

Supplemental Data 3

R scripts to perform the selection and validation of the calibration model

A video “user guide” containing instructions on how to setup and execute the R scripts is available at <https://youtu.be/azpD2GG0qNA>. Please watch it before using the R scripts.

The “.R” files ready for use are available at <https://bit.ly/2V1LVPx>.

```
1 #####
2 ## File to be run ##
3 #####
4
5 ## Parameters to be set
6 directory <- "~/Desktop/Bridge/Calibration" # set work directory : folder where the
   calibration data is
7
   # and the two R sheets Calibration2Ks and
   Calibration2CVM
8 filename <- "Model.txt" # name of the file that contains the raw data (calibration data)
9 result_filename <- "My_Results.txt" # name of the file that contains the results
10 dec <- "." # , if the decimals are ex. 0,5 . if the decimals are ex 0.5
11 nbgrille = 50 # global paramter for integral approximations
12 b <- 1000 #global paramter for bootstraping
13 stat <- 2 # 1: KS, 2: CVM
14 alpha <- 0.05 ## The alpha to use in hypothesis testing
15 ## end of parameter setting
16
17
18 #####
19 ## Code ##
20 #####
21 source("CodeV5.R") ## Contains the main coded functions
22 A <- read.table(filename,header=F,dec=dec) ## reading the data
23 A <- as.matrix(A)
24 stats = c("Kolmogorov Smirnov","Cramer von Mises")
25
26
27 Decision <- rep(0,3)
28 DCV <- rep(0,3)
29
30 # weight : 0
31 weight <- 0;
32 P0 = fitPlus(A, weight , b, nbgrille , stat)
33 coP0 <- matrix(c(rev(P0$Linear$param),0,rev(P0$Quadratic$param)),ncol=3,byrow=T) ## Contains
   the fitted coefficients
34 colnames(coP0) <- c("b0", "b1", "b2")
```

```

35 rownames(coP0) <- c("Linear", "Quadratic")
36
37
38 List0 <- list(Fitted_Param = coP0, Pval_Lin = P0$Linear$pvalN, Stat_Lin = P0$Linear$statN ,
39             Pval_Quad = P0$Quadratic$pvalN, Stat_Quad = P0$Quadratic$statN, Pval_PartialF = P0$
           pvalFtest)
40
41 Decision[1] = 1 + (List0$Pval_PartialF < 0.05)
42
43 # weight : 1
44 weight <- 1;
45 P1 = fitPlus(A, weight, b, nbgrille, stat)
46 coP1 <- matrix(c(rev(P1$Linear$param), 0, rev(P1$Quadratic$param)), ncol=3, byrow=T) ## Contains
           the fitted coefficients
47 colnames(coP1) <- c("b0", "b1", "b2")
48 rownames(coP1) <- c("Linear", "Quadratic")
49
50
51 List1 <- list(Fitted_Param = coP1, Pval_Lin = P1$Linear$pvalN, Stat_Lin = P1$Linear$statN ,
52             Pval_Quad = P1$Quadratic$pvalN, Stat_Quad = P1$Quadratic$statN, Pval_PartialF = P1$
           pvalFtest)
53
54 Decision[2] = 1 + (List1$Pval_PartialF < 0.05)
55
56 # weight : 2
57 weight <- 2;
58 P2 = fitPlus(A, weight, b, nbgrille, stat)
59 coP2 <- matrix(c(rev(P2$Linear$param), 0, rev(P2$Quadratic$param)), ncol=3, byrow=T) ## Contains
           the fitted coefficients
60 colnames(coP2) <- c("b0", "b1", "b2")
61 rownames(coP2) <- c("Linear", "Quadratic")
62
63
64 List2 <- list(Fitted_Param = coP2, Pval_Lin = P2$Linear$pvalN, Stat_Lin = P2$Linear$statN ,
65             Pval_Quad = P2$Quadratic$pvalN, Stat_Quad = P2$Quadratic$statN, Pval_PartialF = P2$
           pvalFtest)
66
67 Decision[3] = 1 + (List2$Pval_PartialF < 0.05)
68
69 ## Variance Test for weight selection
70 Ano = A[, -1] / sqrt(A[, 1])
71 spoids = sum(1 / sqrt(A[, 1]))
72 var1 = apply(Ano / spoids, 1, var)
73 Ano2 = A[, -1] / A[, 1]
74 spoids2 = sum(1 / (A[, 1]))
75 var2 = apply(Ano2 / spoids2, 1, var)
76 var0 = apply(A[, -1] / length(A[, 1]), 1, var)
77 s1 = signif(sum((var0 - mean(var0))^2), 3)
78 s2 = signif(sum((var1 - mean(var1))^2), 3)
79 s3 = signif(sum((var2 - mean(var2))^2), 3)
80 sss = c(s1, s2, s3)
81 z1 = paste("Variance test for weight selection")
82 z2 = paste("Scores:", "No weight:", s1, "x^(-1):", s2, "x^(-2):", s3)
83 ccc = c("no weight", "x^(-1)", "x^(-2)")

```

```

84 ccb = paste("Selected weight: ",ccc[which(sss==min(sss))],sep="")
85 zaa = paste(z1,z2,ccb,sep="\n")
86
87 best <- which(sss==min(sss))
88
89 awiner = eval(parse(text = paste("List",best-1,sep="")))
90
91 ## Ftest Hétéro
92 pop1 = A[1,-1]
93 pop2 = A[nrow(A),-1]
94 ttt = var.test(pop1,pop2,alternative = "less")
95 bb = paste("F-test for heteroscedasticity")
96 bb1 = paste("p-value: ",signif(ttt$p.value,4))
97 ddd = "No"
98 if(ttt$p.value<alpha){ddd="Yes"}
99 bb2 = paste("Weighting needed:",ddd,sep=" ")
100 pop1 = paste(bb,bb1,bb2,sep="\n")
101
102
103 v = paste("Partial F-test for model order selection")
104 v1 = paste("p-value",signif(awiner$Pval_PartialF,4),sep=": ")
105 v2 = paste("Model selected: ",c("linear","quadratic")[Decision[best]],sep="")
106 vv = paste(v,v1,v2,sep="\n")
107
108
109 w = paste("Normality of the standerized residuals")
110 w1 = paste("Test used: ",stats[stat],sep="")
111 aw= eval(parse(text = paste("awiner$Pval_",c("Lin","Quad")[Decision[best]],sep="")))
112 w2 = paste("p-value: ",signif(aw,4))
113 ww = "No"
114 if(alpha<aw)
115 {
116   ww = "Yes"
117 }
118 ww1 = paste("Validation test passed:",ww,sep=" ")
119 vw =paste(w,w1,w2,ww1,sep="\n")
120
121 md = c("Linear","Quadratic")[Decision[best]]
122 c("1","1/x","1/x^2")[best]
123 if(ww=="Yes")
124 {
125   zw=paste("Model selected: ",md,", ",c("1","1/x","1/x^2")[best],sep=" ")
126   zw2 = paste("Calibration equation:")
127   if(md=="Linear")
128   {
129     co1 = awiner$Fitted_Param[1,1]
130     co2 = awiner$Fitted_Param[1,2]
131     zw1 = paste(signif(co2,4)," x + ",signif(co1,4),sep="")
132   }
133   if(md=="Quadratic")
134   {
135     co1 = awiner$Fitted_Param[2,1]
136     co2 = awiner$Fitted_Param[2,2]
137     co3 = awiner$Fitted_Param[2,3]

```

```

138   zw1 = paste(signif(co3,4), " x^2 + ", signif(co2,4), " x + ", signif(co1,4), sep="")
139 }
140 vw2 = paste(zw, zw2, zw1, sep="\n")
141 }
142 if(ww == "No"){
143   vw2 = paste("No model selected, validation test failed")
144 }
145
146 ## Plots
147 namePlot = strsplit(result_filename, ".", fixed=TRUE)[[1]][1]
148 # Plot of variance
149 var_level = apply(A[, -1], 1, var)
150 pdf(file=paste(namePlot, "Variance.pdf", sep="_"))
151 plot(A[, 1], var_level, xlab="Concentration", ylab="Variance", main="Variance plot", mgp = c(2,
152   0.8, 0), axes = T)
153 dev.off()
154 ## Calibration curve
155 f <- function(x)
156 {
157   predic(rev(awiner$Fitted_Param[Decision[best],]), x)
158 }
159 Cal.dots = predic(rev(awiner$Fitted_Param[Decision[best],]), A[, 1])
160 pdf(file=paste(namePlot, "Calibration curve.pdf", sep="_"))
161 plot(rep(A[, 1], each=(ncol(A[, -1]))), c(t(A[, -1])), xlab="Concentration", ylab="Signal", main="
162   Calibration curve", mgp = c(2, 0.8, 0), axes = T)
163 curve(f, add=T)
164 dev.off()
165 mt = paste(popl, zaa, vv, vw, vw2, sep="\n\n")
166 write(mt, file=result_filename)
167 cat(mt)

```

```

1 # A : sample of data
2 # nbgrille : preision for the approximation of the grid
3 # variance to be used in the normal distribution function
4
5 ## Computes  $\sqrt{n} * \sup_x | F_n(x) - \Phi(x) |$  where Phi is the normal distribution
6   function mean 0
7 ## variance sigma, F_n is the empirical distribution function computed from A
8
9
10 library("mvtnorm")
11
12
13
14 pval <- function(dis, cv){
15   mean(cv>dis)
16 }
17
18
19 OptParam <- function(X, Y, poids, indice){
20   ## trouve les param optimaux pour indice = 1: modèle linéaire, 2 : modèle quadratique
21   W <- diag(1/abs(X)^poids)
22   xf <- matrix(rep(X, each = indice + 1), ncol=indice+1, byrow=T)
23   ex <- matrix(rep(0:(indice), length(X)), ncol=indice + 1, byrow=T)
24   Xp = xf^ex
25   matInv <- solve(t(Xp)%*%W%*%Xp)
26   param.optimaux <- rev(matInv%*%t(Xp)%*%W%*%Y)
27   return(param.optimaux)
28 }

```

```

24 predic <- function(param,x)
25 {
26   xf <- matrix(rep(x,each = length(param)),ncol=length(param),byrow=T)
27   ex <- matrix(rep(0:(length(param)-1),length(x)),ncol=length(param),byrow=T)
28   par <- matrix(rep(rev(param),length(x)),ncol=length(param),byrow=T)
29   Temp <- par * xf^ex
30   return(apply(Temp,1,sum))
31 }
32
33 CV <- function(don,indice,poids)
34 {
35   X = rep(don[,1],ncol(don[, -1]))
36   Y = c(don[, -1])
37   CVc = 0
38   corr <- (1/X^poids)
39   for(i in 1:length(Y))
40   {
41     p <- OptParam(X[-i],Y[-i],poids,indice)
42     pr <- (predic(p,X[i]) - Y[i])
43     CVc = CVc + (pr)^2 * corr[i]
44   }
45   return(CVc/sum(corr))
46 }
47
48 fit <- function(A,poids,b,nbrille,stat,indice){
49   ## A : matrix of data, column 1 contains the covariates
50   ## poids : 1/X^poids i.e poids = 0 : no additional weight, poids =2 : inverted quadratique
51   ## b : number of bootstrap replicates to use, suggestion is 1000
52   ## nbrille : number of points used to approximate the statistic, suggestion is 50
53   ## stat : 1 is Kolmogorov smirnov statistic, 2 is cramer von mises
54   ## indice : 1 is linear, 2 is quadratic 3 is cubic
55
56   don <- as.matrix(A)
57   nbr <- ncol(A)-1
58
59   W <- diag(rep(1/abs(don[,1])^poids,nbr)) # Weights
60
61 # Optimal parameters
62 X = rep(don[,1],ncol(don[, -1]))
63 Y = c(don[, -1])
64 W <- diag(1/abs(X)^poids)
65 xf <- matrix(rep(X,each = indice +1),ncol=indice+1,byrow=T)
66 ex <- matrix(rep(0:(indice),length(X)),ncol=indice +1 ,byrow=T)
67 Xp = xf^ex
68 meanXp <- apply(Xp,2,mean)
69 matInv <- solve(t(Xp)%*%W%*%Xp)
70 matt <- matInv%*%t(Xp)%*%W
71 param.optimaux <- rev(matt%*%Y)
72
73 #Residuals
74 poidsob <- 1/abs(X)^poids
75 poidsob <- poidsob/sum(poidsob)
76 epsilonp <- (predic(param.optimaux,X) - Y) * sqrt(poidsob)

```

```

77 var.param <- matInv * var(epsilon) ## variance of estimated residuals
78
79 ## Bootstrap
80 n <- length(epsilon)
81 sigma <- sqrt(var(epsilon))
82
83 ## Computationi of the data process
84 grille = (1:(nbgrille))/(nbgrille+1) ## Approximation over the grid
85 grrep <- t(matrix(rep(grille, each=n), byrow=T, ncol=n))
86 epsilonrep <- matrix(rep(epsilon, each=nbgrille), ncol=nbgrille, byrow=T)/sigma
87 Fn <- apply(pnorm(epsilonrep)<=grrep, 2, mean)
88 monproc <- sqrt(n)*(Fn - grille)
89
90 proc=matrix(rep(0, b*nbgrille), ncol=b)
91 Xb = predic(param.optimaux, X);
92 for(i in 1:b)
93 {
94   ech_boot = sample(epsilon, n, replace=T)
95   Yb = Xb + ech_boot / sqrt(poidsob)
96   param.optimauxb <- rev(matt%*%Yb)
97   epsilonpb <- (predic(param.optimauxb, X) - Yb) * sqrt(poidsob)
98   epsilonrepb <- matrix(rep(epsilonpb, each=nbgrille), ncol=nbgrille, byrow=T)/sqrt(var(
99   epsilonpb))
100   Fnb <- apply(pnorm(epsilonrepb)<=grrep, 2, mean)
101   monprocb <- sqrt(n)*(Fnb - Fn)
102   proc[, i] = monprocb
103 }
104 ## Computation of the pvalue
105 distr <- rep(0, b)
106 mastat <- 0
107 if(stat==1){ distr <- apply(abs(proc), 2, max)
108   mastat <- max(abs(monproc)) }
109 if(stat==2){ distr <- apply(proc^2, 2, mean)
110   mastat <- mean(monproc^2) }
111 quantileb = quantile(distr, 0.95)
112 pvalN = mean(distr > mastat)
113
114 liste <- list(param=param.optimaux, varparam=var.param, pvalN=pvalN, statN=mastat, qBoot=
115   quantileb, esp=epsilon)
116 }
117
118 fitPlus <- function(A, poids, b, nbgrille, stat)
119 {
120   ## Performs the fit plus the partial F test
121   fitLin <- fit(A, poids, b, nbgrille, stat, 1)
122   fitQ <- fit(A, poids, b, nbgrille, stat, 2)
123
124   SSREGL <- sum(fitLin$esp^2)
125   SSREGQ <- sum(fitQ$esp^2)
126   degf <- length(fitQ$esp)-3
127   stat <- (SSREGL-SSREGQ)/(SSREGQ/(degf))
128   pvalF <- 1-pf(stat, 1, degf)

```

```

129
130 L2 <- list(Linear= fitLin , Quadratic = fitQ , Ftest=stat , pvalFtest=pvalF)
131 return(L2)
132 }

1 ## Code KS
2
3 KS <- function(A, nbgrille , sigma){
4   # A : sample of data
5   # nbgrille : preision for the approximation of the grid
6   # variance to be used in the normal distribution function
7
8   ## Computes  $\sqrt{n} * \sup_x | F_n(x) - \Phi(x) |$  where Phi is the normal distribution
   function mean 0
9   ## variance sigma , F_n is the empirical distribution function computed from A
10
11   n<- length(A)
12   fn <- function(t){sqrt(n) * abs(mean(A <= qnorm(t,0,sd=sqrt(sigma))) - t ) }
13   max(sapply((1:nbgrille)/nbgrille , fn))
14 }
15 pval <- function(dis , cv){
16   mean(cv>dis)
17 }
18
19
20 # Linear fit using KS statistic
21 fit1KS <- function(A, poids , b, nbgrille){
22   don <- as.matrix(A)
23   nbr <- ncol(A)-1
24
25   W <- diag(rep(1/abs(don[,1])^poids , nbr)) # matrix of weights
26
27 # Finding the optimal parameters for the regression
28   x1 <- don[,1]
29   Xp <- matrix(c(rep(1, length(don[,1]) * nbr) , rep(x1, nbr)) , ncol=2)
30   matInv <- solve(t(Xp)%*%W%*%Xp)
31   param.optimaux <- rev(matInv%*%t(Xp)%*%W%*%c(don[, -1]))
32
33 # Computing the residuals
34   temp <- param.optimaux[1]*don[,1]+ param.optimaux[2]
35   temp2 <- matrix(rep(temp, nbr) , nrow=length(don[,1]))
36   temp3 <- (don[, -1] - temp2)
37
38   poidsob <- 1/abs(don[,1])^poids
39   poidsob <- poidsob/sum(poidsob)
40   epsilonp <- c(temp3)*(sqrt(rep(poidsob, nbr))) # residuals
41
42   var.param <- matInv * var(epsilonp) # the variance of the estimated paramters
43
44 ## Bootstrap
45   n <- length(epsilonp)
46   sigma <- sqrt(var(epsilonp))
47   distrCv <- rep(0, b)
48
49   for(j in 1:b){

```

```

50
51 B <- rnorm(n,0,sqrt(sigma))
52 C <- rmvnorm(1,mean=c(0,0),var.param)
53 drift <- (C[1] + C[2]*mean(don[,1])) # Drift function to used when esimated residuals are
    involved
54 n<- length(A)
55 fn <- function(t){
56     sqrt(n) * abs(mean(A <= qnorm(t,0,sd=sqrt(sigma))) -
57         t + drift*dnorm(qnorm(t,0,sqrt(sigma)),0,sd=sqrt(sigma))/sqrt(n)) }
58 distrCv[j] <- max(sapply((1:nbgrille)/nbgrille,fn))
59 }
60 cv0 <- KS(epsilon, nbgrille, sigma)
61 pvalN <- pval(cv0, distrCv)
62
63 liste <- list(param=param.optimaux, res1=cbind(rep(poidsob, nbr), c(temp3)), epsp=epsilon,
    varparam=var.param, pvalN=pvalN)
64 return(liste)
65 }
66
67 #quadratic fit using KS statistic
68 fit2KS <- function(A, poids, b, nbgrille){
69 don <- as.matrix(A)
70 nbr <- ncol(A)-1
71
72 W <- diag(rep(1/abs(don[,1])^poids, nbr)) # Weights
73
74 # Optimal parameters
75 x1 <- don[,1]
76 Xp <- matrix(c(rep(1, length(don[,1])*nbr), rep(x1, nbr), rep(x1^2, nbr)), ncol=3)
77 matInv <- solve(t(Xp)%*%W%*%Xp)
78 param.optimaux <- rev(matInv%*%t(Xp)%*%W%*%c(don[, -1]))
79
80 #Residuals
81 temp <- param.optimaux[1]*don[,1]^2 + param.optimaux[2]*don[,1] + param.optimaux[3]
82 temp2 <- matrix(rep(temp, nbr), nrow=nrow(don))
83 temp3 <- (don[, -1] - temp2)
84
85 poidsob <- 1/abs(don[,1])^poids
86 poidsob <- poidsob/sum(poidsob)
87 epsilon <- c(temp3)*(sqrt(rep(poidsob, nbr)))
88 var.param <- matInv * var(epsilon)
89
90 ## Bootstrap
91 n <- length(epsilon)
92 sigma <- sqrt(var(epsilon))
93 distrCv <- rep(0,b)
94
95 for(j in 1:b){
96 B <- rnorm(n,0,sqrt(sigma))
97 C <- rmvnorm(1,mean=c(0,0,0),var.param)
98 drift <- (C[1] + C[2]*mean(don[,1]) + C[3]*mean(don[,1]^2)) # Drift function to used when
    esimated residuals are involved
99 n<- length(A)
100 fn <- function(t){

```



```

101     sqrt(n) * abs(mean(A <= qnorm(t,0,sd=sqrt(sigma))) -
102         t + drift*dnorm(qnorm(t,0,sqrt(sigma)),0,sd=sqrt(sigma))/sqrt(n)) }
103     distrCv[j] <- max(sapply((1:nbgrille)/nbgrille,fn))
104 }
105 cv0 <- KS(epsilonp,nbgrille,sigma)
106 pvalN <- pval(cv0,distrCv)
107 pvalN <- pval(cv0,distrCv)
108
109 liste <- list(param=param.optimaux,varparam=var.param,pvalN=pvalN)
110 return(liste)
111 }

1 ## Code CVM
2 CVM <- function(A,nbgrille,sigma)
3 {
4   # A : sample of data
5   # nbgrille : preision for the approximation of the grid
6   # variance to be used in the normal distribution function
7
8   ## Computes sqrt(n) * \integral( F_n(x) - Phi(x) )^2 dx where Phi is the normal
9   ## distribution function mean 0
10  ## variance sigma, F_n is the empirical distribution function computed from A
11  cv =0
12  n <- length(A)
13  for(i in 1:nbgrille){
14    cv <- cv + n/nbgrille * (mean(A <= qnorm(i/nbgrille,mean=0,sd=sqrt(sigma)))- i/
15      nbgrille )^2
16  }
17  cv
18 }
19
20
21
22 #Linear fit with CVM statistic
23 fit1CVM <- function(A,poids,b,nbgrille){
24 don <- as.matrix(A)
25 nbr <- ncol(A)-1
26 nbniv <- nrow(don)
27
28 # Weights
29 W <- diag(rep(1/abs(don[,1])^poids,nbr))
30 # Optimal parameters
31 x1 <- don[,1]
32 Xp <- matrix(c(rep(1,length(don[,1])*nbr),rep(x1,nbr)),ncol=2)
33 matInv <- solve(t(Xp)%*%W%*%Xp)
34 param.optimaux <- rev(matInv%*%t(Xp)%*%W%*%c(don[,-1]))
35
36 #Residuals
37 temp <- param.optimaux[1]*don[,1]+ param.optimaux[2]
38 temp2 <- matrix(rep(temp,nbr),nrow=length(don[,1]))
39 temp3 <- (don[,-1] - temp2)
40
41 poidsob <- 1/abs(don[,1])^poids

```

```

42 poidsob <- poidsob/sum(poidsob)
43 epsilonp <- c(temp3)*(sqrt(rep(poidsob, nbr)))
44
45 var.param <- matInv * var(epsilonp)
46
47 ## Bootstrap
48 n <- length(epsilonp)
49 sigma <- sqrt(var(epsilonp))
50 distrCv <- rep(0,b)
51
52 for(j in 1:b){
53
54   B <- rnorm(n,0,sqrt(sigma))
55   C <- rmvnorm(1,mean=c(0,0),var.param)
56   drift <- (C[1] + C[2]*mean(don[,1])) # Drift function to used when esimated residuals are
      involved
57   cv <- 0
58   for(i in 1:nbgrille){
59     cv <- cv + n/nbgrille * (mean(B <= qnorm(i/nbgrille,mean=0,sd=sqrt(sigma)))- i/
      nbgrille + drift*dnorm(qnorm(i/nbgrille,0,sqrt(sigma)),0,sd=sqrt(sigma))/sqrt(n))^2
60   }
61   distrCv[j] <- cv
62 }
63 cv0 <- CVM(epsilonp, nbgrille, sigma)
64 pvalN <- pval(cv0, distrCv)
65
66 liste <- list(param=param.optimaux, varparam=var.param, pvalN=pvalN)
67 return(liste)
68 }
69
70 # Quadratic fit statistic CVM
71 fit2CVM <- function(A, poids, b, nbgrille){
72 don <- as.matrix(A)
73 nbr <- ncol(A)-1
74 nbniv <- nrow(A)
75
76
77 #Weights
78 W <- diag(rep(1/abs(don[,1])^poids, nbr))
79 #Optimal paramters
80 x1 <- don[,1]
81 Xp <- matrix(c(rep(1, length(don[,1])*nbr), rep(x1, nbr), rep(x1^2, nbr)), ncol=3)
82 matInv <- solve(t(Xp)%*%W%*%Xp)
83 param.optimaux <- rev(matInv%*%t(Xp)%*%W%*%c(don[,-1]))
84
85
86 #Residuals
87 temp <- param.optimaux[1]*don[,1]^2 + param.optimaux[2]*don[,1] + param.optimaux[3]
88 temp2 <- matrix(rep(temp, nbr), nrow=nbniv)
89 temp3 <- (don[,-1] - temp2)
90
91 poidsob <- 1/abs(don[,1])^poids
92 poidsob <- poidsob/sum(poidsob)
93 epsilonp <- c(temp3)*(sqrt(rep(poidsob, nbr)))

```

```

94 var.param <- matInv * var(epsilonp)
95
96 ## Bootstrap
97 n <- length(epsilonp)
98 sigma <- sqrt(var(epsilonp))
99 distrCv <- rep(0,b)
100
101 for(j in 1:b){
102
103   B <- rnorm(n,0,sqrt(sigma))
104   C <- rmvnorm(1,mean=c(0,0,0),var.param)
105   drift <- (C[1] + C[2]*mean(don[,1]) + C[3]*mean(don[,1]^2)) # Drift function to used when
106     esimated residuals are involved
107   cv <- 0
108   for(i in 1:nbgrille){
109     cv <- cv + n/nbgrille * (mean(B <= qnorm(i/nbgrille,mean=0,sd=sqrt(sigma)))- i/
110       nbgrille + drift*dnorm(qnorm(i/nbgrille,0,sqrt(sigma)),0,sd=sqrt(sigma))/sqrt(n))^2
111   }
112   distrCv[j] <- cv
113 }
114
115 cv0 <- CVM(epsilonp,nbgrille,sigma)
116 pvalN <- pval(cv0,distrCv)
117
118 liste <- list(param=param.optimaux,varparam=var.param,pvalN=pvalN)
119 return(liste)
120 }

```