Proceso de medición idealizado (promedio de tiempo finito); filtro de Kalman de tiempo continuo vs. tiempo discreto, ejemplo Matlab

© 2025, Antonio Sala Piqueras, Universitat Politecnica de Valencia, Spain. Todos los derechos reservados.

Presentaciones en vídeo:

https://personales.upv.es/asala/YT/V/medabs.html [discusión genérica sobre el proceso de medida, notas no Matlab]

https://personales.upv.es/asala/YT/V/medex1.html [preliminares, discretización y simulación proceso ejemplo]

https://personales.upv.es/asala/YT/V/medex2.html [filtros de Kalman discretos y continuos, comparación, varianza de diferencias finitas]

Objetivo: comprender el comportamiento de "medición a bajo nivel", comparando un filtro Kalman de tiempo continuo con uno de tiempo discreto asumiendo que la medición se produce como $y_{meas} = \frac{1}{T_s}\int_0^{T_s}(y(\tau) + measurementnoise)\, d\tau$ siendo"measurement noise" ruido blanco de medida, via ejemplo numérico.

Tabla de Contenidos

Proceso del que se toman medidas....1
Dispositivo de medición (idealizado)....2
Simulación del proceso estocástico....3
Estimación de la posición mediante integral en un intervalo....4
Comparación con filtro de Kalman en tiempo discreto....5
Convergencia al filtro de Kalman en tiempo continuo....5
Gráfica según período de muestreo....6

Proceso del que se toman medidas

Masa-muelle-amortiguador, mediremos posición.

```
wn=0.25; %natural frequency
dampingcoeff=.7071;
Areduced=[0 1;-wn^2 -2*dampingcoeff*wn];
Greduced=[0;1];
eig(Areduced)
```

```
ans = 2×1 complex
  -0.1768 + 0.1768i
```

```
  -0.1768 - 0.1768i
```

```
Cr=[1 0];
sys_reduced=ss(Areduced,Greduced,Cr,0);
zpk(sys_reduced)
```

```
ans =

            1
  ----------------------
  (s^2 + 0.3535s + 0.0625)

Continuous-time zero/pole/gain model.
Model Properties
```

Dispositivo de medición (idealizado)

La medida la asumiremos como $\frac{1}{T_s}\int_0^{T_s} y(\tau)d\tau$... contaminada por ruido...

Es el filtro óptimo si se quiere estimar una constante con cero información a priori... Pero si y no es constante, un tiempo largo la hará más "borrosa".

En teoría, reiniciamos el integrador después de cada medición pero, bueno, medir su valor con "error cero" será lo mismo, ya que los incrementos se calcularán de manera trivial:

$$\frac{1}{T_s}\int_a^{a+T_s} y(\tau)d\tau = \frac{1}{T_s}\cdot\left(\int_0^{a+T_s} y(\tau)d\tau - \int_0^{a} y(\tau)d\tau\right) = \frac{1}{T_s}\cdot(q(a+T_s)-q(a))$$

siendo q el estado del integrador y a un instante arbitrario.

Pondremos un polo en $-\epsilon$ en vez del integrador puro, para tener varianza finita en ciertas matrices... pero básicamente es un integrador a no ser que el período de muestreo sea extremadamente grande...

```
epsi=2e-5;
A=[-epsi 1 0;zeros(2,1) Areduced];
C=[1 0 0]; %measurement of the "integral".
G=[0 1;Greduced zeros(2,1)]; %noise in integral added as input.
sys_with_integrating_sensor=ss(A,G,C,0); %AUGMENTED process
```

Parámetros de densidad espectral de potencia de ruidos blancos de proceso y de medida:

```
W_ad_sensor=4e-3; %measurement noise to integrator
W_process=1; %process noise, causing random acceleration.
W=diag([W_process W_ad_sensor]); %continuous-time PSD matrix.
```

Discretización del proceso aumentado:

```
Ts=0.001;
Ad=expm(A*Ts);
eig(Ad)'
```

```
ans = 1×3 complex
   1.0000 + 0.0000i   0.9998 - 0.0002i   0.9998 + 0.0002i
```

```
syms t T real
Bdsym=simplify(int(expm(A*t)*G*W*G'*expm(A'*t),0,T));
format long
Wd=eval(subs(Bdsym,T,Ts)) %There may be correlation process-
measurement noise...
```

```
Wd = 3×3
10^-3 ×
   0.003999099135399   0.000000000142109   0.000000166611613
   0.000000000142109   0.000000333244543   0.000499823251095
   0.000000166611613   0.000499823251095   0.999646512492847
```

Euler-Maruyama, aproximación varianza para $T_s \to 0$:

```
G*W*G'*Ts %variance for noise simulation at small Ts.
```

```
ans = 3×3
10^-3 ×
   0.004000000000000                   0                   0
                   0                   0                   0
                   0                   0   1.000000000000000
```

Covarianza estacionaria (por eso hay que resetear el integrador en la electrónica "práctica"):

```
format short
P=lyap(A,G*W*G') %would be infinity for pure integrator...
```

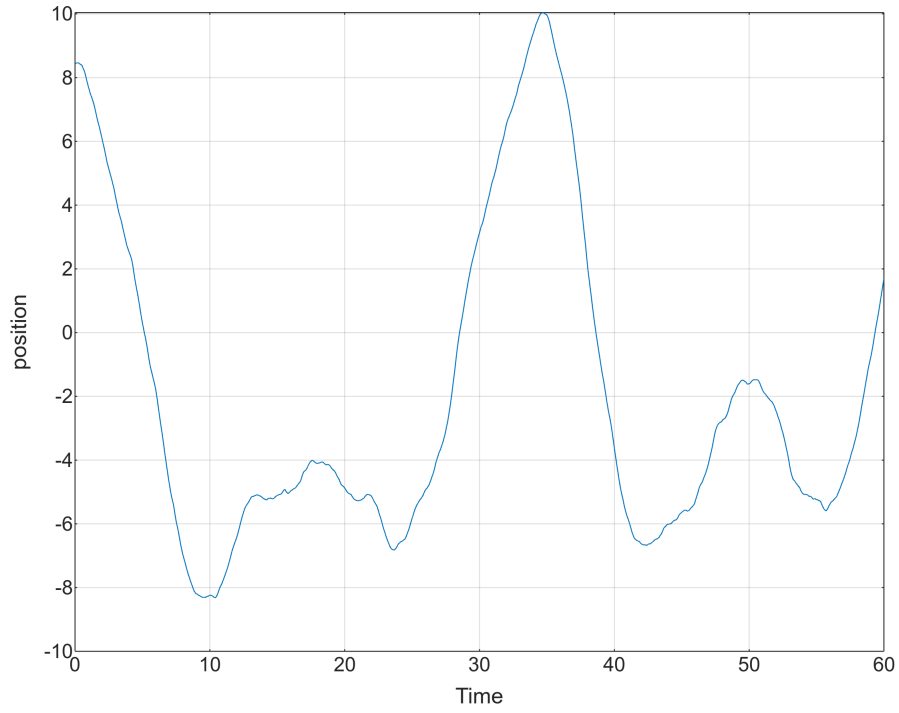
```
P = 3×3
10^6 ×
    6.3997    0.0001   -0.0000
    0.0001    0.0000         0
   -0.0000         0    0.0000
```

```
P(2:3,2:3) %physical (position, speed) covariance
```

```
ans = 2×2
   22.6276         0
         0    1.4142
```

Simulación del proceso estocástico

```
Nsim=round(60/Ts);
w=mvnrnd([0,0,0],Wd,Nsim);
Trg=(0:(Nsim-1))*Ts;
sysd=ss(Ad,eye(3),[0 1 0],0,Ts); %simulate to only "position", not
"position integral"!
y=lsim(sysd,w,Trg,mvnrnd([0;0;0],P));
plot(Trg,y), grid on, xlabel("Time"), ylabel("position")
```



Estimación de la posición mediante integral en un intervalo

Covarianza estacionaria

$$cov(x(t+T_s), x(t)) = e^{A \cdot T_s} P$$

Un vector $[x(0); x(Ts)]$ tendrá covarianza:

```
bigP=[P P*Ad'; Ad*P P]
```

```
bigP = 6×6
10^6 ×
    6.3997    0.0001   -0.0000    6.3997    0.0001   -0.0000
    0.0001    0.0000         0    0.0001    0.0000   -0.0000
   -0.0000         0    0.0000   -0.0000    0.0000    0.0000
    6.3997    0.0001   -0.0000    6.3997    0.0001   -0.0000
    0.0001    0.0000    0.0000    0.0001    0.0000         0
   -0.0000   -0.0000    0.0000   -0.0000         0    0.0000
```

El filtro óptimo para constante $\frac{1}{T_s}\int_0^{T_s} pos(\tau)\, d\tau$ lo obtendremos como una "diferencia finita" de los estados del integrador.

```
FinDiff=1/Ts*[-1 1] %naive finite-difference
```

```
FinDiff = 1×2
       -1000        1000
```

Su varianza, a partir de la varianza de los estados, será:

```
MM=[-1/Ts 0 0 1/Ts -1 0 ];
```

```
MM*bigP*MM'
```

```
ans = 4.0022
```

Comparación con filtro de Kalman en tiempo discreto

Suponemos que "medimos" el estado del integrador con una varianza básicamente "cero" cada T_s:

```
sysd=ss(Ad,eye(3),[1 0 0],0,Ts);
[~,L,~,~,z2disc]=kalman(sysd,Wd,1e-12);
```

La varianza del error de estimación a posteriori resulta:

```
format long
z2disc
```

```
z2disc = 3×3
   0.000000000001000   0.000000000005263   0.000000000013848
   0.000000000005263   0.021076077975833   0.055537607600379
   0.000000000013848   0.055537607600379   0.313646863524495
```

```
trace(z2disc)
```

```
ans =
   0.334722941501328
```

```
format short
```

La ganancia del filtro de Kalman DISCRETO es:

```
L
```

```
L = 3×1
    1.0053
    5.2771
   13.8431
```

Convergencia al filtro de Kalman en tiempo continuo

```
[~,Lc3,z2x]=kalman(sys_with_integrating_sensor,W,1e-12);
z2x
```

```
z2x = 3×3
    0.0000    0.0000    0.0000
    0.0000    0.0211    0.0555
    0.0000    0.0555    0.3137
```

```
Lc3 %idealized case, clean measurements do NOT exist...
```

```
Lc3 = 3×1
10^5 ×
    0.6325
    3.3331
    8.7821
```

Estamos multiplicando un resultado "infinitesimalmente pequeño" del integrador por una ganancia del observador "infinitamente grande"... Este no es el camino a seguir: usemos el sistema NO aumentado.

El filtro de Kalman en tiempo continuo con sensor de posición contaminado con, por supuesto, ruido blanco con la misma PSD, que ingresa a otra "electrónica" proporcionada por el KF en lugar del "integrador promediador" se calcula con:

```
[~,Lc2,z2reduced]=kalman(sys_reduced,W_process,W_ad_sensor);
Lc2
```

```
Lc2 = 2×1
    5.2699
   13.8858
```

```
format long
z2reduced
```

```
z2reduced = 2×2
   0.021079540020250   0.055543375933163
   0.055543375933163   0.313662035773152
```

```
trace(z2reduced)
```

```
ans =
   0.334741575793402
```

```
format short
```

Gráfica según período de muestreo

Copiemos el código y demos valores a T_s en un rango.

```
function [VVnaive,VVKal]=PredVariance(Ts,Data)
  P=Data.P;
  Ad=expm(Data.A*Ts);Bd=P-Ad*P*Ad';Bd=(Bd+Bd')/2;
  bigP=[P P*Ad';Ad*P P];
  Gnaive=[-1/Ts 0 0 1/Ts -1 0];
  VVnaive=Gnaive*bigP*Gnaive';
  sysd=ss(Ad,eye(3),[1 0 0],0,Ts);
  [~,~,~,~,z2]=kalman(sysd,Bd,1e-12);
  VVKal=z2(2,2);
end
Data.P=P;Data.A=A;


function Tsbest=PlotThings(Data)
Tsrange=logspace(-5,3,100);
N=length(Tsrange);
VVnaiveplot=zeros(1,N);
```

```
VVKalmanplot=zeros(1,N);
for k=1:N
    [VVnaiveplot(k),VVKalmanplot(k)]=PredVariance(Tsrange(k),Data);
end
loglog(Tsrange,[1./Tsrange*4e-3;VVnaiveplot;
VVKalmanplot],LineWidth=2), grid on,
yline(Data.P(2,2),':',LineWidth=2)
ylim([1e-3 Data.P(2,2)*2])
xlabel("log Ts"),ylabel("log Prediction error variance (position)")
[~,idxbest]=min(VVnaiveplot);
Tsbest=Tsrange(idxbest);
end

Tsbest=PlotThings(Data)
```

```
Warning: The [G 0;H I]*[Qn Nn;Nn' Rn]*[G 0;H I]' matrix should be positive semi-definite. Type
"help kalman" for more information.
Warning: The [G 0;H I]*[Qn Nn;Nn' Rn]*[G 0;H I]' matrix should be positive semi-definite. Type
"help kalman" for more information.
Warning: The [G 0;H I]*[Qn Nn;Nn' Rn]*[G 0;H I]' matrix should be positive semi-definite. Type
"help kalman" for more information.
Tsbest = 0.1918
```

```
yline(z2reduced(1,1), 'g',LineWidth=2) %
legend("ideal finite diff variance if constant","Finite Diff actual
variance","Kalman Discr. order 3","prior position variance","Kalman
Continous Time",Location="best")
hold off
```

