Report GPR LAB1

WANG Andi

1. **Using a similar structure, implement the functions for the following kernels: squared exp.(seKern), Matern5/2 (mat5_2Kern), Brownian (brownKern) and sinc (sincKern).**

   For the function squared exp. , we have the codes:

```
seKern<-function(x,y,param){
  # input:
  #  x,y: input vectors
  #  param: parameters (sigma,theta)
  # output:
  #  kern: covariance matrix cov(x,y)
  sigma<-param[1]
  theta<-param[2]
  dist<-outer(x/theta,y/theta,'-')
  kern<-sigma^2*exp(-1/2*dist^2)
  return(kern)
}
```

   For the function Matern5/2, we have the codes:

```
mat5_2Kern<-function(x,y, param){
  # input:
  #  x,y: input vectors
  #  param: parameters (sigma,theta)
  # output:
  #  kern: covariance matrix cov(x,y)
  sigma<-param[1]
  theta<-param[2]
  dist<-outer(x/theta,y/theta,'-')
  kern<-sigma^2*(1+sqrt(5)*abs(dist)+5/3*dist^2)*exp(-sqrt(5)*abs(dist))
  return(kern)
}
```

   For the function Brownian, we have the codes:

```
brownKern<-function(x,y,param){
  # input:
  #  x,y: input vectors
  #  param: parameters sigma
  # output:
  #  kern: covariance matrix cov(x,y)
  sigma<-param[1]
  n1<-length(x)
  n2<-length(y)
  kern<-matrix(0,n1,n2)

  for(i in 1:n1){
    for(j in 1:n2){
      kern[i,j]<-sigma^2*min(x[i],y[j])
    }
  }
  return(kern)
}
```

   For the function sinc, we have the codes:

```
sincKern<-function(x,y,param){
  # input:
  #  x,y: input vectors
  #  param: parameters (sigma,theta)
  # output:
  #  kern: covariance matrix cov(x,y)
  sigma<-param[1]
  theta<-param[2]
  dist<-outer(x/theta,y/theta,'-')

  kern<-sigma^2*1/dist*sin(dist)
  kern[is.na(kern)]<-sigma^2
  return(kern)
}
```

2. **Create a grid of 100 points on x, y [0, 1] and compute the covariance matrix associated to one of the kernel you wrote previously. How can you simulate zero-mean Gaussian samples based on this matrix? The function mvrnorm() from package MASS can be useful here.**

We create a grid on x and y with:

```
x <- seq(0, 1, 0.01) # regular grid
y<-x
```

Then we test one of the function kernel and plot it, we test with the function sinc, and we can get the image:

```
param <- c(1,0.2) # covariance parameters
k1 <- sincKern(x, x, param) # computing the covariance matrix using an exp. kernel
image2D(k1, theta = 90, xlab = "x", ylab = "y") # plotting the covariance matrix
```

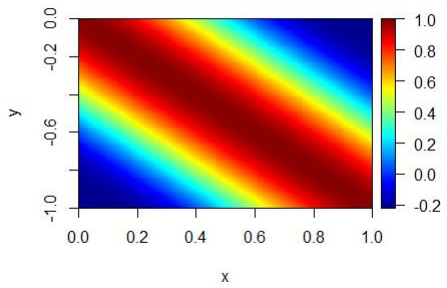
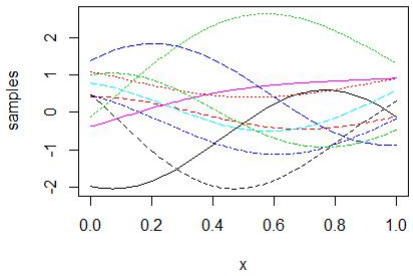


Figure 1 sigma=1, theta=0.2

Then we simulate zero-mean samples with the codes:

```
samples<-t(mvrnorm(100,mu=rep(0,nrow(k1)),Sigma=k1))
matplot(x,samples[,seq(10)],type = "l",ylab="samples")
```

And get the image:



3. **Change the kernel and the kernel parameters. What are the effects on the sample paths? Write down your observations.**

We also take the function sinc as an example for well explaining the observations:

We firstly enlarge the sigma:

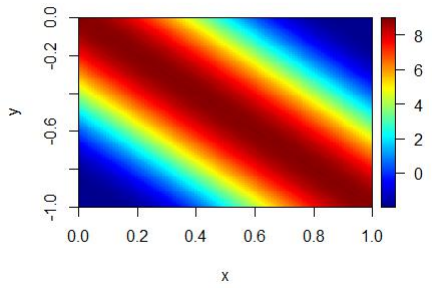


Figure 2 sigma=3, theta=0.2

We can find the image is almost the same as before, except the legend on the right, which can show the range of the kernel. With the enlargement of the sigma, the range of the kernel is also enlarged.

Then we change the theta:

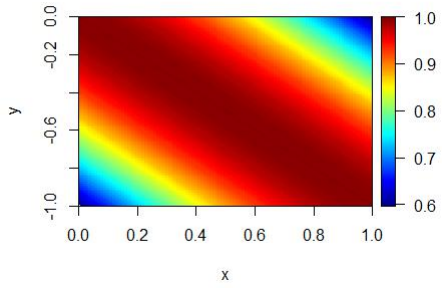


Figure 3 sigma=1, theta=0.6

We can know by this image, when we amplify the theta, the kernel will firstly change more slowly and then faster from the peak to the lowest value with x and y while comparing with the first image.

So we can say that the sigma decide the value of peak of the kernel, and theta decide the speed of change with x and y.

Some other test with different models:

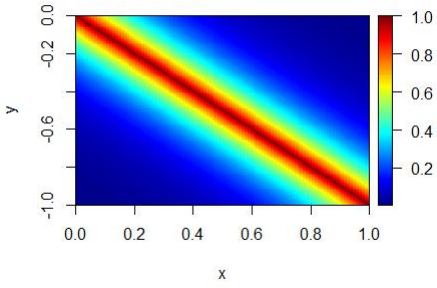


Figure 4 exponential example

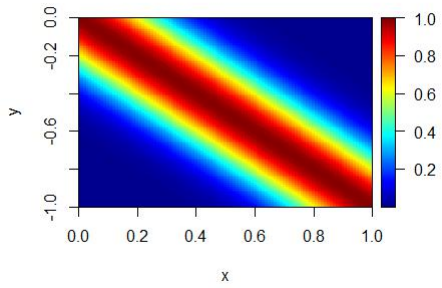


Figure 5 Squared exp. example

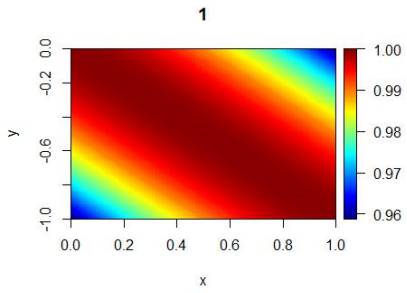
4. **Generate a large number of samples and extract the vectors of the samples valuated at two points of the input space. Plot the associated cloud of points. What happen if the two input points are close by? What happen if they are far away?**

   We then generate a large number of samples and extract the points from the samples with the codes:

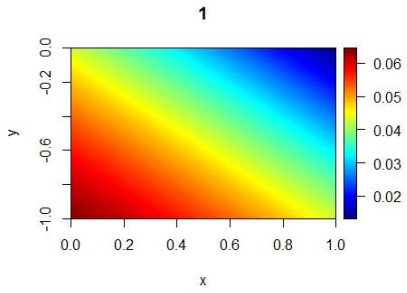
```
x1<-seq(0,10,0.001)
y1<-x1
index<-seq(1,101,1)
x<-x1[index]
alpha<-0
y<-y1[index]+alpha*matrix(0.001,length(x),1)
k3<-sincKern(x,y,param)
image2D(k3,theta = 90, xlab = "x", ylab = "y")
```

   We then change the value of alpha for testing 2 different input points.

   When the points closed by:



   When the points far away:



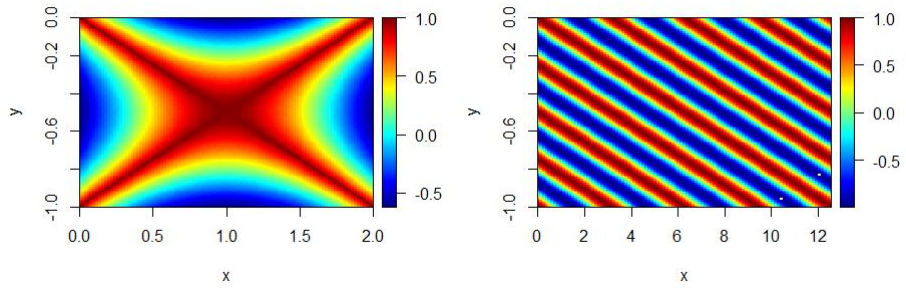
We can find that the peak of kernel moves along the axis y=x-1 and the value of kernel change more slowly.

5. **Design the corresponding kernels to model:**
   - **smooth and symmetric functions respect to the axis x = 1,**
   - **smooth and p-periodic functions in the interval [0, 4*pi]**

   We create these two models with the codes:

```
Kern5_1<-function(x,y,param){
  # input:
  #  x,y: input vectors
  #  param: parameters (sigma,theta)
  # output:
  #  kern: covariance matrix cov(x,y)
  sigma<-param[1]
  theta<-param[2]
  n1<-length(x)
  n2<-length(y)
  kern<-matrix(0,n2,n1)
  for(i in 1:n1){
    for(j in 1:n2){
      dist<-outer(4*(y[j]-0.5)^2/theta,(x[i]-1)^2/theta,'-')
      kern[j,i]<-sigma^2*cos(sqrt(abs(dist)))
    }
  }
  return(kern)
}
Kern5_2 <- function(x, y, param){
  # input:
  #  x,y: input vectors
  #  param: parameters (sigma,theta)
  # output:
  #  kern: covariance matrix cov(x,y)
  sigma <- param[1]
  theta <- param[2]
  dist <- outer(y*(8*pi),2*x, '-')
  kern <- sigma^2*cos(dist)
  return(kern)
}
```

   For test the subject, we plot the images:



   And then we do the same studies as before and get the similar conclusion.

6. **Bonus question. Let the process given by**

   xt = t + cos(2pt) + yt, (1)

   for discrete values of t, with yt = Φ*yt−1 + εt an autoregressive model with white noise εt and Φ on]0, 1[. Design a proper kernel for the process of Equation (1), and generate some samples.

   How can it be extended for continuous time series?

   We can study the series:

$$y_t = \Phi y_{t-1} + \varepsilon_t = \Phi(\Phi y_{t-2} + \varepsilon_{t-1}) + \varepsilon_t = \phi^t y_0 + \sum_{i=0}^{i=t-1} \phi^t \varepsilon_{t-i}$$

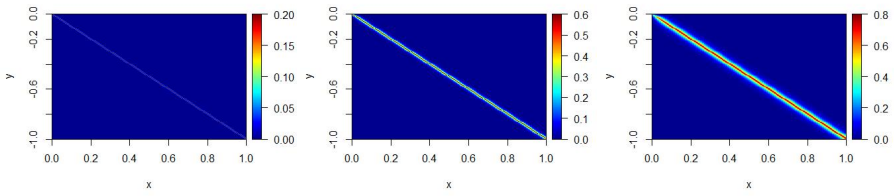
$$\mathrm{cov}(x_{t-h},y_t) = \mathrm{cov}(t-h+\cos(2\pi(t-h)) + y_{t-h},y_t) = \mathrm{cov}(y_{t-h},y_t)$$
$$= \mathrm{cov}\left(y_{t-h},\phi^h y_{t-h} + \sum_{i=0}^{i=h-1} \phi^t \varepsilon_{t-i}\right) = \phi^h \mathrm{cov}(y_{t-h},y_{t-h})$$
$$\text{because of } \varepsilon_i \text{ and } y_{t-h} \text{ independent if } i \geq t-h$$

by the same way, we have $\mathrm{cov}(x_t,y_{t-h}) = \mathrm{cov}(t+\cos(2\pi(t)) + y_t,y_{t-h}) = \mathrm{cov}(y_{t-h},y_t) = \mathrm{cov}\left(y_{t-h},\phi^h y_{t-h} + \sum_{i=0}^{i=h-1} \phi^t \varepsilon_{t-i}\right) = \phi^h \mathrm{cov}(y_{t-h},y_{t-h})$

$$\mathrm{var}(y_t) = \mathrm{var}\left(\sum_{i=0}^{i=t-1} \phi^t \varepsilon_{t-i}\right) = \frac{1-\phi^t}{1-\phi}$$

```
kernBon<-function(y,param,t){
  phi<-param[1]
  n<-length(y)
  kern<-matrix(0,n,n)
  mu<-t[1]-1
  for(i in 1:n){
    for(j in 1:n){
      kern[i,j]<-phi^abs(i-j)*(phi-phi^(mu+min(i,j)))
    }
  }
  return(kern)
}
```

We test with different parameters (bigger than bigger):



We replace xt and yt with the relation between x and xt and also relation of y and yt, then we can get the time series.

7. **Create a design of experiment X composed of 5 to 20 points in the input space (regularly spaced for instance) and compute the vector of observations Y = f (X).**

We choose X and calculate the Y with the codes:

```
x<-seq(0,1,0.01)
y<-x

index<-sort(sample(100,5))
xtest<-x[index]
ytest<-xtest+sin(4*pi*xtest)
f<-x+sin(4*pi*x)
```

8. **Write two functions m and c that return the conditional mean and covariance. These functions take as inputs the scalar/vector of prediction point(s) x, the DoE vector X, the vector of responses Y, a kernel function kern, and the vector param.**

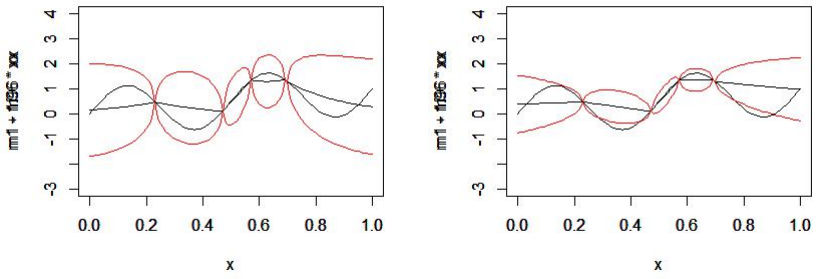
```r
m<-function(k,x,xtest,ytest,param){
  # m.result<-k(x,xtest,param)%*%solve(k(xtest,xtest,param)+1e-6*diag(length(xtest)))%*%(ytest)
  m.result<-k(x,xtest,param)%*%solve(k(xtest,xtest,param))%*%(ytest)
  return(m.result)
}
cfunction<-function(k,x,y,xtest,param){
  # c.result<-k(x,y,param)-k(x,xtest,param)%*%solve(k(xtest,xtest,param)+1e-6*diag(length(xtest)))%*%k(xtest,y,param)
  c.result<-k(x,y,param)-k(x,xtest,param)%*%solve(k(xtest,xtest,param))%*%k(xtest,y,param)
  return(c.result)
}
```

If these 2 functions can not work, we use the part on green.

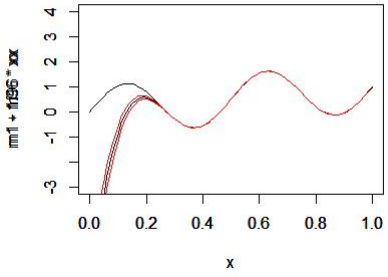
9. **Draw on the same graph f (x), m(x) and 95% confidence intervals: m(x) ± 1.96sqrt(c(x, x)).**

```r
m1<-m(k,x,xtest,ytest,param)
c1<-cfunction(k,x,x,xtest,param)
xx<-matrix(0,ncol=1,nrow = length(c1[,1]))
xx[1]<-sqrt(c1[1,1])
for(i in 1:length(xx)){
  xx[i]<-sqrt(c1[i,i])
}
xx[is.na(xx)]<-0
plot(x,f,type = "l",xlim = c(0,1),ylim=c(-3,4))
par(new=TRUE)
plot(x,m1,type = "l",xlim = c(0,1),ylim=c(-3,4))
par(new=TRUE)
plot(x,m1-1.96*xx,type = "l",xlim = c(0,1),ylim=c(-3,4),col='red')
par(new=TRUE)
plot(x,m1+1.96*xx,type = "l",xlim = c(0,1),ylim=c(-3,4),col='red')
```

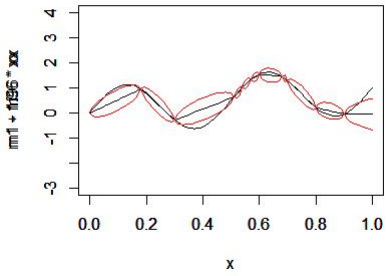
We use the function exponential as an example with different theta:



We use the function sinc as an example with different theta:



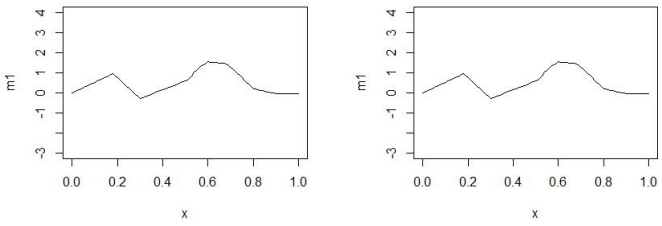
And function Brownian:



10. Change the kernel as well as the values in param. What is the effect of

- **σ2 on m(x)? Can you prove this result?**
- **σ2 on the conditional variance var(x) = c(x, x)? Can you prove this result?**
- **θ on m(x) (try (very) small and large values)?**
- **θ on v(x) (try (very) small and large values)?**

1)When we change the sigma and then we get:

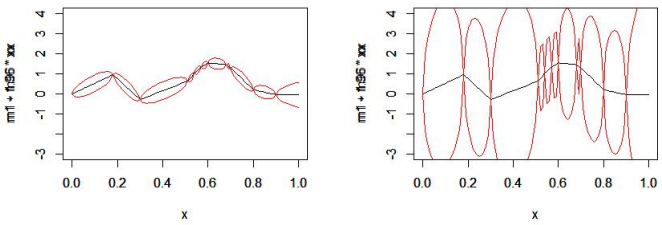


We can see there is no effect with changing sigma.

$$\mathrm{m(x)} = \mathrm{k1} * \mathrm{k2}^{-1} * \mathrm{Y}$$

there is an σ 2 in k1 and an 1/ σ 2 in $\mathrm{k2}^{-1}$, so when we calculate, these two will offset each other and will have no effect on m.

2)

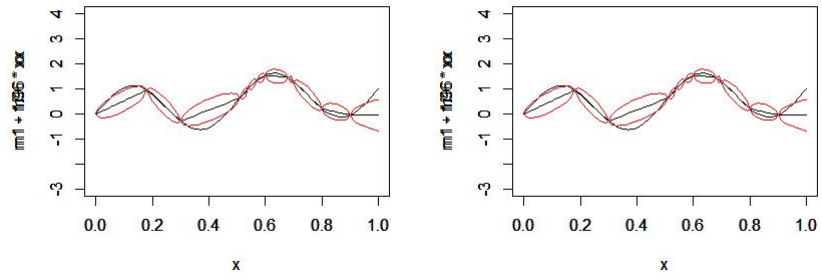


We can find in the images, c will amplify with the enlargement of σ 2.

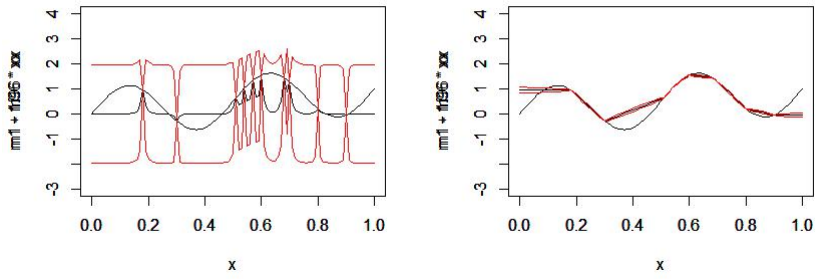
$$c(x,x) = k1 - k2 * k3^{-1} * k4$$

the σ 2 in k2 and $k3^{-1}$can offset each other and there will leave one in k4 and one in k1 so we can say $c \propto \sigma^2$

3)theta=0.01 vs theta=100



No effect on m(x)



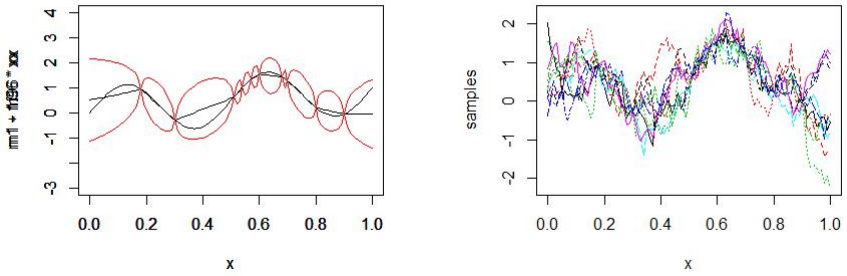
For this function, it will affect c(x,x)

We can say it depends on the function.

4)we can also see the images in question 3), some will be affected when we change the theta, and some will not change, this will depend on the model of kernel used.

11. Generate samples from the conditional process.

```
samples<-t(mvrnorm(100,mu=m1,Sigma=c1))
matplot(x,samples[,seq(10)],type = "l",ylab="samples")
```



->

12. **Bonus question. After testing different kernels and various values for σ2 and θ, which one would you recommend?**

    I will recommend the model exponential, I have done the test with sigma smaller than smaller, when sigma=1, exponential and Brownian both work well, all the f can be include in the confident interval. And then I test sigma=0.5, exp can also work well, and its efficacy will change little with sigma and theta. So I will recommend this model, because it is stable.