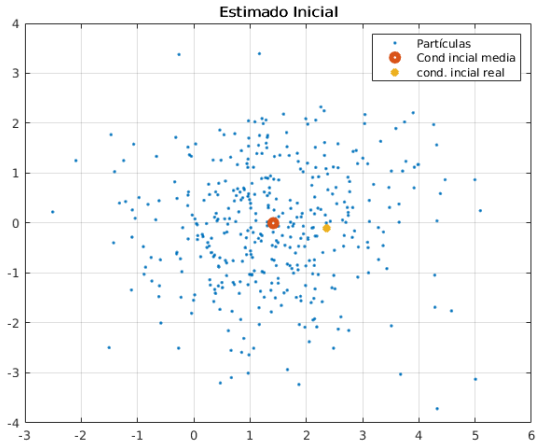
```
LinealizacionD=eval(jacobian(EcSalida(x,u),u))%no depende de x, hacemos eval y punto, s
```

```
LinealizacionD = 0
```

2.-Simulación del filtro de Kalman extendido, unscented, particle filter

```
%tipoobservador='Extended';
%tipoobservador='Unscented';
tipoobservador='Particle';
if(strcmp(tipoobservador,'Extended'))
    observador= extendedKalmanFilter(EcEstadoDiscreto,EcSalida);
    observador.StateTransitionJacobianFcn=LinealizacionA_Compilada;
    observador.MeasurementJacobianFcn=@(x,u) LinealizacionC;
elseif(strcmp(tipoobservador,'Unscented'))
    observador= unscentedKalmanFilter(EcEstadoDiscreto,EcSalida);
    observador.Alpha=1;observador.Kappa=3;
else %El filtro de partículas necesita simular "todas" las partículas:
    observador=particleFilter(@PF_simula,@PF_probabilidad_de_medida);
end

condiniobserv=[0.45*pi;0];
xreal=[0.75*pi;-.1]; %condiciones iniciales proceso real

if(strcmp(tipoobservador,'Particle'))
    rng('shuffle')
    initialize(observador, 400, condiniobserv, 1.25^2*eye(2));
%    dibujemos las partículas iniciales en el caso particle filter
    plot(observador.Particles(1,:),observador.Particles(2,:),'.')
    hold on
    plot(condiniobserv(1),condiniobserv(2),'o','LineWidth',4)
    plot(xreal(1),xreal(2),'x','LineWidth',4)
    hold off, grid on
    legend('Partículas','Cond incial media','cond. incial real')
    title('Estimado Inicial')
else
    %parámetros de varianza de proceso, medida y estimado inicial:
    observador.MeasurementNoise=0.08^2;
    observador.ProcessNoise=diag([0.001 0.02].^2);
```

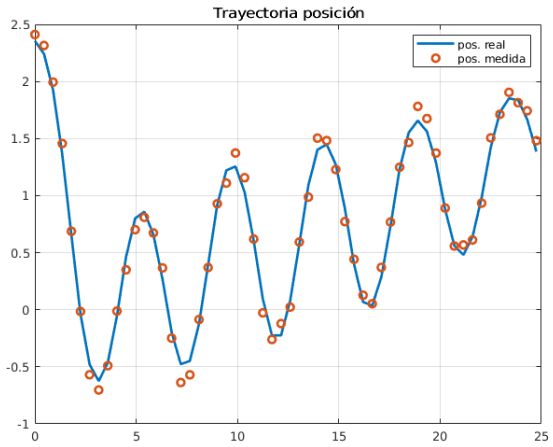
```
    observador.StateCovariance=eye(2)*1.25^2;
    observador.State=condiniobserv; %condiciones iniciales observador
end
```



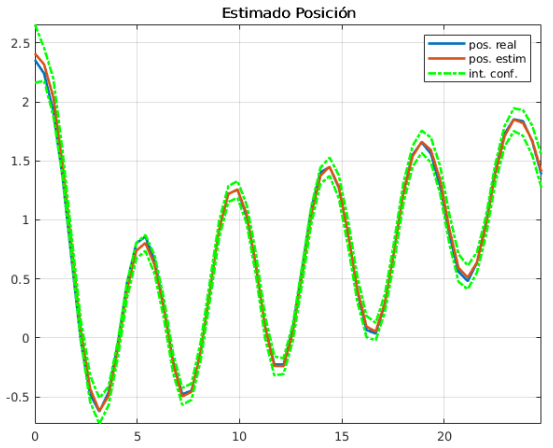

```
rng(1234) %para poder comparar con distintos parámetros la misma serie de datos
Tiempos=0:Ts:25;
uu=-0.75*sign(sin(1.4*Tiempos))+0.04*Tiempos;
N=length(Tiempos);
gxr=zeros(N,2);grm=zeros(N,1);grstd=zeros(N,2);
ruidosproceso=diag([.001;.02])*randn(2,N);ruidosmedida=0.08*randn(1,N);
for k=1:N

    medida=EcSalida(xreal,uu(k))+ruidosmedida(k);%Lectura de sensor con ruido real
    xestim(k,:)=correct(observador,medida,uu(k)); %corrector

    grstd(k,:)=sqrt(diag(observador.StateCovariance))';
    grm(k,:)=medida;  grx(k,:)=xreal';


    if(k<N)
        xreal=EcEstadoDiscreto(xreal,uu(k))+ruidosproceso(:,k); %simulación del estado
        predict(observador,uu(k)); %estimación en bucle abierto de observador para sigu
    end
end
norm(grx-xestim,'fro')
```

```
ans = 0.5132
```

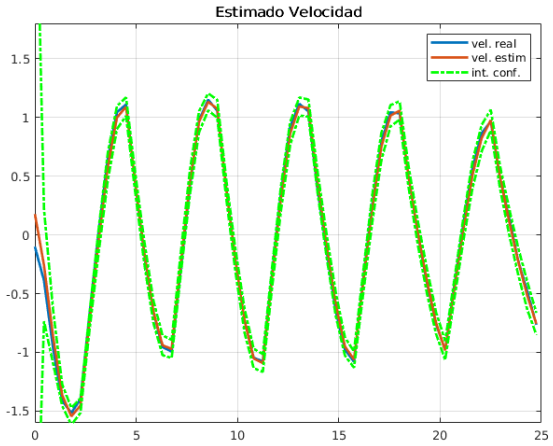
```
plot(Tiempos,grx(:,1),Tiempos,grm,'o','LineWidth',2)
legend('pos. real','pos. medida'), grid on
title('Trayectoria posición')
```



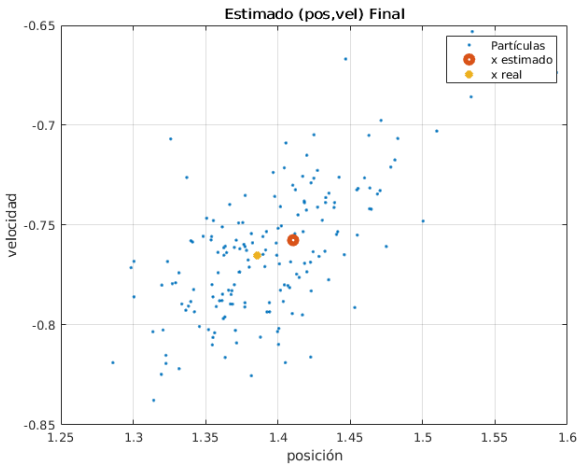

```
plot(Tiempos,[grx(:,1) xestim(:,1)],Tiempos, [xestim(:,1)-3*grstd(:,1) xestim(:,1)+3*gr
legend('pos. real','pos. estim','int. conf.'), grid on, axis tight
title('Estimado Posición')
```




```
plot(Tiempos,[grx(:,2) xestim(:,2)],Tiempos, [xestim(:,2)-3*grstd(:,2) xestim(:,2)+3*gr
legend('vel. real','vel. estim','int. conf.'), grid on, ylim([-1.6 1.8])
title('Estimado Velocidad')
```



```
if(strcmp(tipoobservador,'Particle'))
%    dibujemos, por ejemplo, las partículas finales en el caso particle filter
    plot(observador.Particles(1,:),observador.Particles(2,:),'.')
    hold on
    plot(xestim(N,1),xestim(N,2),'o','LineWidth',4)
    plot(xreal(1),xreal(2),'x','LineWidth',4)
    hold off, grid on
    legend('Partículas','x estimado','x real')
    title('Estimado (pos,vel) Final'), xlabel('posición'), ylabel('velocidad')
end
```




```
function newparticles=PF_simula(oldparticles,u)
    global EcEstadoDiscreto
    [n,npartic]=size(oldparticles);
    newparticles=zeros(n,npartic);
    for j=1:npartic
        newparticles(:,j)=EcEstadoDiscreto(oldparticles(:,j),u)...
            +diag([.001;.02])*randn(2,1); %añado ruido proceso al simular
    end
end

function p=PF_probabilidad_de_medida(particles,medida,u)
    global EcSalida
    npartic=size(particles,2);
    p=zeros(1,npartic);
    sigm=0.08;%desv.tip.ruido medida
    for j=1:npartic
        particle_output=EcSalida(particles(:,j),u);
        normalizado=(medida-particle_output)/sigm;
        p(j)=normpdf(normalizado);
    end
end
```