Derivative/Velocity info of a Gaussian Process: one-dimensional Matlab example

This code ran in Matlab **R2023b**

Presentations in video:

http://personales.upv.es/asala/YT/V/gpvel1EN.html

http://personales.upv.es/asala/YT/V/gpvel2EN.html

Objectives: we may have observations of the gradient (measurements of, say, position and speed in a physical system excited by noise) or we might wish to estimate such gradient. We'll discuss prediction given such observations.

Table of Contents

Problem statement

We have a GP with quadratic-exponential Kernel, and we wish to estimate its position and velocity, given position and velocity measurements with some measurement noise at several places.

Let us define the core kernel function of a stochastic process in $\mathbb{R}$ as:

```
sg=8; %length scale
M=4; %prior variance
kappa=@(h) M*exp(-h^2/sg^2); %stationary
K=@(x1,x2) kappa(x1-x2);

syms x1 x2 real
Ksym=K(x1,x2) %symbolic
```

```
Ksym =
```

$$4\,e^{-\frac{(x_1-x_2)^2}{64}}$$

```
syms h real
```

```
PSD=fourier(kappa(h))
```

PSD = $32\sqrt{\pi}\,e^{-16w^2}$

Variance and covariance of the estimated derivatives

Let us consider a stochastic process with covariance kernel $k(x,x')$.

- Let us consider covariance between the stochastic process $f(x)$ and its partial derivatives $\frac{\partial f}{\partial x_i}(x')$ at an arbitrary different point x'.

$$cov(f(x),\frac{\partial f}{\partial x_i}(x')) = \frac{\partial k(x,x')}{\partial x_i'}$$

$$cov(\frac{\partial f}{\partial x_i}(x), f(x')) = \frac{\partial k(x,x')}{\partial x_i}$$

In our particular one-dimensional case, we have:

Between $f(x_1)$ and $\frac{df}{dx}(x_2)$:

```
cov_f_df_sym=simplify(diff(Ksym,x2),100)
```

cov_f_df_sym =

$$\frac{e^{-\frac{(x_1-x_2)^2}{64}}(x_1-x_2)}{8}$$

```
cov_f_df=matlabFunction(cov_f_df_sym); %compiled for later use.
```

Between $f'(x_1)$ and $f(x_2)$ ... well, it will be the same just swapping letters...

```
simplify(diff(Ksym,x1),100)
```

ans =

$$-\frac{e^{-\frac{(x_1-x_2)^2}{64}}(x_1-x_2)}{8}$$

```
cov_df_f=matlabFunction(simplify(diff(Ksym,x1)));
```

- If we consider covariance between partial derivatives

$$cov\left[\frac{\partial f}{\partial x_i}(x), \frac{\partial f}{\partial x_j}(x')\right] = \frac{\partial^2 k}{\partial x_i \partial x'_j}(x, x')$$

In our particular case, autocovariance of $\frac{df}{dx}$:

```
cov_df_df=simplify(diff(Ksym,x1,x2),70)
```

```
cov_df_df =
```

$$\frac{e^{-\frac{(x_1-x_2)^2}{64}}\left(-x_1{}^2+2\,x_1\,x_2-x_2{}^2+32\right)}{256}$$

```
cov_df_df=matlabFunction(cov_df_df);
```

Particularisation to stationary processes: statistical properties of its (partial) derivatives

- regarding covariance of a stationary process and its first derivatives, if $k(x, x') = \kappa(x' - x)$, being $\kappa(h)$ a given autocovariance function (necessary, it must be an even function, i.e., $k(x, x') = \kappa(h) \equiv \kappa(-h) = k(x', x)$), depending only on the "difference" between points $h = x' - x$.

$$cov(f(x), \frac{\partial f}{\partial x_i}(x + h)) = -\frac{\partial \kappa}{\partial h_i}(-h) = \frac{\partial \kappa}{\partial h_i}(h)$$

$$cov(f(x + h), \frac{\partial f}{\partial x_i}(x)) = -\frac{\partial \kappa}{\partial h_i}(h)$$

- Regarding covariance between partial derivatives, we have

$$cov\left[\frac{\partial f}{\partial x_i}(x), \frac{\partial f}{\partial x_j}(x + h)\right] = -\frac{\partial^2 \kappa}{\partial h_i \partial h_j}(h)$$

Stationary one-dimensional process (partial derivatives not needed)

If $x \in \mathbb{R}$ and we change it to t (usual choice for "time"), partial derivatives get converted to plain ordinary derivatives, and h is understood as a time difference. We would then have:

$$cov(f(t_2), \frac{df}{dt}(t_1)) = -\frac{d\kappa}{dh}(t_2 - t_1) = \frac{d\kappa}{dh}(t_1 - t_2)$$

$$cov\left(\frac{df}{dt}(t_1), \frac{df}{dt}(t_2)\right) = -\frac{d^2\kappa}{dh^2}(t_1 - t_2)$$

```
fourier(kappa(h))
```

ans = $32\sqrt{\pi}\,e^{-16w^2}$

```
kposvel=simplify(diff(kappa(h)))
```

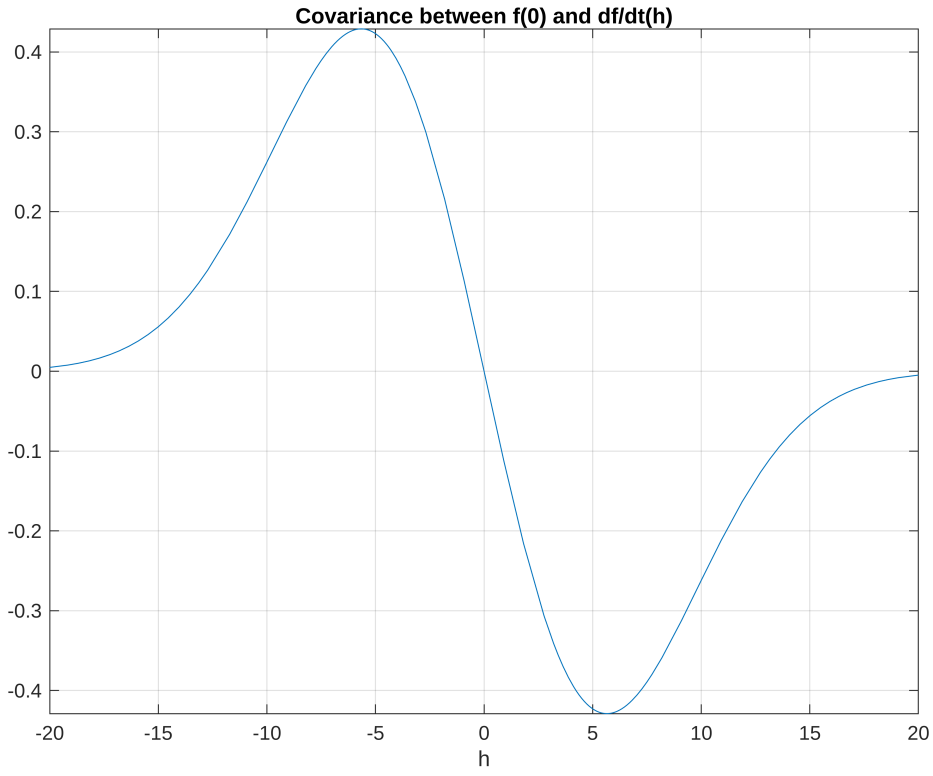
kposvel =

$$-\frac{h\,e^{-\frac{h^2}{64}}}{8}$$

```
fourier(kposvel)
```

ans = $32w\sqrt{\pi}\,e^{-16w^2}i$

```
fplot(kposvel,[-20 20]), grid on, title("Covariance between f(0) and df/
dt(h)"), xlabel("h")
```



```
kvel=-simplify(diff(kappa(h),2))
```

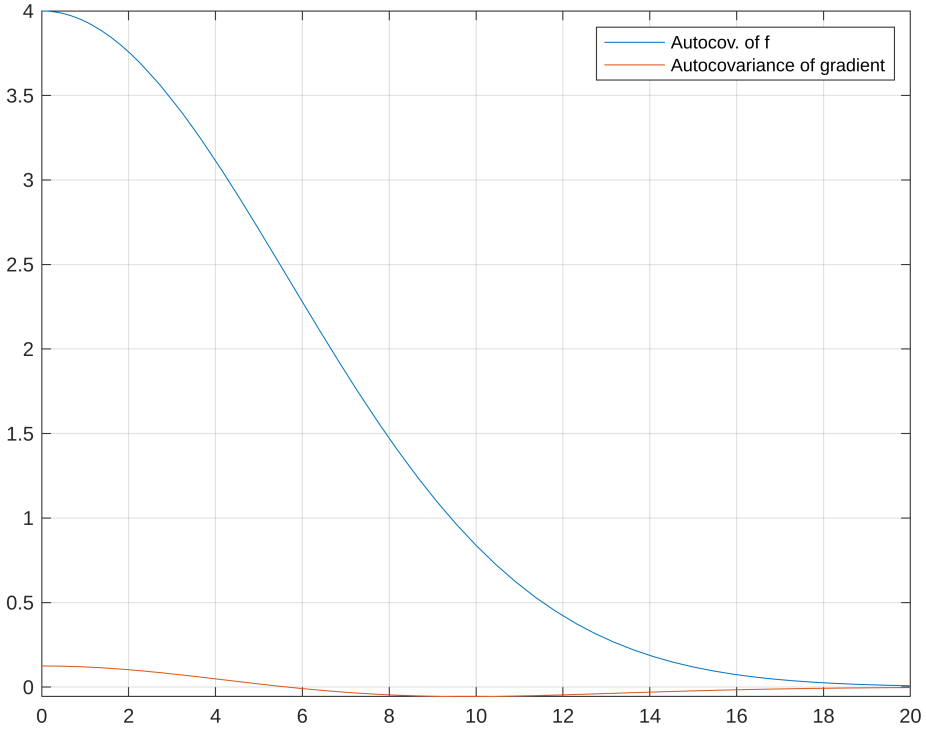
kvel =

$$-\frac{e^{-\frac{h^2}{64}}\,(h^2-32)}{256}$$

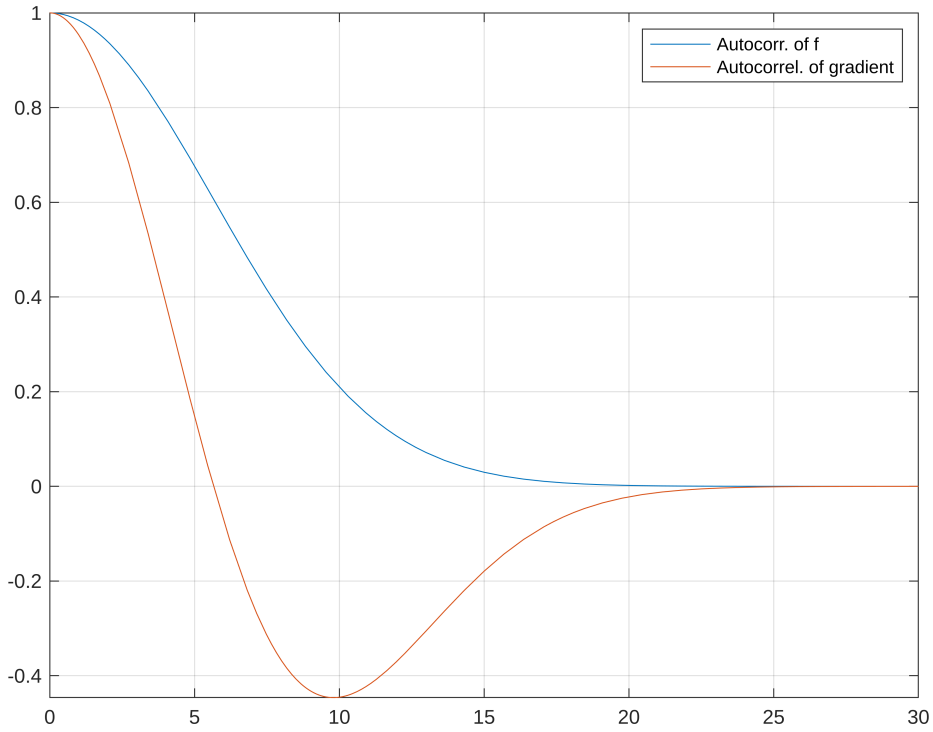
```
fourier(kvel)
```

ans = $32\,w^2\,\sqrt{\pi}\,e^{-16\,w^2}$

```
fplot([kappa; kvel],[0 20]), grid on, legend("Autocov. of f",
"Autocovariance of gradient")
```




```
fplot([@(h) kappa(h)/4; kvel*8],[0 30]), grid on, legend("Autocorr. of f",
"Autocorrel. of gradient")
```



Estimation with position measurements

Let us have two "position" measurements:

```
X=[0;2]; %abscissae where pos. measurements are taken
Y=[0;1]; %the actual measurements
finite_difference_velocity_estimate=diff(Y)/diff(X)
```

```
finite_difference_velocity_estimate = 0.5000
```

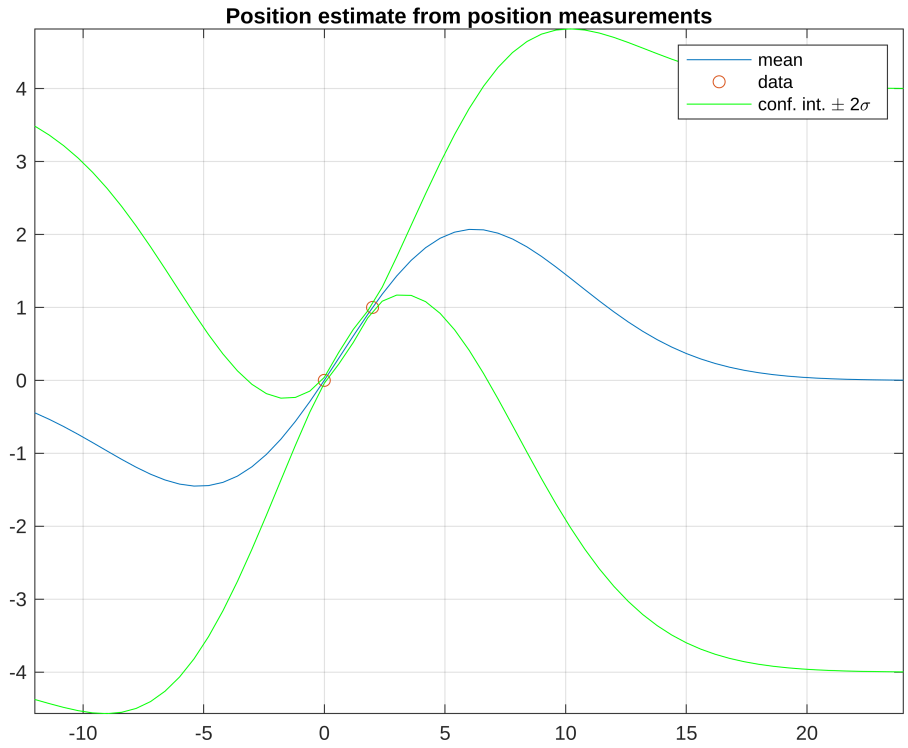
```
KK=KernelMatrix(X,X,K)
```

```
KK = 2×2
    4.0000    3.7577
    3.7577    4.0000
```

The estimation of the GP with those measurements is:

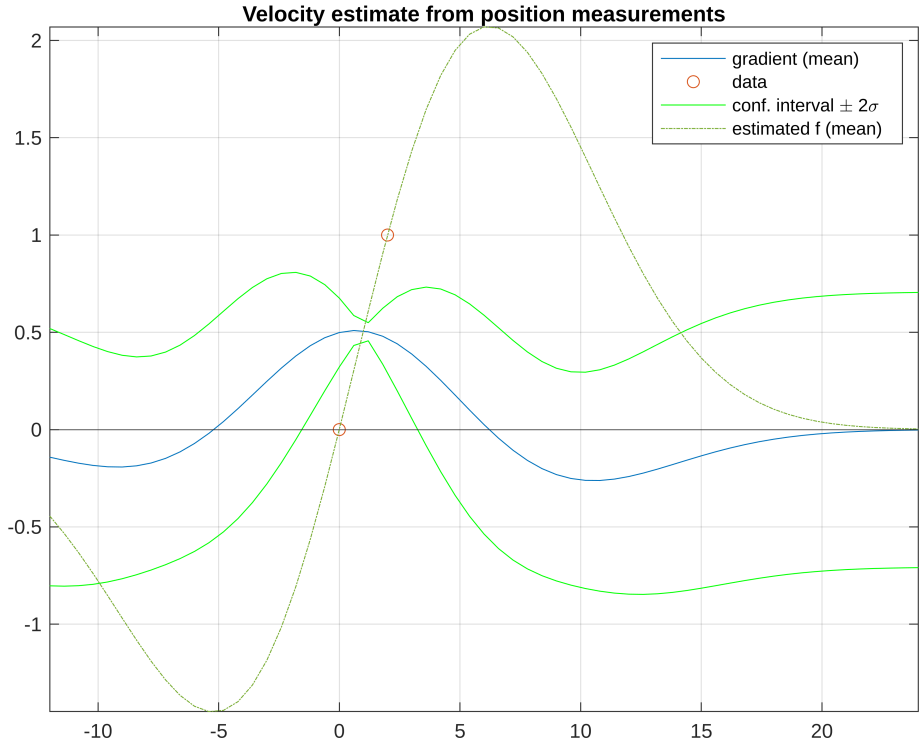
```
noisv=0.02^2; %measurement noise variance
XTest=6*(-2:0.1:4)'; %test points for plot;
KKtest = KernelMatrix(XTest,X,K);
INVK = inv(KK+eye(2)*noisv);
Yest = KKtest*INVK*Y; %estimated mean (prior mean = 0 is assumed)

%variance estimation (standard dev in plot)
N=size(XTest,1);
```

```
stdXTest=zeros(N,1);
for k=1:N
    vark1(k) = K(XTest(k),XTest(k)) - KKtest(k,:)*INVK*KKtest(k,:)';
    stdXTest(k) = sqrt(vark1(k));
end
plot(XTest,Yest,X,Y,'o',XTest,Yest+2*stdXTest,'g-',XTest,Yest-2*stdXTest,'g-'
), grid on, axis tight
legend("mean","data","conf. int. \pm 2\sigma")
title("Position estimate from position measurements")
```



Estimation of the function derivative (velocity) with position data

```
KgradTest=KernelMatrix(XTest,X,cov_df_f);
GradYest=KgradTest*INVK*Y; %estimated mean of the derivative

%estimated variance of the derivative
stdGXTest=zeros(N,1);
for k=1:N
    vark=cov_df_df(XTest(k),XTest(k))-KgradTest(k,:)*INVK*KgradTest(k,:)';
    stdGXTest(k)=sqrt(vark);
end
plot(XTest,GradYest,X,Y,'o',XTest,GradYest+2*stdGXTest,'g',XTest,GradYest-2*s
tdGXTest,'g',XTest,Yest,'-.'), grid on
yline(0), legend("gradient (mean)","data","conf. interval \pm
2\sigma","","estimated f (mean)"), axis tight
title("Velocity estimate from position measurements")
```



Adding slope (velocity) measurements

```
Xder=11; %point where slope is measured
Yder=-0.0; %slope measurement
```

We build the information covariance matrix with 3 measurements (2 positions, 1 slope):

```
K12_3=KernelMatrix(X,Xder,cov_f_df)
```

```
K12_3 = 2×1
   -0.2076
   -0.3173
```

```
K3_3=KernelMatrix(Xder,Xder,cov_df_df)
```

```
K3_3 = 0.1250
```

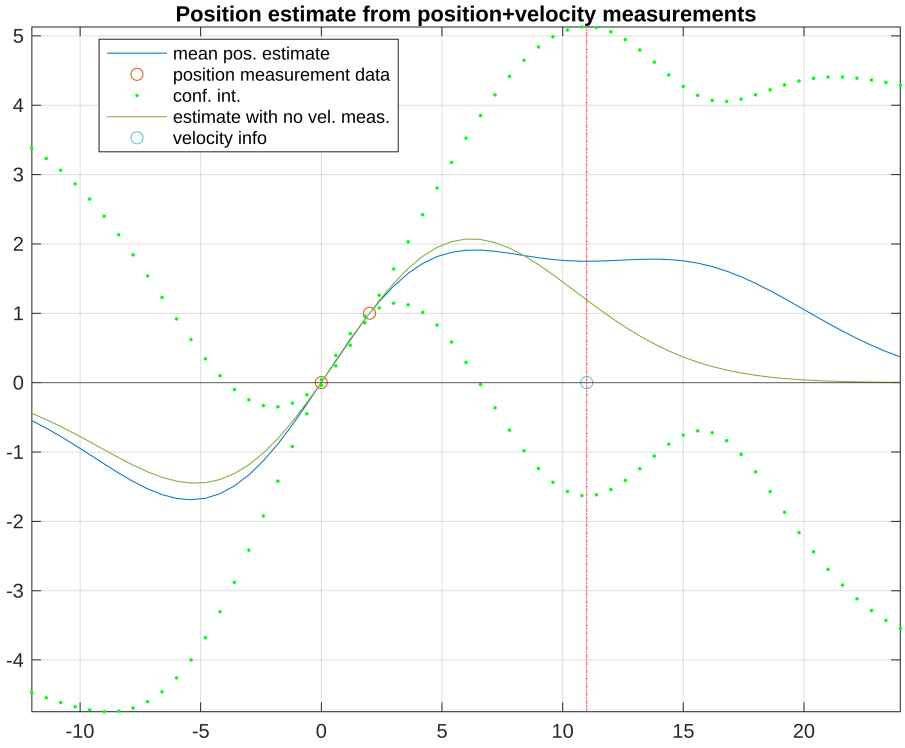
```
KKbig=[KK K12_3;K12_3' K3_3] %The new Kernel matrix of the available samples
```

```
KKbig = 3×3
    4.0000    3.7577   -0.2076
    3.7577    4.0000   -0.3173
   -0.2076   -0.3173    0.1250
```

Estimating the function at other points

We need to add the covariance of f at a test point with f' at the measured point:

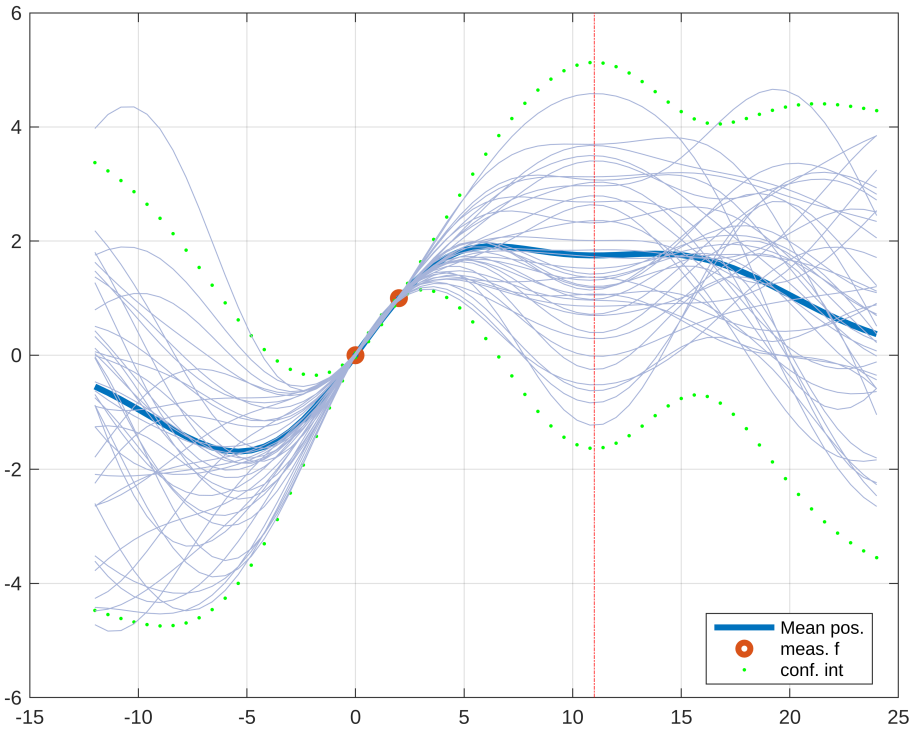
```
KKtest_3=KernelMatrix(XTest,Xder,cov_f_df);
KKtest_big=[KKtest KKtest_3];
noisedervz=0.01^2; %noise with which slope is measured
INVK2=inv(KKbig+blkdiag(eye(2)*noisv,noisedervz));

Yest2=KKtest_big*INVK2*[Y;Yder]; %estimated mean (zero prior mean position
and speed)

%variance estimation
N=size(XTest,1);
stdXTest=zeros(N,1);
%only diagonal of covariance matrix for just confidence interval plot
%for k=1:N
%    vark1(k)=K(XTest(k),XTest(k))-KKtest_big(k,:)*INVK2*KKtest_big(k,:)';
%    stdXTest(k)=sqrt(vark1(k));
%end
%WHOLE covariance matrix for later simulation
VarFULL=KernelMatrix(XTest,XTest,K)-KKtest_big*INVK2*KKtest_big';
stdXTest=sqrt(diag(VarFULL));

plot(XTest,Yest2,X,Y,'o',XTest,Yest2+2*stdXTest,'g.',XTest,Yest2-2*stdXTest,'
g.',XTest,Yest,Xder,0*Yder,'o')
yline(0), grid on, axis tight,xline(Xder,'-.r')
legend("mean pos. estimate","position measurement data","conf.
int.","","estimate with no vel. meas.","velocity info",Location="best")
title("Position estimate from position+velocity measurements")
```




```
VarFULL=0.5*(VarFULL+VarFULL');
```

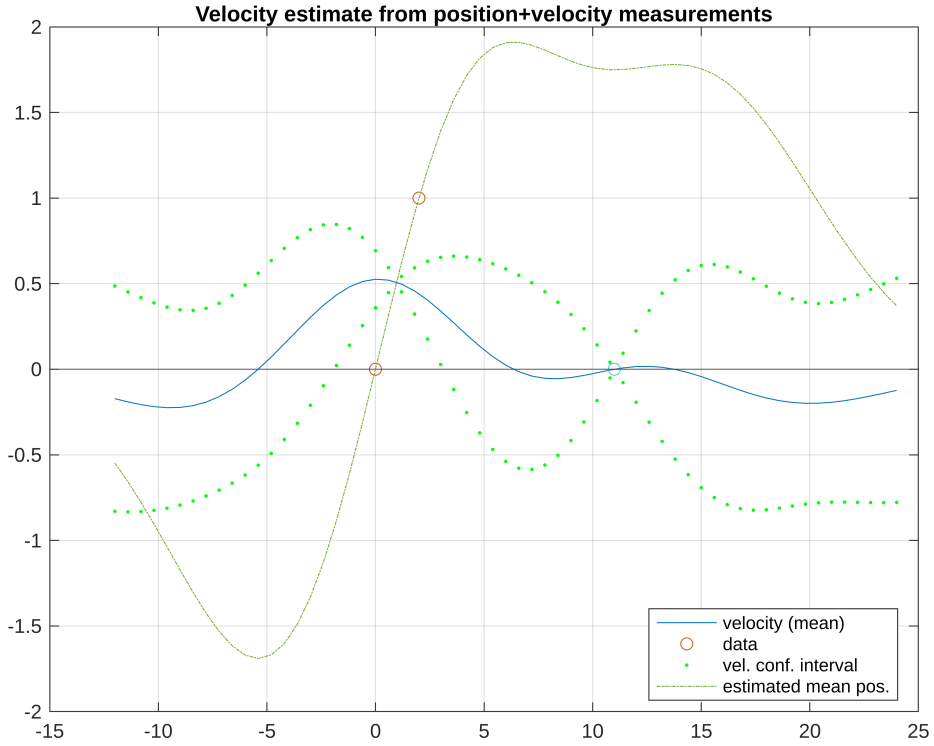
Let us simulate some realizations:

```
ttt=mvnrnd(Yest2',VarFULL,40);
plot(XTest,Yest2,X,Y,'o',XTest,Yest2+2*stdXTest,'g.',XTest,Yest2-2*stdXTest,'
g.','LineWidth',3)
hold on,
plot(XTest,ttt,Color=[.65 .7 .85]), grid on, xline(Xder,'r-.'), hold off
legend("Mean pos.","meas. f","conf. int",Location="best")
```



Let us estimate the gradient (speed) of the function

```
KgradTest2=KernelMatrix(XTest,Xder,cov_df_df);
KgradTestbig=[KgradTest KgradTest2];
GradYest=KgradTestbig*INVK2*[Y;Yder]; %mean
%varianza estimada
stdGXTest=zeros(N,1);
for k=1:N
    vark=cov_df_df(XTest(k),XTest(k))-
KgradTestbig(k,:)*INVK2*KgradTestbig(k,:)';
    stdGXTest(k)=sqrt(vark);
end
plot(XTest,GradYest,X,Y,'o',XTest,GradYest+2*stdGXTest,'g.',XTest,GradYest-2*
stdGXTest,'g.',XTest,Yest2,'-.',Xder,Yder,'o'), grid on
yline(0), legend("velocity (mean)","data","vel. conf. interval
","","estimated mean pos.",Location="best")
title("Velocity estimate from position+velocity measurements")
```




```
function KK=KernelMatrix(x1,X2,K)
for i=1:size(x1,1)
    for j=1:size(X2,1)
        KK(i,j)=K(x1(i),X2(j));
    end
end
end
```