**Solution TP Méta modélisation par processus gaussiens**

library(DiceKriging)

N <- 200 #nb of points in the trajectory

x <- seq(0,1,length.out=N)

dist\_all <- as.matrix(dist(x))

n <- 10 # simulation of n trajectories

Z1 <- matrix(rnorm(N\*n),N,n) # simulation of a gaussian centred vector with covariance matricx identity

f0 = rep(0,N)

# gaussian covariance

sig2 <- 0.5

theta <- 0.05

## theta <- 0.2

lambda <- 10^(-10)

cov = sig2\*(exp(-(dist\_all/theta)^2) + lambda \* diag(1,N)) # cov between x

Z2 <- matrix(0,N,n)

for (i in 1:n){

Z2[,i] <- f0+t(chol(cov))%\*%Z1[,i]

}

plot(x,Z2[,1], main=paste('Simulation of GP trajectories with gaussian covariance and theta =',theta),type='l',ylim=c(min(Z2),max(Z2)))

for (i in 2:n)

lines(x,Z2[,i],col=i)

# exponential covariance

sig2 <- 0.5

theta <- 0.05

cov <- sig2\*exp(-(dist\_all/theta))

Z2 <- Z1

for (i in 1:n)

Z2[,i] <- f0+t(chol(cov))%\*%Z1[,i]

plot(x,Z2[,1], main=paste('Simulation of GP trajectories with exponential covariance and theta =',theta),type='l',ylim=c(min(Z2),max(Z2)))

for (i in 2:n)

lines(x,Z2[,i],col=i)

# gaussian covariance with nugget effect

sig2 <- 0.5

theta <- 0.2

lambda <- 0.05

cov <- sig2\*(exp(-(dist\_all/theta)^2) + lambda \* diag(1,N))

Z2 <- Z1

for (i in 1:n)

Z2[,i] <- f0+t(chol(cov))%\*%Z1[,i]

plot(x,Z2[,1], main=paste('Simulation of GP trajectories with gaussian covariance and theta =',theta),type='l',ylim=c(min(Z2),max(Z2)))

for (i in 2:n)

lines(x,Z2[,i],col=i)

##### or with DiceKriging

type <- "exp"

coef <- c(theta = 0.05)

sigma <- 0.5

model <- km(design=x, response=rep(0,N),coef.trend = 0,covtype=type, coef.cov=coef, coef.var=sigma)

Z2 <- simulate(model, nsim=10, newdata=NULL)

plot(x,Z2[1,], main=paste('Simulation of GP trajectories with exponential covariance and theta =',theta),type='l',ylim=c(min(Z2),max(Z2)))

for (i in 2:n)

lines(x,Z2[i,], main=paste('Simulation of GP trajectories with exponential covariance and theta =',theta))

type <- "gauss"

lambda <- 10^(-10) # diagonal term to outperform the conditioning of the covariance matrix

model <- km(design=x, response=rep(0,N),coef.trend = 0,covtype=type, coef.cov=coef, coef.var=sigma,nugget=lambda)

Z2 <- simulate(model, nsim=10, newdata=NULL)

plot(x,Z2[1,], main=paste('Simulation of GP trajectories with exponential covariance and theta =',theta),type='l',ylim=c(min(Z2),max(Z2)))

for (i in 2:n)

lines(x,Z2[i,], main=paste('Simulation of GP trajectories with exponential covariance and theta =',theta))

###############################################################

## Exercice 1.b : simulations of 2D gaussian processes ##

###############################################################

rm(list=ls(all=TRUE))

# Parameters initialization

d = 2 # inputs space dimension

N\_dim = 50 # nb of points in each dimension of the inputs space

N\_tot = N\_dim^d

x <- as.matrix(expand.grid(seq(0,1,length.out=N\_dim),seq(0,1,length.out=N\_dim)))

mu <- 0

sig2 <- 0.5

dist1 <- as.matrix(dist( x[,1] ))

dist2 <- as.matrix(dist( x[,2] ))

# isotropic Gaussian covariance

theta = 0.1

lambda <- 10^(-10) # diagonal term to outperform the conditioning of the covariance matrix

cov = sig2\*(exp(-((dist1/theta)^2 +(dist2/theta)^2))+ lambda \* diag(1,N\_tot))

f0 = x %\*% c(0,0)

n <- 1 # simulation of n trajectories

Z1 <- matrix(rnorm(N\_tot\*n),N\_tot,n) # simulation of a centred gaussian vector with covariance matrix identity

Z2=f0+t(chol(cov))%\*%Z1 # then linear transformation

par(mfrow=c(1,1))

filled.contour(matrix(Z2,N\_dim,N\_dim),main=paste('Simulation trajectoires PG de covariance gaussienne isotrope avec theta =', theta),color.palette=heat.colors)

# anisotropic gaussian covariance

theta1 = 0.1

theta2 = 0.03

cov = sig2\*(exp(-((dist1/theta1)^2 +(dist2/theta2)^2)) + lambda \* diag(1,N\_tot))

f0 = x %\*% c(0,0)

Z2=f0+t(chol(cov))%\*%Z1 # then linear transformation

filled.contour(matrix(Z2,N\_dim,N\_dim),main=paste('Simulation of GP trajectories with anisotropic gaussian covariance and theta1 =', theta1,'& theta2=',theta2),color.palette=heat.colors)

# anisotropic exponential covariance

cov = sig2\*exp(-(dist1/theta1 + dist2/theta2))

f0 = x %\*% c(0,0)

Z2=f0+t(chol(cov))%\*%Z1 # then linear transformation

filled.contour(matrix(Z2,N\_dim,N\_dim),main=paste('Simulation of GP trajectories with anisotropic exponential covariance and theta1 =', theta1,'& theta2=',theta2),color.palette=heat.colors)

rm(list=ls(all.names=TRUE))

library(lhs)

library(DiceKriging)

library(DiceView)

######################## Learning basis and test basis ##########################

N\_BA = 15 ## Number of points in the learning sample (BA)

d = 1 ## Dimension of the inputs space

S\_BA = optimumLHS(N\_BA,d) ## Simulation os the learning basis BA by optimized LHS => use the function optimumLHS

N\_BT = 100 ### Number of points in the test basis

S\_BT = seq(0,1,length.out=N\_BT) ### Simulation of an initial test basis

## S\_BT : inputs for the test basis

## Y\_BT : output of the emulator

fx\_exo2 <- function(X)

sin(30\*(X-0.9)^4)\*cos(2\*(X-0.9)) + (X-0.9)/2

Y\_BA <- fx\_exo2(S\_BA) ## Y\_BA : output of the emulator on the learning basis BA

Y\_BT <- fx\_exo2(S\_BT) ## Y\_BT :output of the emulator on the test basis BT

plot(S\_BT,Y\_BT,type='l')

points(S\_BA,Y\_BA,pch=3,col=2)

legend("topright",y=c(0.17,0.33),c('Theoretical function','Pts in the learning basis'),lty=c(1,0),

pch=c(-1,3),col=c(1,2),cex=0.8)

metamodel <- km(formula=~1,design=S\_BA,response=Y\_BA,covtype="matern5\_2") # Construction of the Gaussian Process emulator

prediction <- predict.km(metamodel,S\_BT,'UK',checkNames=FALSE) # Prediction of the metamodel on the test basis => use of the function predict.km

par(mfrow=c(2,1))

plot(S\_BT,Y\_BT,type='l',main='Estimation of f by the GP metamodel')

points(S\_BA,Y\_BA,pch=3,col='red')

lines(S\_BT,prediction$mean,col='blue')

lines(S\_BT,prediction$lower95,col='blue',lty=2)

lines(S\_BT,prediction$upper95,col='blue',lty=2)

plot(S\_BT,prediction$sd,type='l',main="sqrt(MSE) of the estimation")

# with DiceView

par(mfrow=c(2,1))

sectionview(metamodel)

plot(S\_BT,prediction$sd,type='l',main="sqrt(MSE) of the estimation")

#### Computation of Q2 on BT ####

# Building of the function allowing to compute Q2

Q2 <- function(y,yhat)

1-mean((y-yhat)^2)/var(y)

Q2\_BT <- Q2(Y\_BT,prediction$mean) # Computation of Q2 on the test basis

#### Display of the results ####

print(paste('Number of points in the learning sample :',N\_BA))

print(paste('Inputs space dimension :',d))

print(paste('Q2 on the test basis :',Q2\_BT))

#### Optional part ####

## Etape 3 : Computation of the likelihood and optimization of hyperparameters

x <- seq(0.01,1,length=1000)

y <- sapply(x,logLikFun,model=metamodel)

plot(x,y,type='l',main='Log-likelihood with respect to theta')

theta\_optim <- x[which.max(y)] # Identification of optimal theta

## Etape 4 : adaptive construction of the design of experiments

par(mfrow=c(2,1))

sectionview(metamodel,xlim=c(0,1))

plot(S\_BT,prediction$sd,type='l',main="sqrt(MSE) of the estimation")

## Adaptive design : one iteration

maxi <- which.max(prediction$sd) # Research of the point where MSE is maximal

points(S\_BT[maxi],prediction$sd[maxi],pch=16,col=3)

S\_BA2 <- rbind(S\_BA,S\_BT[maxi]) # add the new point to the learning basis BA

Y\_BA2 <- c(Y\_BA,fx\_exo2(S\_BT[maxi])) # add the corresponding output Y

metamodel2 <- km(formula=~1,design=S\_BA2,response=Y\_BA2,covtype="matern5\_2") # Update the GP metamodel

prediction2 <- predict.km(metamodel2,S\_BT,'UK',checkNames=FALSE) # Update the predictions and the MSE of the metamodel

# Display of the results

par(mfrow=c(2,1))

sectionview(metamodel2)

plot(S\_BT,prediction2$sd,type='l',main="sqrt(MSE) of the estimation when adding one point")

points(S\_BT[maxi],prediction2$sd[maxi],pch=16,col=3)

## Adaptive design : Automation of the process

nIte <- 15

for (i in 1:nIte){

maxi <- which.max(prediction2$sd) # Research of the point where MSE is maximal

S\_BA2 <- rbind(S\_BA2,S\_BT[maxi]) # add the new point to the learning basis BA

Y\_BA2 <- c(Y\_BA2,fx\_exo2(S\_BT[maxi]))

capture.output(metamodel2 <- km(formula=~1,design=S\_BA2,response=Y\_BA2,covtype="matern5\_2")) # Update the GP metamodel

prediction2 <- predict.km(metamodel2,S\_BT,'UK',checkNames=FALSE) # Update predictions and MSE

# Affichage

par(mfrow=c(2,1))

sectionview(metamodel2)

plot(S\_BT,prediction2$sd,type='l',main="sqrt(MSE) wahen adding one point")

points(S\_BT[maxi],prediction2$sd[maxi],pch=16,col=3)

Sys.sleep(2)

}

Q2\_BTnew <- Q2(Y\_BT,prediction2$mean)

rm(list=ls(all.names=TRUE))

library(lhs)

library(DiceKriging)

######################## learning basis and test basis ##########################

N\_BA = 80; ## Number of points in the learning basis

d = 2; ## Inputs space dimension

S\_BA = optimumLHS(N\_BA,d); ## Construction of the learning basis

## S\_BA : inputs

## Y\_BA : outputs of the emulator

N\_BT\_dim = 70;

N\_BT = N\_BT\_dim^2; ### Number of points in the test basis

S\_BT\_dim <- seq(0,1,length.out=N\_BT\_dim)

S\_BT <- expand.grid(S\_BT\_dim,S\_BT\_dim) ### Construction of an initial test basis

## S\_BT : inputs in the test basis

## Y\_BT : corresponding emulator's outputs

schwefel <- function(x){

x2 <- x\*400-200

rowSums(-x\*sin(sqrt(abs(x2))))

}

Y\_BA <- schwefel(S\_BA)

Y\_BT <- schwefel(S\_BT)

par(mfrow=c(1,1))

filled.contour(S\_BT\_dim,S\_BT\_dim,matrix(Y\_BT,N\_BT\_dim,N\_BT\_dim),main='Function Schwefel',xlab='X1',ylab='X2',color.palette=heat.colors,

plot.axes={points(S\_BA[,1],S\_BA[,2],pch=3)})

######################### METAMODELING #############################

metamodel <- km(design=S\_BA,response= Y\_BA, covtype="matern3\_2")

# ###### Prediction on BT with GP metamodel #######

predictions\_BT<- predict(metamodel,S\_BT,'UK',checkNames=F)

filled.contour(S\_BT\_dim,S\_BT\_dim,matrix(predictions\_BT$mean,N\_BT\_dim,N\_BT\_dim),xlab='X1',ylab='X2',main='Prediction du metamodele',color.palette=heat.colors)

filled.contour(S\_BT\_dim,S\_BT\_dim,matrix(abs(Y\_BT - predictions\_BT$mean),N\_BT\_dim,N\_BT\_dim),xlab='X1',ylab='X2',main='Erreur du metamodele PG (valeur absolue)',color.palette=heat.colors)

# ###### Computation of Q2 on BT #########

Q2 <- function(y,yhat)

1-mean((y-yhat)^2)/var(y)

Q2\_BT = Q2(Y\_BT,predictions\_BT$mean) # Computation of Q2 on the test basis

#### Results' display ####

print(paste('Number of points in the test basis:',N\_BA))

print(paste('Inputs space dimension:',d))

print(paste('Q2 on the test basis:',Q2\_BT))

rm(list=ls(all.names=TRUE))

library(lhs)

library(DiceKriging)

library(rgl)

library(sensitivity)

library(boot)

library(numbers)

#### Learning basis and test basis ####

N\_BA <- 150; ## Number of points in the learning basis

d <- 3; ## Inputs space dimension

S\_BA <- randomLHS(N\_BA,d) ## Construction of the learning basis

## S\_BA : inputs for the learning basis

## Y\_BA : corresponding outputs of the emulator

S\_BT <- randomLHS(100,3) ## Construction of the test basis

ishigami <- function(X,coeff){

## Analytical Function of Ishigami for scaled inputs => inputs in [0:1]

## Ishigami function : Y = sinX1+a\*(sinX2).^2 + b\*X3.^4\*sinX1 for Xi ~ U[-pi;+pi]

## with a = coeff(1); b=coeff(2)

## Generally : coeff = [7 0.1];

X <- 2\*pi\*X-pi

sin(X[,1]) + coeff[1]\*sin(X[,2])^2 + coeff[2]\*X[,3]^4\*sin(X[,1])

}

coeff\_fx\_test = c(7, 0.1)

Y\_BA <- ishigami(S\_BA,coeff\_fx\_test)

Y\_BT <- ishigami(S\_BT,coeff\_fx\_test)

#### METAMODELING ####

metamodel <- km(formula=~1,design=S\_BA,response=Y\_BA,covtype="matern5\_2")

plot(metamodel) # Affichage des resultats

## Computation of Q2 on the test basis

Q2 <- function(y,yhat)

1-mean((y-yhat)^2)/var(y)

prediction <- predict(metamodel, S\_BT, 'UK', checkNames=FALSE)$mean

Q2\_BT<-Q2(Y\_BT,prediction)

print(paste('Q2 sur Base de test :',Q2\_BT))

## Computation of Q2 by cross validation

prediction\_LOO <- leaveOneOut.km(metamodel,'UK')$mean

Q2\_LOO<- Q2(Y\_BA,prediction\_LOO)

print(paste('Q2 sur Base de test :',Q2\_LOO))

#### Estimation of first-order and total Sobol' indices ####

# Building of the gaussian process metamodel used to compute Sobol' indices

f <- function(x)

predict(metamodel,x,type='UK',checkNames=FALSE)$mean

## Computation of Sobol' indices by SobolEff Janon et al. (2014)

n <- 1000

sample1 <- data.frame(X1=runif(n,0,1),X2=runif(n,0,1),X3=runif(n,0,1))

sample2 <- data.frame(X1=runif(n,0,1),X2=runif(n,0,1),X3=runif(n,0,1))

indices.sobol <- sobolEff(f,sample1,sample2,order=1)

indices.sobol.totaux <- sobolEff(f,sample1,sample2,order=0)

indices.sobol

indices.sobol.totaux

## Computation of first- and closed second-order Sobol' indices with replicated designs

indices.sobol.rep.o1 <- sobolroalhs(model = f, factors = 3, N = 1000, order = 1, nboot=100)

print(indices.sobol.rep.o1)

plot(indices.sobol.rep.o1)

indices.sobol.rep.o2 <- sobolroalhs(model = f, factors = 3, N = 1000, order = 2, nboot=100)

print(indices.sobol.rep.o2)

plot(indices.sobol.rep.o2)

## Computation of Sobol' indices by Fast (Saltelli et al. [1999]),spectral approach, which requises regularity

## in the sense fast decrease of Fourier coefficients

n\_RBDFast <- 1000

M\_RBDFast <- 4

indices.fast <- fast99(model=f,factors=d,n=n\_RBDFast,M=M\_RBDFast,q='qunif',q.arg=list(min=0,max=1))

#### Theoretical values ####

ishigami.sobol.indices <- function(coeff){

D <- (coeff[1]^2)/8 + coeff[2]\*(pi^4)/5 + coeff[2]^2\*(pi^8)/18 + 0.5

D1 = coeff[2]\*(pi^4)/5 + coeff[2]^2\*(pi^8)/50 + 0.5

D2 = (coeff[1]^2)/8

D13 = coeff[2]^2\*(pi^8)\*(1/18-1/50)

S1 <- St <- array(0,3)

S1[1] = D1/D

S1[2] = D2/D

S1[3] = 0

St[1] = (D1+D13)/D

St[2] = D2/D

St[3] = D13/D

list(S11=S1[1],S12=S1[2],S13=S1[3],St11=St[1],St12=St[2],St13=St[3])

}

# Calcul des indices th?oriques

indices.theo <- ishigami.sobol.indices(coeff\_fx\_test)

# Comparaison des indices estim?s et th?oriques

print('Theoretical indices :')

indices.theo

print('First-order and total indices estimated by SobolEff :')

indices.sobol

indices.sobol.totaux

print('First-order and total indices estimated by fast1999 :')

indices.fast

print('First-order indices estimated by sobolroalhs :')

indices.sobol.rep.o1

print('Closed second-order indices estimated by sobolroalhs :')

indices.sobol.rep.o2

par(mfrow=c(1,2))

plot(indices.sobol)

abline(h=indices.theo$S11)

abline(h=indices.theo$S12)

abline(h=indices.theo$S13)

plot(indices.sobol.totaux)

abline(h=indices.theo$St11)

abline(h=indices.theo$St12)

abline(h=indices.theo$St13)

## Play with N\_BA (predictivity of the metamodel) and

## with n (the samples size for estimating DSobol' indices)

# valeurs thÃ©oriques S3=0,S2 environ Ã©gal Ã  0.4424, S1 environ Ã©gal Ã  0.3139