Matérn + splines + mean estimates: Matlab example

Presentations in video: https://personales.upv.es/asala/YT/V/maternr1EN.html

https://personales.upv.es/asala/YT/V/maternr2EN.html

Objectives: understand the Matérn kernel and its relation to splines (piecewise polynomial interpolation) via some GP regression examples.

Matérn kernel: definition

```
syms x real
n=2; %"spectral factor " A/(tau*s+1)^n
rho=5; %distance parameter, proportional to "tau"
sgy=3; %non-parametric standard deviation
sgmed=2; %parametric component standard dev. (only a constant mean
level here)
nu=n-1/2;
maternsym(x)=simplify( ...
    sgy^2*2^(1-nu)/gamma(nu)*(sqrt(2*nu)*x/
rho)^nu*besselk(nu,sqrt(2*nu)*x/rho));
vpa(maternsym,4)
```

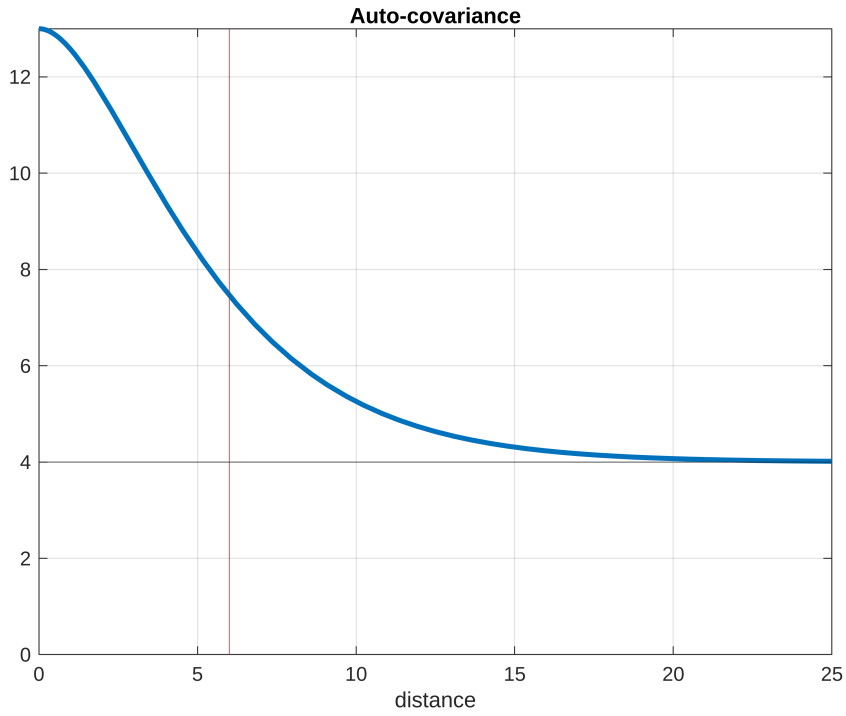
ans(x) = $1.039\,e^{-0.3464\,x}\,(3.0\,x+8.66)$

```
PSD=vpa(simplify(fourier(maternsym(abs(x)))),4)
```

PSD =

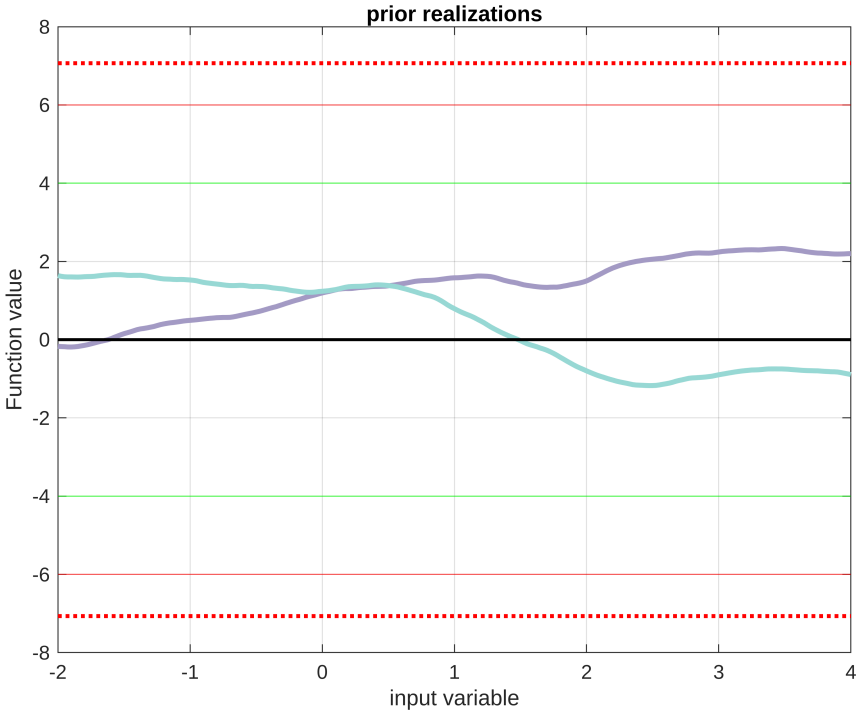
$$\frac{935.3}{(25.0\,w^2+3.0)^2}$$

```
maternnum=matlabFunction(maternsym);%compile to numerical from
symbolic
Data.kernel=@(x1,x2) ...
    maternnum(max(norm(x2-x1),1e-12)) + sgmed^2; %Matérn kernel,
non-centered
%Data.kernel=@(x1,x2)  sgy^2*exp(-norm(x2-x1)^2/2/rho^2)+sgmed^2;
%Gaussian kernel, non-centered

Xtest=-2:(1/40):4; %uniform grid to test stuff
maxd=max(Xtest)-min(Xtest);
LL=length(Xtest);
fplot(maternsym+sgmed^2,[0 ,max(5*rho,maxd)],LineWidth=2.5), grid
on, xline(maxd,'r')
yline(sgmed^2)
title("Auto-covariance"), xlabel("distance")
ylim([0 sgmed^2+sgy^2])
```



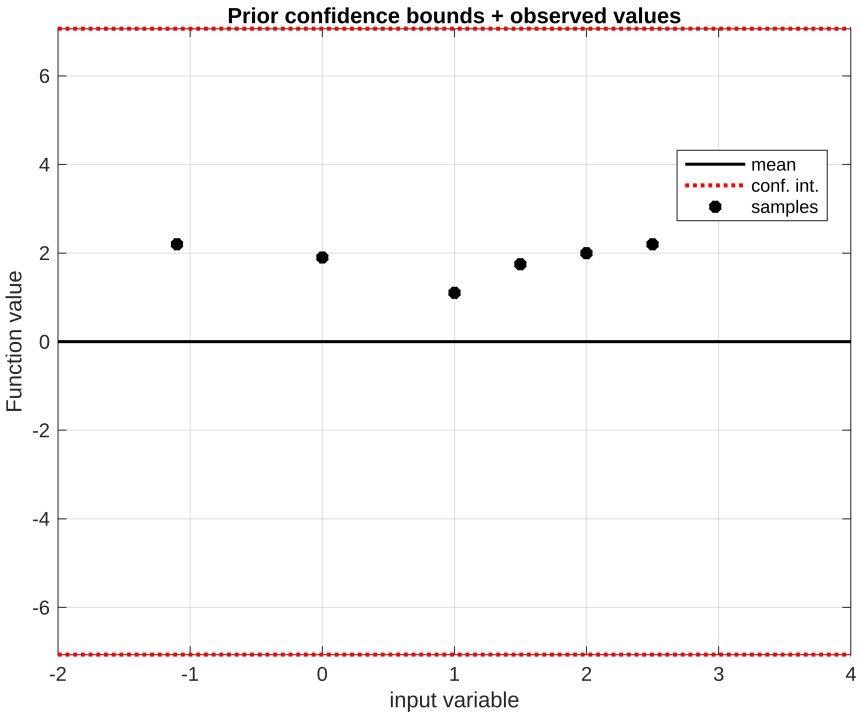
Prior

```
Data.X=[]; Data.Y=[]; %To test "prior" model, empty data
Data.K=@K;
STDMeasureNoise=4e-4;
VarMeasureNoise=STDMeasureNoise^2;

Data.priormean=@(x) zeros(size(x));
[MeanPrior,CovMatrixPrior]=predictGP(Xtest,Data,VarMeasureNoise);
sg=sqrt(diag(CovMatrixPrior));
PredPrior=[MeanPrior-1.96*sg MeanPrior MeanPrior+1.96*sg];
NTrials=2;
FunctionRealization=mvnrnd(MeanPrior,CovMatrixPrior,NTrials);%clean
for i=1:NTrials
    col=[.5 .6 .65]+rand()*[.3 0 0]+rand()*[0 .3 0]+rand()*[0 0
0.2];
    plot(Xtest,FunctionRealization(i,:),Color=col,LineWidth=2.5),
hold on
end
plot(Xtest,PredPrior(:,2),'k',LineWidth=1.5), grid on
plot(Xtest,PredPrior(:,[1 3]),'r:',LineWidth=2)
yline(2*sgy,'r'),yline(-2*sgy,'r')
yline(2*sgmed,'g'),yline(-2*sgmed,'g')
title("prior realizations")
xlabel("input variable"), ylabel("Function value")
```



Loading the Gaussian process with past observations

```
X=[-1.1 0 1 1.5 2 2.5];
Y=[2.2 1.9 1.1 1.75 2 2.2];
Ndat=6;
Data.X=X(1:Ndat);
Data.Y=Y(1:Ndat);

figure()
plot(Xtest,PredPrior(:,2),'k',LineWidth=1.5), grid on, hold on
plot(Xtest,PredPrior(:,[1 3]),'r:',LineWidth=2)
plot(Data.X,Data.Y,'*k',LineWidth=3),
hold off
xlabel("input variable"), ylabel("Function value")
title("Prior confidence bounds + observed values")
legend("mean","conf. int.","","samples",Location="best")
axis tight
```

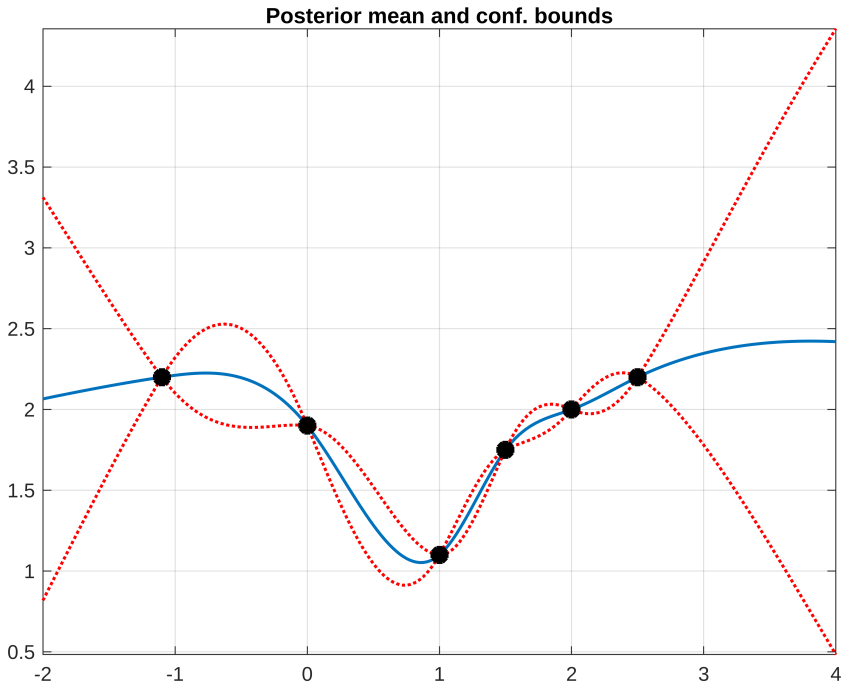


Posterior

```
[post_mean,post_var]=predictGP(Xtest,Data,VarMeasureNoise);
sg=sqrt(diag(post_var));
Pred=[post_mean-1.96*sg post_mean post_mean+1.96*sg];
```

Plots

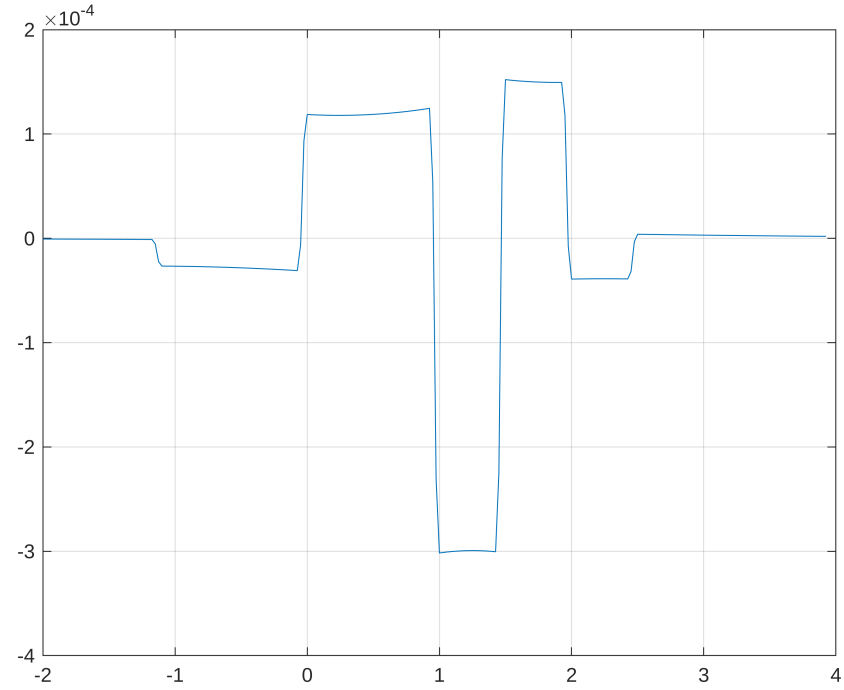
```
figure()
plot(Xtest,Pred(:,2),LineWidth=1.5)
hold on
plot(Xtest,Pred(:,[1 3],:),'r:',LineWidth=1.5)
plot(Data.X,Data.Y,'*k',LineWidth=3,MarkerSize=9)
grid on
hold off, axis tight, title("Posterior mean and conf. bounds")
```




```
degree_spline=floor(2*n-1)
```

```
degree_spline = 3
```

```
plot(Xtest(1:(end-degree_spline)),diff(post_mean,degree_spline)),
grid on
```



Compare GP and splines:

● Splines are not "statistics", so easier to explain in a curve-fitting context, conceptually simpler. Derivative constraints may be added.

● GP extend to multiple dimensions.

The formula is always the same: Gaussian process ridge regression.

Also, we control the "spatial frequency" and "extrapolation" so if distances are larger than "max. correlation distance", we don't interpolate/extrapolate.

GP allows for approximate fit (measurement noise) and analysis of the likelihood that the data sample is generated by the prior.

Derivative constraints may be added, to mimic splines... but, well, that needs changes to the GP regression code, omitted for brevity and simplicity.

```
function [mu,Var]=predictGP(x1,Data,VarMeasureNoise)
%set up the PRIOR mean and covariance at prediction points x1
priormean=Data.priormean;
kernel=Data.kernel;
K=@Data.K;
mu=priormean(x1)';
Var=K(x1,x1,kernel);
N=length(Data.X);
if(N>0) %compute POSTERIOR with some data
    K1X=K(x1,Data.X,kernel);
    Gain=K1X/(K(Data.X,Data.X,kernel)+VarMeasureNoise*eye(N));
    mu=mu+Gain*(Data.Y-priormean(Data.X))';
    Var=Var-Gain*K1X';
    Var=0.5*(Var+Var'); %roundoff-errors, correct them
end
end


function km=K(x1,x2,kernel)
N1=length(x1); N2=length(x2); %1D case
km=zeros(N1,N2);
for i=1:N1
    for j=1:N2
        km(i,j)=kernel(x1(:,i),x2(:,j)); %do it for all pairs
(x1_i,x2_j)
    end
end
end
```