Appendix A

R Code

```
getwd()
setwd("C:/Users/Lara/Documents/Unif/M moire/Donn es/Dataset")
data2<-read.table("Normalized_data.txt", header = TRUE, fill=
    TRUE, dec=",")
attach(data2)

library(survival)
library(nloptr)
library(numDeriv)
library(MASS)
library(pander)
library(pbivnorm)
library(robustHD)
```

A.1 Common Functions

```
## LIST OF COMMON FUNCTIONS ##

slog = function(y) ## Log-transformation
{
  sy = (y>0)*y+(y<=0)*1 # sy>0
  return(log(sy))
}

YJtrans = function(y,theta) # Yeo-Johnson transformation
{
  sg = y>=0
  if (theta==0) {temp = slog(y+1)*sg+(1-sg)*(0.5-0.5*(y-1)^2)}
  if (theta==2) {temp = sg*(-0.5+0.5*(y+1)^2)-slog(-y+1)*(1-sg)
   }
  if ((theta!=0) & (theta!=2)) {temp =
    sg*(spower(y+1,theta)-1)/theta+(1-sg)*(1-spower(-y+1,2-
   theta))/(2-theta)}
  return(temp)
}

DYJtrans = function(y,theta) # Derivative of Yeo-Johnson
    transformation
{
  sg = y>=0
  temp = spower(y+1,theta-1)*sg+spower(-y+1,1-theta)*(1-sg)
  return(temp)
}

IYJtrans = function(y,theta) # Inverse of Yeo-Johnson
    transformation
{
  sg = y>=0
  if (theta==0) {temp =(exp(y)-1)*sg+(1-sg)*(1-spower(-2*y
   +1,0.5))}
  if (theta==2) {temp = sg*(-1+spower(2*y+1,0.5))+(1-exp(-y))*
   (1-sg)}
  if ((theta!=0) & (theta!=2)) {temp =
    sg*(spower(abs(theta)*y+1,1/theta)-1)+(1-sg)*(1-spower
   (1-(2-theta)*y,1/(2-theta)
    ))}
```

```
  return(temp)
}

##Likelihoods

LikI = function(par,Y,Delta,M){ #Independence model assumption
  k = ncol(M)
  beta = par[1:9] #demand
  eta = par[10:18] #supply
  sigma1 = par[k+1]
  sigma2 = par[k+2]
  alp1 = par[k+3]
  transY = YJtrans(Y,alp1)
  DtransY = DYJtrans(Y,alp1)
  z1 = (transY-(M[,1:9]%*%beta))/sigma1
  z2 = (transY-(M[,10:18]%*%eta))/sigma2
  tot =((1/sigma1)*dnorm(z1)*(1-pnorm(z2)))^Delta*((1/sigma2)*
   dnorm(z2)*(1-pnorm(z1)))^(1-Delta)*DtransY
  p1 = pmax(tot,1e-100)
  Logn = sum(log(p1));
  return(-Logn)
}



spower = function(y,pw)
{
  sy = (y>0)*y+(y<=0)*1 ## sy>0
  return(sy^pw)
}


LikF = function(par,Y,Delta,M){ #Joint model with dependent
    censoring
  k = ncol(M)
  beta = par[1:9] #demand
  eta = par[10:18] #supply
  sigma1 = par[k+1]
  sigma2 = par[k+2]
  rho = par[k+3]
  alp1 = par[k+4]
  transY = YJtrans(Y,alp1)
  DtransY = DYJtrans(Y,alp1)
```

```
  z1 = (transY-(M[,1:9]%*%beta))/sigma1
  z2 =((1-rho*sigma2/sigma1)*transY-(M[,10:18]%*%eta-rho*(
   sigma2/sigma1)*M[,1:9]%*%beta))/(sigma2*(1-rho^2)^0.5)
  z3 = (transY-(M[,10:18]%*%eta))/sigma2
  z4 =((1-rho*sigma1/sigma2)*transY-(M[,1:9]%*%beta-rho*(sigma1
   /sigma2)*M[,10:18]%*%eta))/(sigma1*(1-rho^2)^0.5)
  tot =((1/sigma1)*dnorm(z1)*(1-pnorm(z2)))^Delta*((1/sigma2)*
   dnorm(z3)*(1-pnorm(z4)))^(1-Delta)*DtransY
  p1 = pmax(tot,1e-100)
  Logn = sum(log(p1));
  return(-Logn)
}


lden_I = function(par,Y,Delta,M){ #To calculate the function at
    fixed parameter values for independence model
  k = ncol(M)
  beta = par[1:9] #demand
  eta = par[10:18] #supply
  sigma1 = par[k+1]
  sigma2 = par[k+2]
  alp1 = par[k+3]
  transY = YJtrans(Y,alp1)
  DtransY = DYJtrans(Y,alp1)
  z1 = (transY-(M[,1:9]%*%beta))/sigma1
  z2 = (transY-(M[,10:18]%*%eta))/sigma2
  tot =
    ((1/sigma1)*dnorm(z1)*(1-pnorm(z2)))^Delta*((1/sigma2)*
   dnorm(z2)*(1-pnorm(z1)))^
    (1-Delta)*DtransY
  return(sum(log(tot)))
}

lden_D = function(par,Y,Delta,M){ #To calculate the function at
    fixed parameter values for dependence model
  k = ncol(M)
  beta = par[1:9] #demand
  eta = par[10:18] #supply
  sigma1 = par[k+1]
  sigma2 = par[k+2]
  rho = par[k+3]
  alp1 = par[k+4]
  transY = YJtrans(Y,alp1)
```

```
  DtransY = DYJtrans(Y, alp1)

  z1 = (transY-(M[,1:9]%*%beta))/sigma1
  z2 =((1-rho*sigma2/sigma1)*transY-(M[,10:18]%*%eta-rho*(
   sigma2/sigma1)*M[,1:9]%*%beta))/(sigma2*(1-rho^2)^0.5)
  z3 = (transY-(M[,10:18]%*%eta))/sigma2
  z4 =((1-rho*sigma1/sigma2)*transY-(M[,1:9]%*%beta-rho*(sigma1
   /sigma2)*M[,10:18]%*%eta))/(sigma1*(1-rho^2)^0.5)
  tot =((1/sigma1)*dnorm(z1)*(1-pnorm(z2)))^Delta*((1/sigma2)*
   dnorm(z3)*(1-pnorm(z4)))^(1-Delta)*DtransY
  return(sum(log(tot)))
}
```

```
CondD = function(par,Y,M){ #Conditional Distibution of D given
   X for dependence model
  k = ncol(M)
  beta = par[1:9] #demand
  eta = par[10:18] #supply
  sigma1 = par[k+1]
  sigma2 = par[k+2]
  rho = par[k+3]
  alp1 = par[k+4]
  transY = YJtrans(Y, alp1)
  z1 = (transY-(M[,1:9]%*%beta))/sigma1
  Dist=pnorm(z1)
  return(Dist)
}

CondS = function(par,Y,M){ #Conditional Distibution of S given
   X for dependence model
  k = ncol(M)
  beta = par[1:9] #demand
  eta = par[10:18] #supply
  sigma1 = par[k+1]
  sigma2 = par[k+2]
  rho = par[k+3]
  alp1 = par[k+4]
  transY = YJtrans(Y, alp1)
  z1 = (transY-(M[,10:18]%*%eta))/sigma2
  Dist=pnorm(z1)
```

```
    return(Dist)
}
```

A.2 Independent Model

```
## ESTIMATION INDEPENDENT MODEL ##

TransS_1YJ = YJtrans(as.vector(slog(data2$TransS_1)),1.5)
data=cbind(data2,TransS_1YJ)

#Matrix of covariates
M=as.matrix(cbind(rep(1,nrow(data)),data$Infl_rate,data$DPrice,
    data$TransS_1YJ,data$NH,data$CC,data$IR,data$HI,data$HINC,
    rep(1,nrow(data)),data$HIT_1,data$Infl_rateT_1,data$DPriceT_
    1,data$TransS_1YJ,data$IRT_1,data$BankT_1,data$NC,data$DLCT_
    1)) # 16 covariates. #8 demand, 8 supply


dataD=data[data$Censor==0,] # Matrix for event times = demand<
    supply
dataS=data[data$Censor==1,] # Matrix for censored times =
    supply<demand
Ylog=as.vector(slog(data$Trans)) # Log transformation of "
    survival time" (=number of transaction)
Delta=as.vector(data$Event) # Event indicator = demand<supply
    (1 in the colomn)


#Surv_D = lm(Trans~Infl_rate+DPrice+TransS_1+NH+CC+IR+HI+HINC,
    data=dataD) #Parameters initiation
#Surv_S = lm(Trans~HIT_1+Infl_rateT_1+DPriceT_1+TransS_1+IRT_1+
    BankT_1+NC+DLCT_1,data=dataS)
surv_D=survreg(Surv(Trans,Event)~Infl_rate+DPrice+TransS_1YJ+NH
    +CC+IR+HI+HINC,data=data) #Parameters initiation
surv_S=survreg(Surv(Trans,Censor)~HIT_1+Infl_rateT_1+DPriceT_1+
    TransS_1YJ+IRT_1+BankT_1+NC+DLCT_1,data=data)

init_beta=surv_D$coefficients #initial values
init_eta=surv_S$coefficients #initial values

#summary(Surv_D)
#summary(Surv_S)

#init_sigma1=c(26.47) #standard deviation
#init_sigma2=c(25.08) #standard devation
```

```
init_sigma1=surv_D$scale #standard deviation
init_sigma2=surv_S$scale #standard devation
init_alp1=c(0.5,1,1.5)

estimates_ind=matrix(ncol=length(init_alp1),nrow=23) #
    Sensitivity
for (i in 1:length(init_alp1)){
  par=c(init_beta,init_eta,init_sigma1,init_sigma2,init_alp1[i
    ])
  # Likelihood maximisation
  res=nloptr(x0=par,eval_f=LikI,M=M,Y=Ylog,Delta=Delta,lb=c(-
    Inf,-Inf,-Inf,-Inf,-Inf,-Inf,-Inf,-Inf,-Inf,-Inf,-Inf,-
    Inf,-Inf,-Inf,-Inf,-Inf,1e-05,1e-5,0),ub=c(Inf,Inf,Inf,
    Inf,Inf,Inf,Inf,Inf,Inf,Inf,Inf,Inf,Inf,Inf,Inf,Inf,Inf,
    Inf,Inf,2),eval_g_ineq=NULL,opts = list(algorithm ="NLOPT_LN
    _BOBYQA","ftol_abs"=1.0e-30,"maxeval"=100000,"xtol_abs"=rep
    (1.0e-30)))
  LiK_ind=lden_I(res$solution,Ylog,Delta,M)
  estimates_ind[,i]=c(init_alp1[i],res$solution,LiK_ind)
  rownames(estimates_ind)=c("init_alp1","beta_int","beta_Infl_
    rate","beta_DPrice","beta_TransQ-1",
                            "beta_NH","beta_CC","beta_IR","
    beta_HI","beta_HINC","eta_int",
                            "eta_HIT-1","eta_Infl_rateT-1","
    eta_DPriceT-1",
                            "eta_TransQ-1","eta_IRT-1","eta_
    BankT-1",
                            "eta_NC","eta_DLCT-1",
                            "sigma1","sigma2",
                            "alp1","logLik")
   }
pander(estimates_ind,caption="Estimates under independence
    model for different initial theta values")
H_ind = hessian(LikI,estimates_ind[c(2:22),1],Y=Ylog,Delta=
    Delta,M=M,method="Richardson",method.args=list(eps=1e-4, d
    =0.01, zer.tol=sqrt(.Machine$double.eps/7e-7), r=6,v=2, show
    .details=FALSE))
H_ind=ginv(H_ind, tol = sqrt(.Machine$double.eps)) #Variance -
    Covariance matrix
se_ind=sqrt(diag(H_ind))
W_ind=as.vector(estimates_ind[c(2:22),1])^2/(se_ind)^2
P_val_ind=round(pchisq(W_ind,df=1,lower.tail=FALSE),5) # Wald
    test P-value
```

```
p_val_ind_res=cbind(estimates_ind[c(2:22),1],se_ind,P_val_ind)
colnames(p_val_ind_res)=c("Estimate","Standard Error","P-value"
    )
pander(p_val_ind_res,caption="Wald test")
```

A.3 Dependent Model

A.3.1 Wald Test

```
## ESTIMATION DEPENDENT MODEL ##

#Trans_Q (number of transactions during the semester) is my Y
    variable. Also included as covariate in t-1 (time series:
    autoregressive model). Therefore, I YJ-transformed it also.
TransS_1YJ = YJtrans(as.vector(slog(data2$TransS_1)),1.5)
data=cbind(data2,TransS_1YJ)

#Matrix of covariates
M=as.matrix(cbind(rep(1,nrow(data)),data$Infl_rate,data$DPrice,
    data$TransS_1YJ,data$NH,data$CC,data$IR,data$HI,data$HINC,
    rep(1,nrow(data)),data$HIT_1,data$Infl_rateT_1,data$DPriceT_
    1,data$TransS_1YJ,data$IRT_1,data$BankT_1,data$NC,data$DLCT_
    1)) # 16 covariates. #8 demand, 8 supply


dataD=data[data$Censor==0,] # Matrix for event times = demand<
    supply
dataS=data[data$Censor==1,] # Matrix for censored times =
    supply<demand
Ylog=as.vector(slog(data$Trans)) # Log transformation of "
    survival time" (=number of transaction)
Delta=as.vector(data$Event) # Event indicator = demand<supply
    (1 in the colomn)


#Surv_D = lm(Trans~Infl_rate+DPrice+TransS_1+NH+CC+IR+HI+HINC,
    data=dataD) #Parameters initiation
#Surv_S = lm(Trans~HIT_1+Infl_rateT_1+DPriceT_1+TransS_1+IRT_1+
    BankT_1+NC+DLCT_1,data=dataS)
surv_D=survreg(Surv(Trans,Event)~Infl_rate+DPrice+TransS_1YJ+NH
    +CC+IR+HI+HINC,data=data) #Parameters initiation
surv_S=survreg(Surv(Trans,Censor)~HIT_1+Infl_rateT_1+DPriceT_1+
    TransS_1YJ+IRT_1+BankT_1+NC+DLCT_1,data=data)

init_beta=surv_D$coefficients #initial values
init_eta=surv_S$coefficients #initial values

#summary(Surv_D)
#summary(Surv_S)
```

```
#init_sigma1=c(26.47) #standard deviation
#init_sigma2=c(25.08) #standard devation
init_sigma1=surv_D$scale #standard deviation
init_sigma2=surv_S$scale #standard devation


alp1_values=c(0.5,1,1.5) #Sensitivity (stability of the
    parameters) #the theta of the YJ: must be between 0 and 2.
rho_values=c(-0.9,-0.5,0,0.5,0.9) #correlation between D and S
estimates=matrix(ncol=length(rho_values),nrow=23) #23 rows for
    the 23 parameters (see "rownames" below)
colnames(estimates)=rho_values
rownames(estimates)=c("beta_int","beta_Infl_rate","beta_DPrice"
    ,"beta_TransQ-1",
                          "beta_NH","beta_CC","beta_IR","beta_HI","
    beta_HINC","eta_int",
                          "eta_HIT-1","eta_Infl_rateT-1","eta_
    DPriceT-1",
                          "eta_TransQ-1","eta_IRT-1","eta_BankT-1",
                          "eta_NC","eta_DLCT-1",
                          "sigma1","sigma2","rho",
                          "alp1","logLik")
estimates_dep=list(alp1=matrix(ncol=length(rho_values),nrow=23)
    ,alp2=matrix(ncol=length(rho_values),nrow=23),alp3=matrix(
    ncol=length(rho_values),nrow=23))
    #estimates_Dep = estimates of the params of the dependent
    model

lb = c(rep(-Inf,dim(M)[2]),1e-05,1e-5,-1,0);
ub = c(rep(Inf,dim(M)[2]+2),1,2)




# nloptr algorithm

#we test for each value of alp1 and rho
for (j in 1:length(alp1_values)){
  for (i in 1:length(rho_values)){
    init_alp1=alp1_values[j]
    init_rho=rho_values[i]
    par = c(init_beta,init_eta,init_sigma1,init_sigma2,init_rho
```

```
    ,init_alp1)

     #nloptr = to optimize with the Likelihood Fctn

     res = nloptr(x0 = par,eval_f = LikF,M=M,Y=Ylog,Delta=Delta,
     lb = lb, ub = ub,eval_g_ineq=NULL,opts = list(algorithm ="
    NLOPT_LN_BOBYQA","ftol_abs"=1.0e-30,"maxeval"=100000,"xtol_
    abs"=rep(1.0e-30)))

     Lik_dep=lden_D(res$solution,Ylog,Delta,M)
     estimates[,i]=c(res$solution,Lik_dep)
     #estimates: list created above: results for each value of i
     #print(Lik_dep)
   }

   estimates_dep[[j]]=estimates
  #estimates_dep: list created above: results for each value of
    i and J
}


estimates_dep
par_est=estimates_dep[[3]][c(1:22),2]

H_dep=hessian(LikF,par_est,Y=Ylog,Delta=Delta,M=M,method="
    Richardson",method.args=list(eps=1e-4, d=0.01, zer.tol=sqrt
    (.Machine$double.eps/7e-7), r=6, v=2,show.details=FALSE))
#variance-covar matrix
H_dep= ginv(H_dep, tol = sqrt(.Machine$double.eps)) #inverse (
    Fisher inf matrix)
se_dep=sqrt(diag(H_dep))
W_dep=as.vector(par_est)^2/(se_dep)^2
P_val_dep=round(pchisq(W_dep,df=1,lower.tail=FALSE),5)
p_val_dep_res=cbind(par_est,se_dep,P_val_dep)
colnames(p_val_dep_res)=c("Estimate","Standard Error","P-value"
    )
pander(p_val_dep_res,caption="Wald test")



#  nlminb algorithm
```

```
#we test for each value of alp1 and rho

for (j in 1:length(alp1_values)){
  for (i in 1:length(rho_values)){
    init_alp1=alp1_values[j]
    init_rho=rho_values[i]
    par = c(init_beta,init_eta,init_sigma1,init_sigma2,init_rho
    ,init_alp1)

    #nloptr = to optimize with the Likelihood Fctn

    res = nlminb(start = par,LikF,M = M,Y = Ylog,Delta = Delta,
    lower = lb ,upper = ub, control = list(eval.max=300,iter.max
    =200))

    Lik_dep=lden_D(res$par,Ylog,Delta,M)
    estimates[,i]=c(res$par,Lik_dep)
    #estimates: list created above: results for each value of i
    #print(Lik_dep)
  }

  estimates_dep[[j]]=estimates
  #estimates_dep: list created above: results for each value of
    i and J
}
#

par_est=estimates_dep[[3]][c(1:22),2]
H_dep=hessian(LikF,par_est,Y=Ylog,Delta=Delta,M=M,method="
    Richardson",method.args=list(eps=1e-4, d=0.01, zer.tol=sqrt
    (.Machine$double.eps/7e-7), r=6, v=2,show.details=FALSE))
#variance-covar matrix
H_dep= ginv(H_dep, tol = sqrt(.Machine$double.eps)) #inverse (
    Fisher inf matrix)
se_dep=sqrt(diag(H_dep))
W_dep=as.vector(par_est)^2/(se_dep)^2
P_val_dep=round(pchisq(W_dep,df=1,lower.tail=FALSE),5)
p_val_dep_res=cbind(par_est,se_dep,P_val_dep)
colnames(p_val_dep_res)=c("Estimate","Standard Error","P-value"
    )
pander(p_val_dep_res,caption="Wald test")
```

A.3.2 Survival Curves

```
## SURVIVAL CURVES ##

Ylog_data=as.vector(slog(data$Trans))
M_data=as.matrix(cbind(rep(1,nrow(data)),data$Infl_rate,data$
    DPrice,data$TransS_1YJ,data$NH,data$CC,data$IR,data$HI,data$
    HINC,rep(1,nrow(data)),data$HIT_1,data$Infl_rateT_1,data$
    DPriceT_1,data$TransS_1YJ,data$IRT_1,data$BankT_1,data$NC,
    data$DLCT_1)) # 16 covariates. #8 demand, 8 supply

D=CondD(par_est,Ylog,M_data)
P=CondS(par_est,Ylog,M_data)
plot(sort(1-D[data$Event==1],decreasing=T)~sort(exp(Ylog_data[
    data$Event==1]),decreasing = T), type="l",xlab="Observed
    Quantity",ylab="Frequency",ylim=c(0,1),xlim=c(1000,16))
par(new=T)
plot(sort(1-P[data$Event==0],decreasing=T)~sort(exp(Ylog_data[
    data$Event==0]),decreasing = T), type="l",xlab="Observed
    Quantity",ylab="Frequency",col="red",ylim=c(0,1),xlim=c
    (1000,16))
```

A.3.3 PP Plot

```
## PP PLOT (= Goodness of fit plot) ##

par1 = par_est[1:9] #variables Demand
s1 = par_est[19] #sigma1
s2 = par_est[20] #sigma2
rho1 = par_est[21] #rho
par2 = par_est[10:18] #variables Supply
Ylog=as.vector(slog(data$Trans)) #log survival transactions
z = YJtrans(Ylog,par_est[22]) #12 = alp1
hist(z)
#loop, calculates the distribution of transformed minimum under
    assumed model.
n = length(z)
Ocdf = rep(0,n)
M_D=as.matrix(cbind(rep(1,nrow(dataD)),dataD$Infl_rate,dataD$
    DPrice,dataD$TransS_1,dataD$NH,dataD$CC,dataD$IR,dataD$HI,
```

```
    dataD$HINC))
M_S=as.matrix(cbind(rep(1,nrow(dataS)),dataS$HIT_1,dataS$Infl_
    rateT_1,dataS$DPriceT_1,dataS$TransS_1,dataS$IRT_1,dataS$
    BankT_1,dataS$NC,dataS$DLCT_1))
for (i in 1:n){
  ob = z[i]
  T1 = (ob-M[,1:9]%*%par1)/s1
  T2 = (ob-M[,10:18]%*%par2)/s2
  X = cbind(T1,T2)
  Ocdf[i] = sum(pnorm(T1)+pnorm(T2)-pbivnorm(X,rho=rho1))/n
}
ord = rank(z,ties.method = "first")
Ecf = (ord-1)/n
dd = cbind(Ocdf,Ecf,z)
#PPplot
sort_dd=dd[order(dd[,3]),]
plot(sort_dd[,1],sort_dd[,2], xlab="Assumed Cumulative
    distribution function",ylab="Empirical Cumulative
    distribution function", main="PP Plot")
abline(a=0,b=1,col="red")
```

A.3.4 Goodness of Fit Test

```
## GOODNESS-OF-FIT TEST ##

#Observed Cramer-von-Mise
Ocf = Ocdf[order(Ocdf)]
Ecf=Ecf[order(Ecf)]
w = rep(0,n)
w[1] = Ocf[1]
w[2:n] = Ocf[2:n]-Ocf[1:(n-1)]
OCVM = sum((Ecf-Ocf)^2*w)*n
#GoF
GofTest = function(n,nsim,par) {
  CVM = rep(0,nsim)
  CVMb.star = rep(0,nsim)
  count1 = 0
  count2 = 0
  for (j in 1:nsim){
    M_0 =as.matrix(cbind(slog(data$Trans),data$Event,rep(1,nrow
    (data)),data$Infl_rate,data$DPrice,data$TransS_1,data$NH,
```

```
    data$CC,data$IR,data$HI,data$HINC,rep(1,nrow(data)),data$HIT
    _1,data$Infl_rateT_1,data$DPriceT_1,data$TransS_1,data$IRT_
    1,data$BankT_1,data$NC,data$DLCT_1))
     sample=sample((nrow(M_0)),n,replace=T) #bootstrap
    echantillon with replacement
     datab = M_0[sample,] # data matrix
     Y = datab[,1]
     Delta = datab[,2]
     M = datab[,3:20]
     W = M
     k = ncol(M)
     u1 = k+1;u2 = k+3

     # dependent Censoring model
     lb = c(rep(-Inf,k),1e-02,1e-02,-1,0)
     ub = c(rep(Inf,(k+2)),1,2)
     init = par
parhat = nloptr(x0=init,eval_f=LikF,Y=Y,Delta=Delta,M=M,lb=lb,
    ub=ub,eval_g_ineq=NULL,opts = list(algorithm = "NLOPT_LN_
    BOBYQA","ftol_abs"=1.0e-30,"maxeval"=100000,"xtol_abs"=rep
    (1.0e-30)))$solution
Z0 = YJtrans(Y,parhat[(k+4)])
# Distribution under model
OCd = rep(0,n)
for (i in 1:n){
  ob = Z0[i]
  T1 = (ob-M[,1:9]%*%parhat[1:9])/parhat[(k+1)]
  T2 = (ob-W[,10:18]%*% parhat[10:k])/parhat[(k+2)]
  X = cbind(T1,T2)
  OCd[i] = sum(pnorm(T1)+pnorm(T2)-pbivnorm(X,rho= parhat[(k+3)
    ]))/n
  #distribution under fitted model
}
ord = 1:n
Ecf = (ord-1)/n # Emprical distribution
# Compute test distribution
OCf = OCd[order(OCd)]
w = rep(0,n)
w[1] = OCf[1]
w[2:n] = OCf[2:n]-OCf[1:(n-1)]
CVM[j] = sum((Ecf-OCf)^2*w)*n # Cramer-Von-Mises
count1 = quantile(CVM,prob=0.9)
count2 = quantile(CVM,prob=0.95)
```

```
    }
    return(list(count=c(count1,count2),CVM))
}
n = 4422
iseed=212367
nsim=200
GOF=GofTest(n,nsim,par_est)
p_val=1-(sum(GOF[[2]]<OCVM)/nsim)
names(GOF$count)=c("Q90","Q95")
names(p_val)="P_value"
names(OCVM)="Observed Cramer-von-Mise"
pander(c(GOF$count,OCVM,p_val))
```