Finite-time averaging idealized measurement process; continuous-time vs discrete-time Kalman filter: Matlab example

Presentacions in video:

https://personales.upv.es/asala/YT/V/medabsEN.html [generic discussion, abstract interpretation, no Matlab code]

https://personales.upv.es/asala/YT/V/medex1EN.html [preliminaries, discretization and simulation of example stochastic process]

https://personales.upv.es/asala/YT/V/medex2EN.html [continuous-time and discrete-time Kalman filters, comparison, finite difference estimation variance]

Objective: understanding the "low-level sensing" behaviour, comparing a continuous-time Kalman Filter with a discrete-time one assuming measurement comes as $y_{meas} = \frac{1}{T_s}\int_0^{T_s}(y(\tau) + measurementnoise)\, d\tau$ with some PSD for the measurement noise, via a numeric example.

Table of Contents

Base process to measure 1
Measurement device (idealized) 2
Simulation (realization of stochastic process) 4
Position estimation via averaging (finite-time integral) 4
Comparison with discrete-time Kalman filter 5
Converging to the continuous-time Kalman filter 5
Optimal sampling period (for non-Kalman approach) 6

Base process to measure

We will have a 2nd order continuous-time process.

```
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
```

Measurement device (idealized)

It will connect to a device, say an RC so that the capacitor is charged during some conversion time, etc. Say, the measurement will be $\frac{1}{T_s}\int_0^{T_s} y(\tau)d\tau$... but it will add some noise... It is the "optimal filter" for constant y and zero prior information (infinite covariance); if y is not constant, the measurement will "blur" it.

In theory, we reset the integrator after each measurement but, well, measuring its value with "zero error" will be the same, as increments will be trivially computed:

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

being $q(t)$ the state of the integrator and a any arbitrary time instant when data acquisition begins.

And, well, we may have some pole $-\epsilon$ instead of pure integrator... This will help us in getting finite steady-state variance and $1/(s+\epsilon)$ is basically an integrator unless sampling frequency is extremely small.

```
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
```

Noise input PSD:

```
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.
```

Discretization of the augmented process:

```
Ts=0.002;
Ad=expm(A*Ts);
```

```
Ad = 3×3
    1.0000    0.0020    0.0000
         0    1.0000    0.0020
         0   -0.0001    0.9993
```

```
eig(Ad)'
```

```
ans = 1×3 complex
   1.0000 + 0.0000i   0.9996 - 0.0004i   0.9996 + 0.0004i
```

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

```
Wd = 3×3
   0.000008000992239   0.000000000002032   0.000000001332388
   0.000000000002032   0.000000002665256   0.000001998586216
   0.000000001332388   0.000001998586216   0.001998586299899
```

Euler-like (Euler-Maruyama) approximation for fast samping rate (small $T_s \to 0$):

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

```
ans = 3×3
   0.000008000000000                   0                   0
                   0                   0                   0
                   0                   0   0.002000000000000
```

Stationary covariance matrix of the states:

```
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
```

```
Pred=lyap(Areduced,Greduced*W_process*Greduced')
```

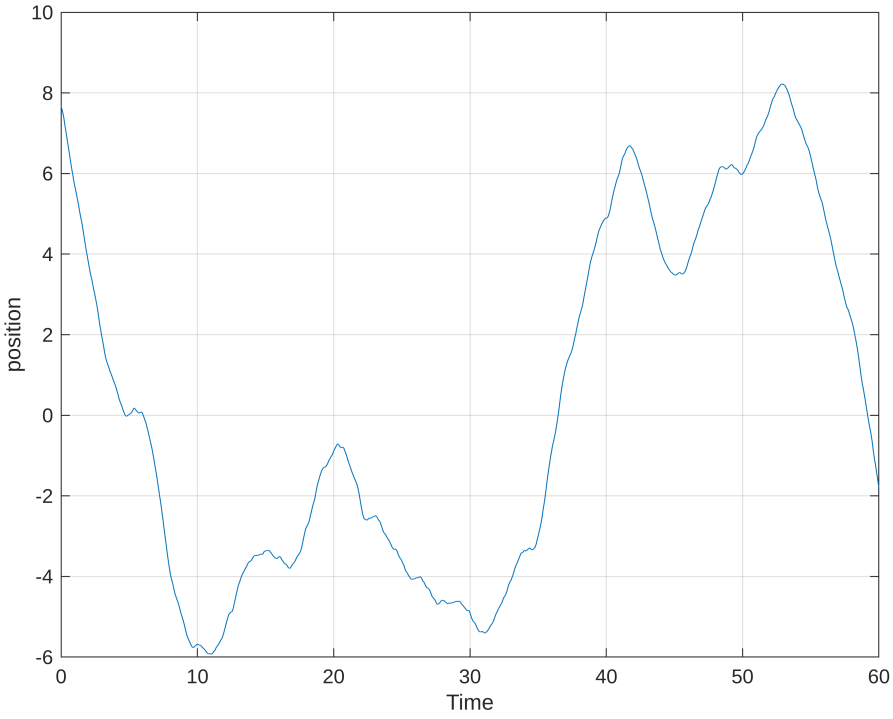
```
Pred = 2×2
   22.6276         0
```

```
         0    1.4142
```

Simulation (realization of stochastic process)

Let us simulate the system to see what it looks like:

```
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")
```



Position estimation via averaging (finite-time integral)

Stationary covariance between states at different times:

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

A vector $[x(0); x(Ts)]$ will have covariance:

```
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
```

If position were constant, this would be the optimal filter $\frac{1}{T_s}\int_0^{T_s} pos(\tau)\,d\tau$... We obtain it with a "finite difference" of the integrator state

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

```
FinDiff = 1×2
  -500   500
```

The actual variance of this estimator is:

```
MM=[-1/Ts 0 0 1/Ts -1 0 ];
MM*bigP*MM'
```

```
ans = 2.0028
```

Comparison with discrete-time Kalman filter

We assume we "measure" the integrator state with basically "zero" variance every Ts:

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

The posterior error variance of the discrete-time Kalman filter is:

```
format long
z2disc
```

```
z2disc = 3×3
   0.000000000001000   0.000000000005256   0.000000000013807
   0.000000000005256   0.021081532560776   0.055546856666116
   0.000000000013807   0.055546856666116   0.313671634407982
```

```
trace(z2disc)
```

```
ans =
   0.334753166969758
```

```
format short
```

The discrete-time Kalman filter gain is:

```
L
```

```
L = 3×1
    1.0105
    5.2834
   13.7963
```

Converging to the continuous-time Kalman filter

The continuous-time Kalman filter with the same info (3rd order system):

```
[~,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
```

We are multiplying an "infinitesimally small" result of the integrator by an "infinitely big" observer gain... This is not the way to go: let's use the NON-augmented sytem.

The continuous-time Kalman filter with position sensor contaminated with, of course, white noise with same PSD, entering to other "electronics" given by the KF instead of "averaging integrator":

```
[~,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
```

Optimal sampling period (for non-Kalman approach)

Let us build a function so we can explore different sampling rates, copying the above code:

```
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);
```

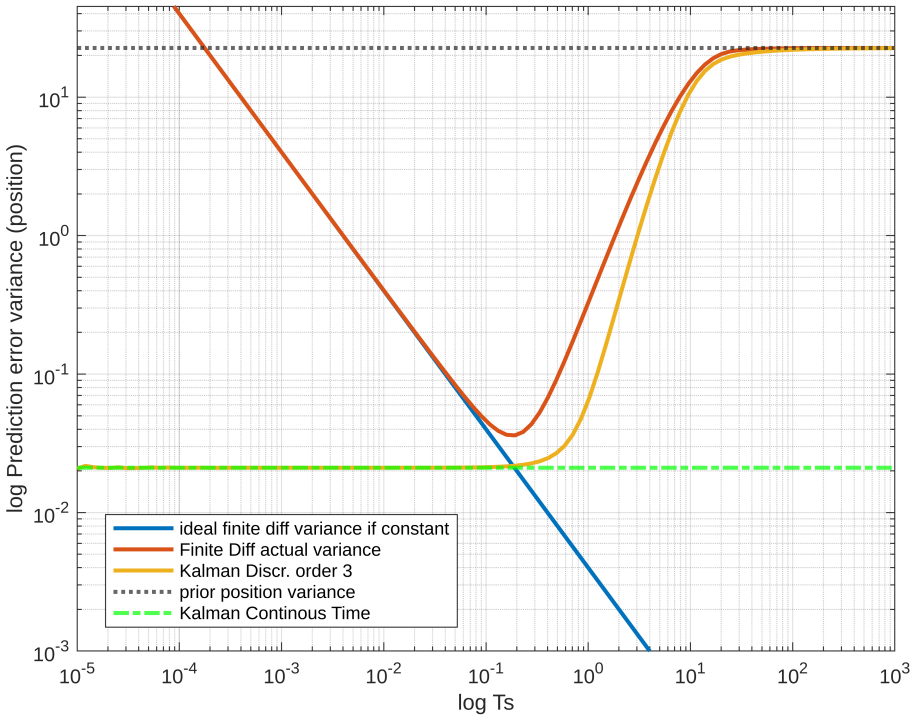
```
end
Data.P=P;Data.A=A;

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
```


log Prediction error variance (position)
log Ts
ideal finite diff variance if constant
Finite Diff actual variance
Kalman Discr. order 3
prior position variance
Kalman Continous Time