Gaussian Process: Karhunen-Loeve (principal) components [Matlab example]

*Executed in Matlab **R2023b**

Presentations in video:

http://personales.upv.es/asala/YT/V/gphk2EN.html

http://personales.upv.es/asala/YT/V/gphk2pEN.html

Objectives: A Gaussian Process is a "random function". Let us look at the "harmonics" (Karhunen-Loeve eigenfunctions, principal components) in which the GP can be decomposed.

Table of Contents

Analysis without data samples (prior mean and covariance kernel)

```
Data.X=[]; Data.Y=[]; %To test "prior" model
Data.K=@K;
Xtest=-2:(1/40):4; %uniform grid to test stuff, dense enough...
LL=length(Xtest)
```

```
LL = 241
```

```
PredPrior=zeros(3,LL); %we'll predict confidence interval and mean
[Mean,CovMatrix]=predictGP(Xtest,Data);
size(Mean)
```

```
ans = 1×2
   241     1
```

```
size(CovMatrix) %to plot conf. interval we might prefer carrying out 121
"single" predictions to get the diagonal.
```

```
ans = 1×2
   241   241
```

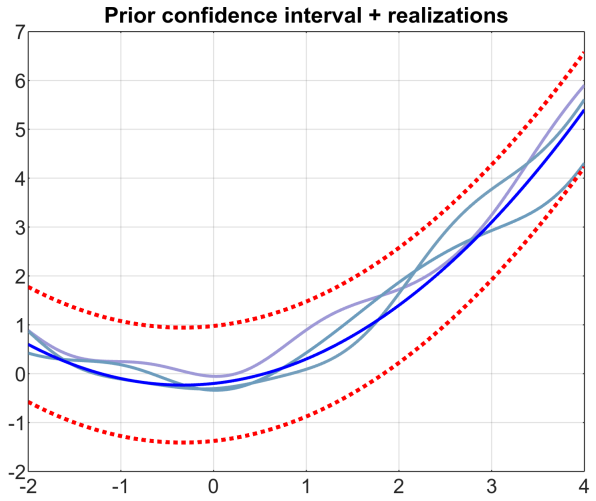
The prior is stationary (given how we defined it at the end of the file).

Confidence bounds and example samples

```
sg=sqrt(diag(CovMatrix)); %std. dev.
```

```
PredPrior=[Mean+1.96*sg Mean Mean-1.96*sg];
NTrials=3;
F1=mvnrnd(Mean,CovMatrix+1e-10*eye(LL),NTrials);
%clean, no "measurement noise" or +eye(xxx) with noise... but the true white
%noise is "infinite frequency", we have a cutoff frequency here due to finite
%separation between points in Xtest

for i=1:NTrials
    if(NTrials<4)
        LW=1.5;
    else
        LW=1;
    end
    col=[.4 .55 .7]+rand()*[.3 0 0]+rand()*[0 .25 0]+rand()*[0 0 0.2];
    plot(Xtest,F1(i,:),Color=col,LineWidth=LW), hold on
end
plot(Xtest,PredPrior(:,2),'b',LineWidth=1.5), grid on
plot(Xtest,PredPrior(:,[1 3]),':r',LineWidth=2), hold off
title("Prior confidence interval + realizations")
```



Square root of covariance matrix (discretized at test points), interpretation

If we have samples of the GP, call them y_k, $1 \le k \le 241$ in our case, and the symmetric covariance matrix $K_{241\times241}$ can be factorised as $K = Q_{241\times241} \cdot Q^T_{241\times241}$, we'll say that Q is a "matrix" square root of K.

Example: if $K = VDV^T$, we can set $Q = V \cdot \sqrt{D}$ (principal component analysis, PCA), as $Q \cdot Q^T = V\sqrt{(D)} \cdot \sqrt{(D)}V^T = K$.

*Other authors may say that B is a square root of a if $B \cdot B = A$, so be aware on which definition is used if you look up elsewhere. In here, we will use the $A = B \cdot B^T$ interpretation, only valid for symmetric matrices, of course.

So, the GP (well, increments from its mean function) can be generated by "latent" standard normal random variables ξ, as $y = Q\xi_{241\times 1}$, $\xi \sim N(0, I_{241})$ because, indeed, the covariance matrix is $E[yy^T] = E[Q\xi\xi^T Q^T] = Q \cdot E[\xi\xi^T] \cdot Q^T = Q \cdot I \cdot Q^T = K$.

*The internals of Matlab's `mvnrnd` work based on this fact.

Matrix square root is not unique, as any orthogonal matrix U, i.e., fulfilling $UU^T = I$ generates another square root QU because $K = (QU) \cdot (U^T Q^T)$.

Different matrix square roots have different interpretations of the "latent" random variables.

There will be several of them with interest in discrete-time stochastic processes: the "orthogonal" (diagonalisation, Karhunen-Loeve), the "triangular" (Cholesly, causal or anticausal representations) and the "symmetric" one, to be discussed in several videos in the collection.

Of course, there is theory generalising to the "discrete-time" going from $-\infty$ to $+\infty$ and to the full "continuous-time" case (spectral factor, all-pass functions, Fourier features) but it is more involved, out of the scope here...

Diagonalization of covariance matrix (discretized at test points), orthogonal square root

```
[V,D]=eig(CovMatrix); %eigenvalues + eigenvectors
[eevv,idx]=sort(diag(D),1,"descend");
eevv'
```

```
ans = 1×241
   29.9945   23.6347   15.9455    9.2612    4.6654    2.0571    0.8018    0.2790 ···
```

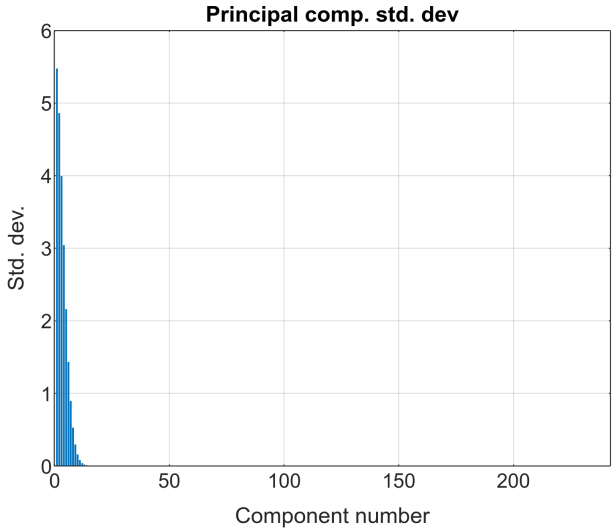
```
V=V(:,idx); D=D(:,idx); %we sort eigenvalues and eigenvectors to match
minev=min(eevv) %eevv are positive... except for numerical tolerance
issues...
```

```
minev = -2.3205e-15
```

```
eevv=eevv+max(0,-minev); %now minimum is non-negative
```

The leading components account for most of the variability:

```
bar(sqrt(eevv)), title("Principal comp. std. dev"), grid on,
xlabel("Component number"), ylabel("Std. dev.")
```




```
varexpl=cumsum(eevv)/sum(eevv);
varexpl'
```

```
ans = 1×241
    0.3457    0.6181    0.8019    0.9087    0.9624    0.9861    0.9954    0.9986 ···
```

```
stdexpl=sqrt(varexpl);
varNOTexpl=(sum(eevv)-cumsum(eevv))/sum(eevv);
varNOTexpl'
```

```
ans = 1×241
    0.6543    0.3819    0.1981    0.0913    0.0376    0.0139    0.0046    0.0014 ···
```

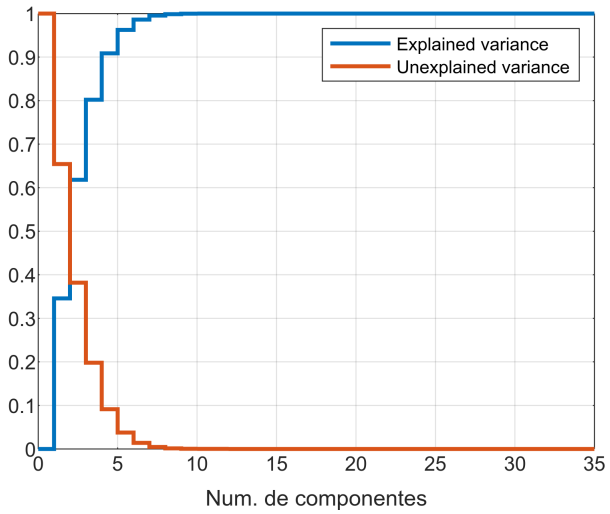
```
stdNOTexpl=sqrt(varNOTexpl);
stdexpl'
```

```
ans = 1×241
    0.5880    0.7862    0.8955    0.9532    0.9810    0.9931    0.9977    0.9993 ···
```

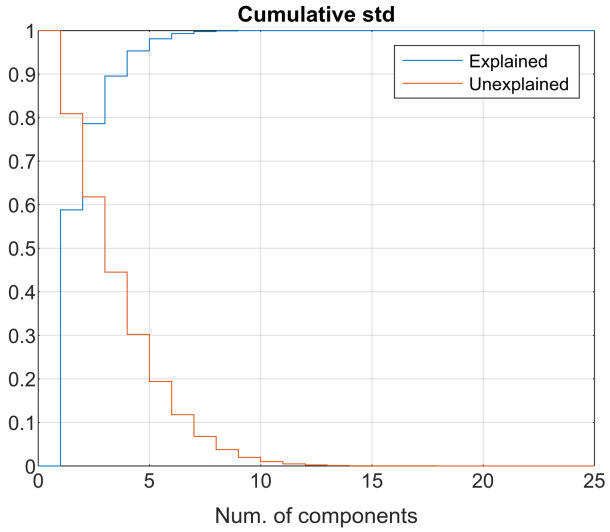
```
stdNOTexpl'
```

```
ans = 1×241
    0.8089    0.6180    0.4451    0.3022    0.1938    0.1177    0.0679    0.0373 ···
```

```
stairs(0:LL,[[0; varexpl] [1;varNOTexpl]], LineWidth=2),xlim([0 35])
grid on, legend("Explained variance","Unexplained variance"), xlabel("Num.
de componentes")
```




```
stairs(0:LL,[[0; stdexpl] [1;stdNOTexpl]]), grid on,ylim([0 1]), xlim([0 25])
title("Cumulative std"),xlabel("Num. of components"),
legend("Explained","Unexplained")
```



Note: we carried out principal component analysis (meaning of eigenvalues and eigenvectors of covariance matrix)... If we have "many" points, we do approach the continuous GP eigenfunctions and eigenvalues, the Karhunen-Loeve ones:

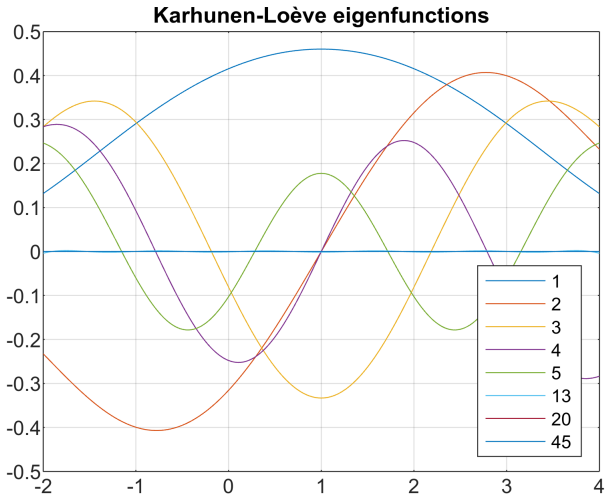
$Kv_i = \lambda_i v_i$ (matrix eigenvectors) get converted in the limit case to $\Rightarrow \quad \int_{-2}^{4} \kappa(x,s) \cdot \phi_i(s)\, ds = \lambda_i \cdot \phi_i(x)$

covariance kernel eigenfunctions

Let us have a look at the eigenvectors (i.e., eigenfunctions):

```
PrincCom=V*diag(sqrt(abs(eevv)));
```

```
plot(Xtest,PrincCom(:,[1 2 3 4 5 13 20 45])),
legend("1","2","3","4","5","13","20","45", Location="best")
%plot(Xtest,PrincCom(:,[114])), legend("1","2","3","4","5","13","20","45",
Location="best")
title("Karhunen-Loève eigenfunctions"), grid on
```

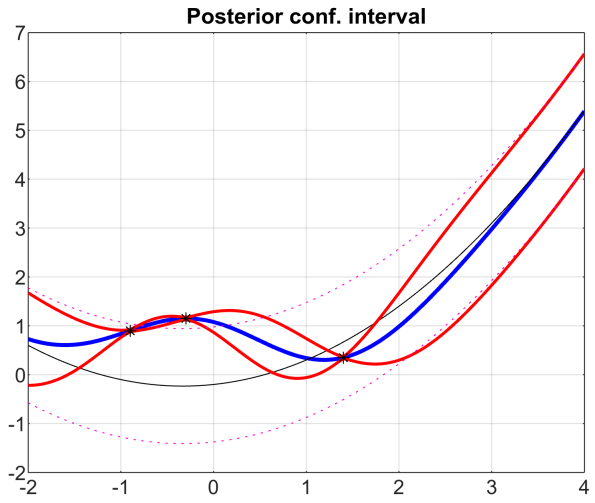



```
Q=PrincCom; %matrix square root for animation
```

*Execute animPCA.m script now.

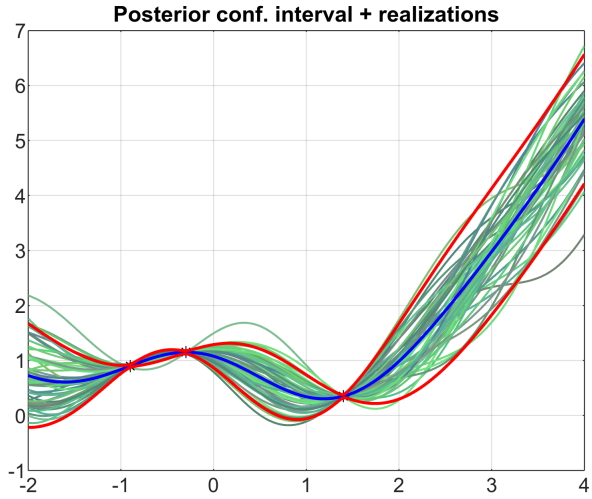
Load with "experimental" samples and redo computations

```
Data.X=[-0.9 -.3 1.4];
Data.Y=[0.9 1.15 0.35];
Pred=zeros(3,LL);
[mm,sss]=predictGP(Xtest,Data);
sg=sqrt(diag(sss));
Pred=[mm+1.96*sg mm mm-1.96*sg];
plot(Xtest,Pred(:,2),'b',LineWidth=2)
hold on
plot(Xtest,PredPrior(:,2),'k')
plot(Xtest,Pred(:,[1 3]),'r',LineWidth=1.5)
plot(Xtest,PredPrior(:,[1 3]),':',Color=[1 .0 .8])
plot(Data.X,Data.Y,'*k');
hold off, grid on
title("Posterior conf. interval")
```



Sample the posterior

```
figure()
NTrials=50;
F1=mvnrnd(mm,sss,NTrials);%clean, no measurement noise in generated samples
for i=1:NTrials
    if(NTrials<4)
        LW=1.5;
    else
        LW=1;
    end
    col=[.3 .5 .4]+rand()*[.2 0 0]+rand()*[0 .4 0]+rand()*[0 0 0.2];
    plot(Xtest,F1(i,:),Color=col,LineWidth=LW), hold on
end
hold on
plot(Data.X,Data.Y,'*k');
plot(Xtest,Pred(:,2),'b',LineWidth=1.5)
plot(Xtest,Pred(:,[1 3]),'r',LineWidth=1.5)
hold off, grid on
title("Posterior conf. interval + realizations")
```

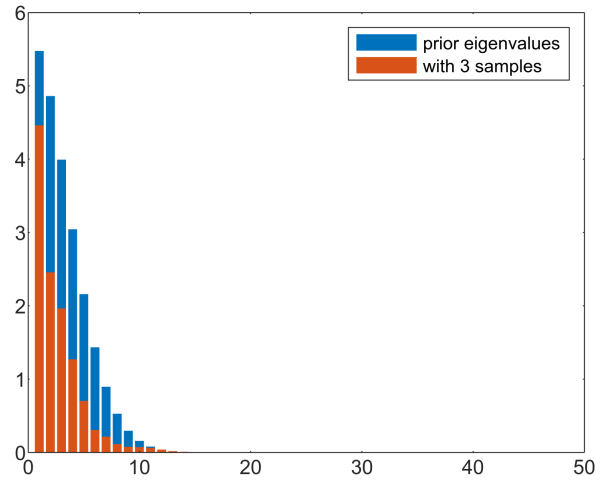


Covariance matrix analysis

```
[V,D]=eig(sss);
[eevv2,idx]=sort(diag(D),1,"descend");
eevv2'
```

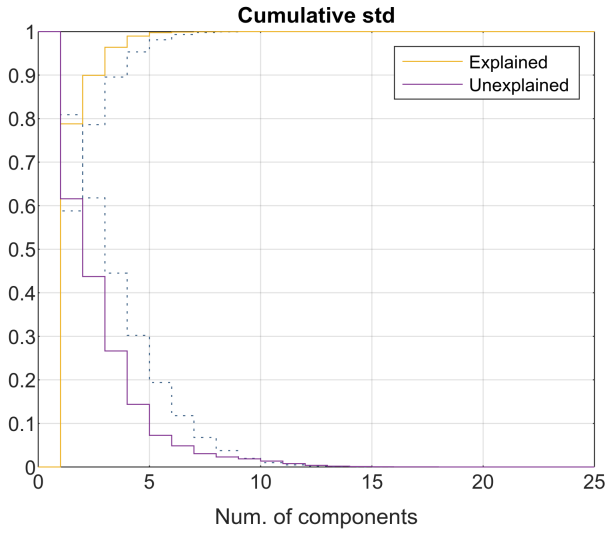
```
ans = 1×241
   19.9134    6.0322    3.8563    1.6128    0.4938    0.0939    0.0455    0.0131 ···
```

```
eevv2=eevv2+max(0,-min(eevv2));
bar(sqrt(eevv+1.5e-15)), hold on,bar(sqrt(eevv2+1.5e-15)), hold off
legend("prior eigenvalues","with 3 samples"),xlim([0 50])
```



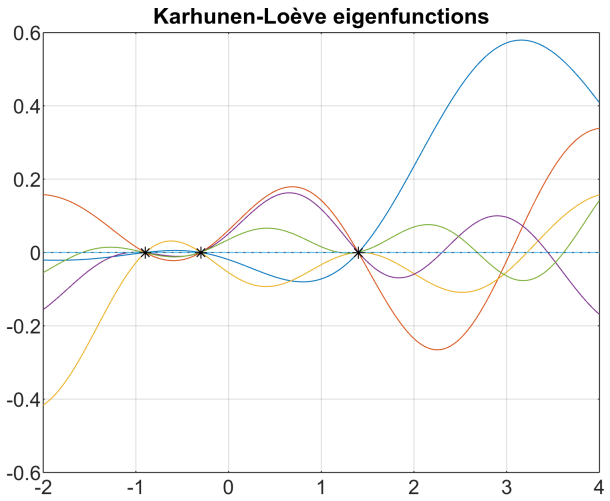

```
dvtexpl2=sqrt(cumsum(eevv2))/sqrt(sum(eevv2));
dvtNOTexpl2=sqrt(sum(eevv2)-cumsum(eevv2))/sqrt(sum(eevv2));
stairs(0:LL,[[0; stdexpl] [1;stdNOTexpl]],':',Color=[.2 .35 .5]), hold on
stairs(0:LL,[[0; dvtexpl2] [1;dvtNOTexpl2]]), hold off
grid on,ylim([0 1]), xlim([0 25])
```

```
title("Cumulative std"),xlabel("Num. of components"),
legend("","","Explained","Unexplained")
```




```
V=V(:,idx);
PrincCom=V*diag(sqrt(abs(eevv2)));
```

```
plot(Xtest,PrincCom(:,[1 2 3 4 5 19]))
hold on
plot(Data.X,zeros(length(Data.X),1),'*k'), grid on
yline(0,':')
hold off %components pass almost at zero at samples (except meas. noise)
title("Karhunen-Loève eigenfunctions")
```




```
Q=PrincCom;
```

*Execute animPCA.m script.

```
function [mu,Var]=predictGP(x1,Data)
arguments
    x1
    Data
end
priormean=@(x) 0.3*x.^2+0.2*x-.2;
lambda=1.5e-2^2; %measurement noise variance
K=@Data.K;
N=length(Data.X);
mu=priormean(x1)';
Var=K(x1,x1); %prior covariance among points in x1
if(N>0) %don't do this if no data available
    K1X=K(x1,Data.X);
    Gain=K1X/(K(Data.X,Data.X)+lambda*eye(N));
    mu=mu+Gain*(Data.Y-priormean(Data.X))';
    Var=Var-Gain*K1X';
    Var=0.5*(Var+Var');
end
end


function km=K(x1,x2)
M=0.6; sg=0.9;%hyperparameters stddev x and y
kernel=@(x1,x2) M^2*exp(-norm(x2-x1)^2/2/sg^2); %stationary, depends only on
DIFFERENCE
%kernel=@(x1,x2) M^2*exp(-norm(x2-x1)/sg); %1sr-order exponential
%kernel=@(x1,x2) pi*sin(pi/4 + (2^(1/2)*abs(x2-x1))/sg)*exp(-(2^(1/2)*abs(x2-
x1))/sg); %butterworth ord2
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));
    end
end
end
```

```
%% AnimPCA.m, Matlab R2023b
Varacum=zeros(LL,1);
figure(100)
clf
figure(101)
clf
stddev=norm(Q(1,:));%if stationary
Amplitudes=randn(LL,1);%to build↙
realization by components
for k=1:LL
    i=k;
    figure(100)
    Varacum=Varacum+Q(:,i).^2;
    if(k>1)
        plot(Xtest,Q(:,(1:(k-1))),↙
LineWidth=1)
    end
    hold on
    plot(Xtest,Q(:,i),LineWidth=3.↙
5), grid on
    if(k>1)
        plot(Xtest,sqrt↙
(Varacum),':k',LineWIdth=2)
```

```
        plot(Xtest,-sqrt↙
(Varacum),':k',LineWIdth=2)
    end
    hold off
    title("Component functions↙
(features), scaled by standard↙
dev")
    ylim([-stddev stddev]*1.8)
    fontsize(20,"points") %This↙
works from version 2023a or↙
posterior
    figure(101)
    plot(Xtest,Q(:,i)/max(abs(Q(:,↙
i))),LineWidth=3), grid on
    text(-1.9,.25,["i=" num2str(i)↙
num2str(sqrt(eevv(i)))])
    ylim([-1 1])
    title("Component functions↙
(features), scaled to unit↙
amplitude")
    fontsize(20,"points") %This↙
works from version 2023a or↙
posterior
```

```
    figure(102)
    for j=1:k %superimpose↙
previous approximations with less↙
components
        RealizTst=Q(:,(1:j))↙
*Amplitudes((1:j));
        if j<k
            plot(Xtest,RealizTst,↙
LineWidth=1,Color=[.9 .95 .7]*((k-↙
j*0.3)/k)^2)
        else
            plot(Xtest,Q(:,i)↙
*Amplitudes(i),LineWidth=3,Color=↙
[.8 .2 .4])
            plot(Xtest,RealizTst,↙
LineWidth=5+0.04*sqrt(i),Color=[.1↙
.2 .4])
        end
        grid on
        hold on
    end
    ylim([-stddev stddev]*2.3)
    title("Building a realization↙
```

```
component by component")
    hold off
    ylabel("Increment with respect↙
to mean")
    fontsize(20,"points") %This↙
works from version 2023a or↙
posterior
    pause
end
```