Finite-Diference speed estimation vs. Kalman filter, sampling period choice: Matlab example

Objective: assessing in formal terms if estimating speed by ·subtracting two close position measurements" (and dividing by sample rate) is a good option (with respect to the best one which would be a full-fledged Kalman filter). Assessing the influence of the sampling rate.

*Note that there exists a non-causal estimator (RTS smoother) that indeed beats Kalman filter using "future" samples. Omitted for brevity.

Presentations in video (English):

https://personales.upv.es/asala/YT/V/fdest1EN.html [motivation]

https://personales.upv.es/asala/YT/V/fdest2EN.html [naïve finite diference]

https://personales.upv.es/asala/YT/V/fdest3EN.html [optimal 2-sample estimate]

https://personales.upv.es/asala/YT/V/fdes4EN.html [Kalman filter]

Presentaciones en ESPAÑOL (Spanish):

https://personales.upv.es/asala/YT/V/fdest1.html [motivación]

https://personales.upv.es/asala/YT/V/fdest2.html [diferencias finitas naïve]

https://personales.upv.es/asala/YT/V/fdest3.html [estimador óptimo 2 muestras]

https://personales.upv.es/asala/YT/V/fdes4.html [filtro de Kalman]

Table of Contents

Base process to measure ... 1
Discretization ... 2
Simulation (realization of stochastic process) ... 3
Prediction of speed from 2 consecutive position measurements ... 5
Comparison with discrete-time Kalman filter ... 6
Which is the best sampling rate? (non-Kalman options) ... 6
*Caution note: computations are invalid for small sampling rates ... 8

Base process to measure

```
pol=.25;
A=[0 1;-pol^2 -2*pol]; G=[0;1];
eig(A)
```

```
ans = 2×1
   -0.2500
   -0.2500
```

```
C=[1 0];
sys=ss(A,G,C,0);
zpk(sys)
```

```
ans =

       1
  ----------
  (s+0.25)^2

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

```
W=1;
```

Stationary covariance matrix of the states:

```
P=lyap(A,G*W*G')
```

```
P = 2×2
    16     0
     0     1
```

Stationary covariance between states at different times:

$$cov(x(t+T_s), x(t)) = e^{AT_s}P$$

Discretization

1.) Discrete-time model to propagate the mean and variance of the Gaussian process

Mean equation: $\overline{x}_{k+1} = A_d\overline{x}_k$, Variance equation: $P_{k+1} = A_dPA_d^T + W_d$

```
Ts=0.025;
syms t real
format long
Wd=eval(int(expm(A*t)*G*W*G'*expm(A'*t),0,Ts))
```

```
Wd = 2×2
   0.000005159748502   0.000308618062654
   0.000308618062654   0.024689767496628
```

```
format short
Ad=expm(A*Ts)
```

```
Ad = 2×2
    1.0000    0.0248
   -0.0016    0.9876
```

```
eig(Ad)
```

```
ans = 2×1
```

```
0.9938
0.9938
```

2.) Discrete-time model for simulation of a given realization

The discrete model is $X_{k+1} = A_d x_k + w_k$, being w_k, with dimensions 2x1, a realization of a 2D normal distribution of zero mean and covariance W_d. In the continuous model there was a disturbance-input matrix G in $\dot{x} = Ax + Gw$, but that has already been integrated into the calculation of $W_d = \int_0^{T_s} e^{A\tau} GWG^T e^{A^T\tau} d\tau$, so it does not appear in the discretized model (but it is implicitly in the structure of W_d).

Stationary covariance, discrete

```
dlyap(Ad,Wd) %coincides with Continuous-time, of course
```

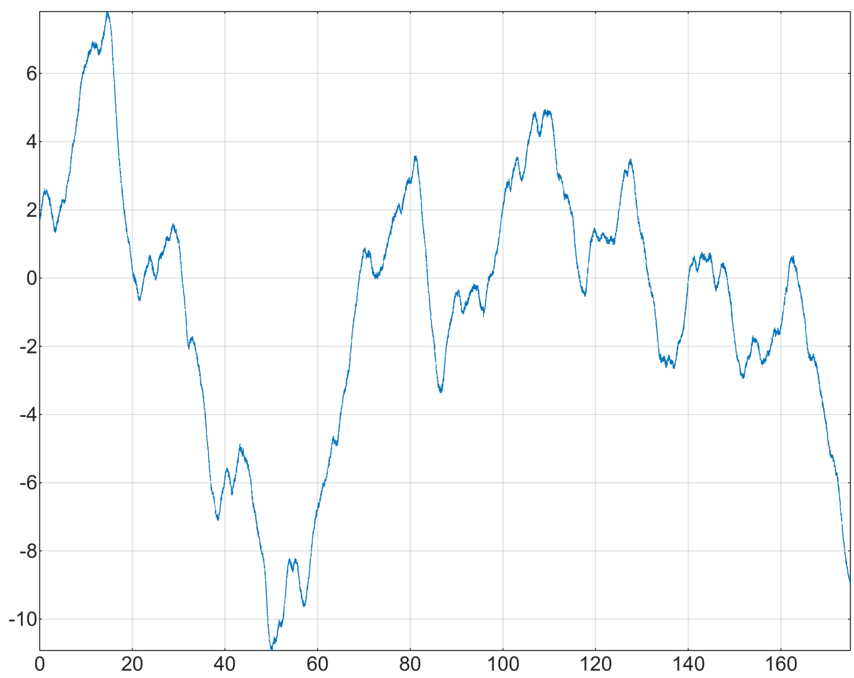
```
ans = 2×2
   16.0000    0.0000
    0.0000    1.0000
```

```
%Wd=P-Ad*P*Ad' also gets the correct result for stable systems
```

Simulation (realization of stochastic process)

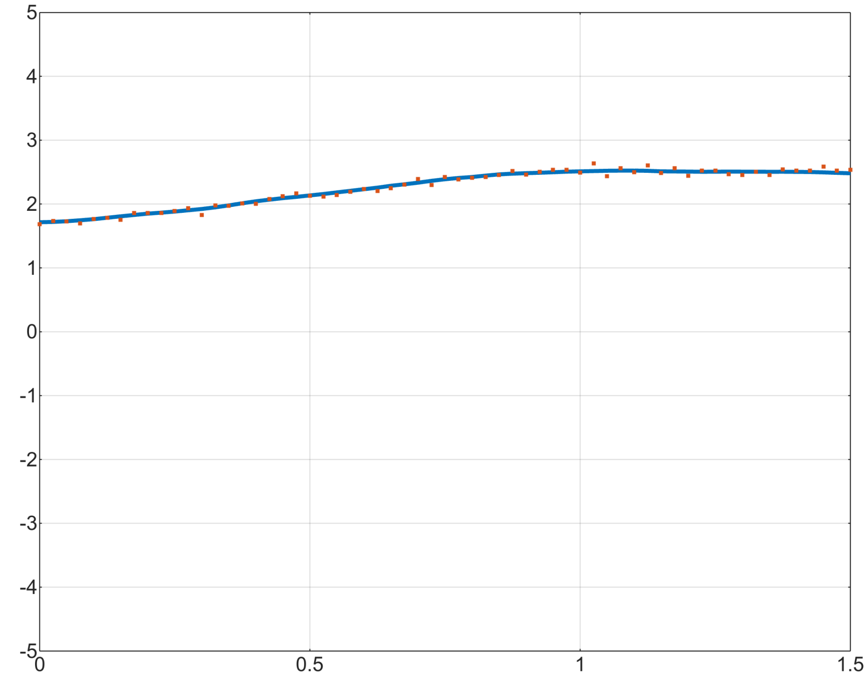
Let us simulate the system to see what it looks like:

```
V=0.002; %measurement noise variance
Nsim=7001;
w=mvnrnd([0,0],Wd,Nsim); %realization of the input process noise
Trg=(0:(Nsim-1))*Ts;
sysd=ss(Ad,eye(2),[1 0],0,Ts); %simulation of x_{k+1}=A·x_{k}+I·w_k
y=lsim(sysd,w,Trg,mvnrnd([0;0],P)); %random initial condition
yn=y+randn(Nsim,1)*sqrt(V); %realization of measurement noise
plot(Trg,yn), grid on, axis tight
```



Let us zoom in "one and a half seconds":

```
Nzoom=ceil(1.5/Ts+1);
plot(Trg(1:Nzoom),y(1:Nzoom),LineWidth=2)
hold on
plot(Trg(1:Nzoom),yn(1:Nzoom),'.'),
hold off
grid on, ylim([-5 5]),xlim([0 1.5])
```



Prediction of speed from 2 consecutive position measurements

Let us prepare "best prediction" of 2nd state at "Ts" given 1st state at "0" and "Ts".

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

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

```
bigP = 4×4
   16.0000         0   15.9997   -0.0248
         0    1.0000    0.0248    0.9876
   15.9997    0.0248   16.0000         0
   -0.0248    0.9876         0    1.0000
```

```
eig(bigP)'
```

```
ans = 1×4
    0.0000    0.0124    1.9879   31.9997
```

The last item, predicted from items 1 and 3 will have a "gain":

```
IndexPos=[1 3]; IndexV=[4];
GG=bigP(IndexV,IndexPos)*inv(bigP(IndexPos,IndexPos))
```

```
GG = 1×2
  -39.9171   39.9163
```

```
1/Ts*[-1 1] %naive filter
```

```
ans = 1×2
   -40    40
```

```
[Ve,De]=eig(bigP(IndexPos,IndexPos)) %Positions are very
correlated... Difference is almost nil:
```

```
Ve = 2×2
   -0.7071    0.7071
    0.7071    0.7071
De = 2×2
    0.0003         0
         0   31.9997
```

```
bigP(IndexV,IndexV) %prior variance
```

```
ans = 1
```

```
ResidueVar=bigP(IndexV,IndexV)-(GG)*bigP(IndexPos,IndexV) %posterior
```

```
ResidueVar = 0.0083
```

If we have "measurement noise" in my position samples, we get that the optimal estimation is:

```
GG=bigP(IndexV,IndexPos)*inv(bigP(IndexPos,IndexPos)+V*eye(2))
```

```
GG = 1×2
   -5.3751    5.3744
```

```
ResidueVar=bigP(IndexV,IndexV)-(GG)*bigP(IndexPos,IndexV)
```

```
ResidueVar = 0.8665
```

Comparison with discrete-time Kalman filter

```
[~,~,Z,~] = dlqe(Ad,eye(2),[1 0],Wd,V);
Z
```

```
Z = 2×2
    0.0007    0.0054
    0.0054    0.1017
```

```
VVKal=Z(2,2)
```

```
VVKal = 0.1017
```

For comparison, with $T_s = 0.025$, Kalman gets 0.1017 variance, naive finite difference gets 6.4 and optimal 2-position estimate gets 0.8665 residual variance.

```
[~,~,~,~,z2]=kalman(sysd,Wd,V);
z2
```

```
z2 = 2×2
    0.0007    0.0054
    0.0054    0.1017
```

Which is the best sampling rate? (non-Kalman options)

Let us build a function so we can explore different sampling rates, just repeating the code.

```
function [VV,GG,VVnaive,VVKal]=PredVariance(Ts,Data)
P=Data.P;
Ad=expm(Data.A*Ts);Bd=P-Ad*P*Ad';
V=Data.V;
bigP=[P P*Ad';Ad*P P];
IndexPos=[1 3]; IndexV=[4];
GG=bigP(IndexV,IndexPos)*inv(bigP(IndexPos,IndexPos)+V*eye(2));
ResidueVar=bigP(IndexV,IndexV)-(GG)*bigP(IndexPos,IndexV);
VV=ResidueVar;
Gnaive=[-1/Ts 0 1/Ts -1];
VVnaive=Gnaive*bigP*Gnaive'+2*V/Ts^2;
[~,~,Z,~] = dlqe(Ad,eye(2),[1 0],Bd,V);
VVKal=Z(2,2);
end
Data.P=P;Data.A=A;Data.V=V;
```

The function below will plot things (we'll use it twice):

```
function Tsbest=PlotThings(Data,lw)
Tsrange=logspace(-4,2.5,100);
N=length(Tsrange);
Vplot=zeros(1,N);
Vnaiveplot=zeros(1,N);
```

```
VKalmanplot=zeros(1,N);
for k=1:N

[Vplot(k),~,Vnaiveplot(k),VKalmanplot(k)]=PredVariance(Tsrange(k),Da
ta);
end
loglog(Tsrange,[Vplot;Vnaiveplot;VKalmanplot],LineWidth=lw), grid
on,
yline(Data.P(2,2),':')
ylim([0 1.5])
xlabel("log Ts"),ylabel("Prediction variance")
[~,idxbest]=min(Vplot);
Tsbest=Tsrange(idxbest);
end
```

Check performance for a range of sampling rates with "high measurement noise"

```
Tsbest1=PlotThings(Data,2)
```

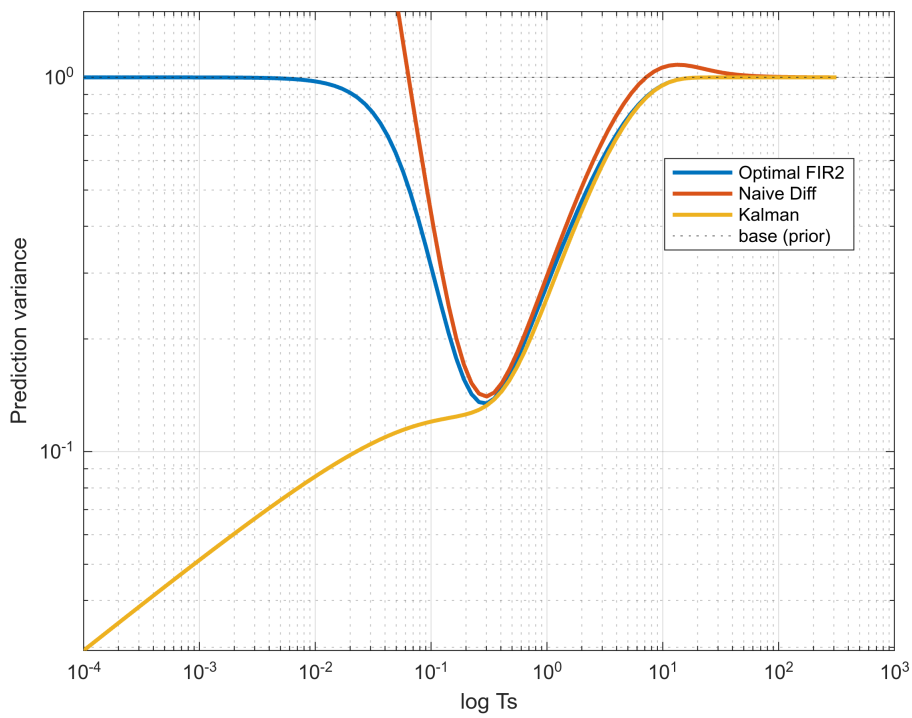
```
Tsbest1 = 0.3019
```

```
hold on
```

And do the same now with "lower measurement noise"

```
Data.V=Data.V/100; %Less noise?
%Tsbest=PlotThings(Data,1)
hold off
legend("Optimal FIR2","Naive Diff","Kalman","base (prior)","FIR
less noise", "Diff less noise", "Kalman less noise",Location="best")
```

```
Warning: Ignoring extra legend entries.
```




```
[VV,GG]=PredVariance(Tsbest,Data)
```

```
VV = 0.0307
GG = 1×2
  -14.8060   14.8040
```

```
1/Tsbest*[-1 1]
```

```
ans = 1×2
  -15.0234   15.0234
```

***Caution note: computations are invalid for small sampling rates**

IDEA 1: Kalman tends to zero? Well, if we throw "infinite coins" in an "infinitesimal time", we can instantly track the "prob. of heads" with zero error. That CANNOT happen. If I measure position with "finite variance" 10 trillion times in a nanosecond, then I would "instantly" get the mean position with zero error. That CANNOT happen.

IDEA 2: we assumed constant measurement noise variance... When sampling rate approaches A/D electronic time constants then other phenomena need to be considered... I mean, physically, there are continuos-time elements in the sensor subject to continuous-time noise and noise accumulation depends on "sensing/AD conversion time" and the probe's time constants, and anti-alias filters, etc.

There are TWO sampling rates to consider:

- Internal rate of the probe/ADconverter/sampler [sampling takes 1 microsecond if AD is rated at 1 MHz max sampling frequency]

- External rate of my motion control software, filter, whatever... which might be in the tens or hundreds of milliseconds.

We are assuming, say, that we are filming a video at 20 fps in a well-lit scene so each frame is exposed 1/500 s; Obviously, signal-to-noise ratio would degrade (higher "sensitivity ISO gain") if we wished to go past 500 fps: exposing 1/1000 s and doubling the photosensor gain will increase "measurement noise variance" by a factor of 2 (under some assumptions). These issues are NOT being considered here, so we will never be able to beat the "continuous-time" Kalman filter if things are modelled to such detail.