```
##########################################
#
#  Reinhard Hoepfner
#
#  Praktikum zur 'Einfuehrung in die Stochastik'
#
#  Euler-Schemata fuer SDE
#
#  Ornstein-Uhlenbeck Prozess
#  Transienz und Rekurrenz
#
#  21.01.19
#
##########################################




##########################################
#
#  0) wiederholungen
#
##########################################


beobachtungsdauer <- 1 ;
  # einen wert 'T' fuer ein zeitintervall '[0,T]' vorgeben,
  # auf dem Pfade simuliert werden sollen
delta <- 10^(-4) ;
  # eine schrittweite fuer euler-schema
  # bzw fuer unabhaengige zuwaechse vorgeben
start <- 0 ;
  # einen startwert fuer den brownschen pfad vorgeben


  # wdh 1-dim brownschen bewegung
  # nutze unabhaengigkeit der zuwaechse
M <- ceiling( beobachtungsdauer / delta ) ;
xx <- delta * ( 0 : M ) ;
zuwaechse <- sqrt(delta) * rnorm( M, 0, 1 )  ;
yy <- cumsum( c( start, zuwaechse ) ) ;


  # wir hatten wegen stetigkeit der pfade linear interpoliert :
plot( xx, yy, type="l", xlab="zeit", ylab="ort", col=2 ) ;

  # genauer aber sieht das eher so aus :
plot( xx, yy, pch=19, cex=0.1, xlab="zeit", ylab="ort", col=2 ) ;
  # wobei man sich 'unendlich zittrige' zwischenstuecke
  # zwischen die gezeichneten punkte denken muss ...



  # alternativ dasselbe
  # mit einem euler-schema programmiert :
yy <- rep( start, length(xx) ) ;
for (i in 2:length(xx) )
  yy[i] <- yy[i-1] + sqrt(delta)*rnorm(1,0,1)  ;
plot( xx, yy, pch=19, cex=0.1, xlab="zeit", ylab="ort", col=2 ) ;


 # ganz drastisch im zoom auf den abfangsabschnitt
plot( xx[xx<0.05], yy[xx<0.05], pch=19, cex=0.4, xlab="zeit", ylab="ort", col=2 ) ;
 # erinnerung : brownsche pfade
 # haben eine quadratische variation !!
```

```
 # dies waren 'exakte simulationen'
 # dank der eigenschaft der unabhaengigkeit der zuwaechse
 # in der brownschen bewegung

 # die folgenden euler-schemata jedoch
 # liefern i.a. nur (gute) approximationen
 # an die loesungen stochastischer differentialgleichungen



##############################
#
#  1) allgemein: euler schema fuer SDE :
#
#  setze voraus: koeffizienten erfuellen
#  eine globale lipschitzbedingung,
#  oder eine lokale lipschitzbedingung zusammen mit einer
#  einer linearen wachstumsbedingung
#
##################################


 # definiere eine (lipschitz-) driftfunktion, z.b.
drift <- function(x){ 1 - 0.5*x } ;
x_gitter <- seq( -10,10,length.out=1000) ;
plot( x_gitter, drift(x_gitter), type="l" ) ;

 # definiere eine (lipschitz-) volatilitaetsfunktion, z.b.
volat <- function(x){ sqrt( 1 + x^2 ) } ;
plot( x_gitter, volat(x_gitter), type="l" ) ;


 # ein euler schema zur simulation der pfade der SDE
 # zu dieser drift und dieser volatilitaetsfunktion
 # sieht so aus :
euler <- function( beobachtungsdauer, delta, start ){
 M <- ceiling( beobachtungsdauer / delta ) ;
 pfad <- rep( start, M ) ;
 alt <- start ;
 for ( i in 2:(M+1) ) {
    alt <- pfad[i-1] ;
    # sichere zustand des prozesses im vorherigen schritt
    pfad[i] <- alt + drift(alt)*delta + volat(alt)*sqrt(delta)*rnorm(1,0,1) ;
    } ; # ende i schleife
 pfad ; # ausgabe
 } ; # ende euler schema


 # test :
delta <- 10^(-4) ;
start <- 2.5 ;

beobachtungsdauer <- 1.5 ;
M <- ceiling( beobachtungsdauer / delta ) ;
xx <- delta * ( 0 : M ) ;
yy <- euler( beobachtungsdauer, delta, start ) ;


plot( xx, yy, type="l", xlab="zeit", ylab="ort", col=2 ) ;
 # weil man weiss, dass die loesung einer
 # durch die brownsche bewegung getriebenen SDE mit lipschitz-koeffizienten wie oben
 # ein stochastischer prozess mit stetigen pfaden ist
```

```
 # jetzt viele trajektorien anschauen
beobachtungsdauer <- 15 ;
M <- ceiling( beobachtungsdauer / delta ) ;
xx <- delta * ( 0 : M ) ;
yy <- euler( beobachtungsdauer, delta, start ) ;
plot( xx, yy, type="l", xlab="zeit", ylab="ort", col=2 ) ;



##########################################
#
#   2) ornstein uhlenbeck prozesse : loese die OU-SDE
#
#   dX_t  =  theta  X_t dt  +  dW_t
#
#   mit reellem parameter  theta
#   (im spezialfall  theta=0  ist das die brownsche bewegung selbst)
#
########################################


 # fuehre ein kurzschreibweisen fuer
 # 'bd' beobachtungsdauer
 # 'sw' schrittweite delta
 # 'st' startwert
 # 'th' fuer ornstein-uhlenbeck-parameter theta in SDE

 # loese die SDE
 # dX_t = theta X_t dt + dW_t


euler_OU <-  function( bd, sw, st, th ) {
            M <- ceiling( bd / sw )  ;
            z <- rep( st , M ) ; # startwert zum ueberschreiben
            zalt <- st ;
            sqsw <- sqrt(sw) ;
            for(i in 2:(M+1)){
               zalt <- z[i-1] ;
               z[i] <- zalt + th*zalt*sw + sqsw*rnorm(1,0,1) ;
               } ; # ende i schleife
            z ;
            } ; # euler schema fuer pfad eines OU prozesses
            # mit driftparameter 'th' und startwert 'st'
            # (euler-schema mit schrittweite 'sw'
            # ueber ein zeitintervall von 0 bis 'bd')


st <- 0 ;
# st <- 14.7 ;
# st <- 131.4 ;
 # startwert  'st'  der simulation
 # spiele mit verschiedenen fallkombinationen
sw <- 10^(-4) ;

bd <- 1 ;
# bd <- 10 ;
# bd <- 143.9 ;
  # waehle  'bd'  beobachtungsdauer



 #  variiere OU parameter  theta  der simulation
 #  'th' < 0 : drift wirkt zentripetal : zurueck zur null
 #  'th' > 0 : drift wirkt zentrifugal

th <- - 0.1 ;     # schwach zentripetal
# th <- - 0.5 ;   # zentripetal
```

```
# th <- - 1.5 ;   # stark zentripetal

# th <- 0.1 ;   # schwach zentrifugal
# th <- 0.5 ;   # zentrifugal
# th <- 1.5 ;   # stark zentrifugal


  # jetzt mehrmals eine simulation rechnen
leg <- "OU pfad mit parameter theta = !!!! ueber das zeitintervall [0,%%%%]" ;
leg <- gsub( "!!!!", th, leg ) ;
leg <- gsub( "%%%%", bd, leg ) ; leg ;
M <- ceiling( bd / sw ) ;
xx <- sw * 0:M  ; length(xx) ;
yy <- euler_OU( bd, sw, st, th ) ;

plot( xx, yy, type="l", xlab="zeit", ylab="ort", col=2, main=leg ) ;


 # gegebenenfalls zoom auf anfangs- und endabschnitt
par( mfrow=c(1,2) ) ;
leg1 <- "OU mit theta = !!!! : zoom auf anfangsstueck" ;
leg1 <- gsub( "!!!!", th, leg1 ) ;
leg1 <- gsub( "%%%%", bd, leg1 ) ; leg1 ;
plot( xx[ xx < 0.005*bd ], yy[ xx < 0.005*bd ], pch=19, cex=0.4, xlab="zeit", ylab="ort", col=2,
main=leg1 ) ;
#
leg2 <- "pfad bis zur zeit %%%% : zoom auf endstueck" ;
leg2 <- gsub( "!!!!", th, leg2 ) ;
leg2 <- gsub( "%%%%", bd, leg2 ) ; leg2 ;
plot( xx[ xx > 0.995*bd ], yy[ xx > 0.995*bd ], pch=19, cex=0.4, xlab="zeit", ylab="ort", col=2,
main=leg2 ) ;
par( mfrow=c(1,1) ) ;


  # hier die verschiedenen fallkombinationen durchprobieren,
  # betrachte immer mehrere laeufe pro fall !!



####################################
#
#   rekurrenter fall theta < 0 :
#
#   zeichnen von mehreren OU pfaden
#   mit verschiedenen startwerten :
#
####################################


th <- - 1.5 ;
# th <- -0.5 ;
# th <- -0.1 ;
  # waehle staerke der backdriving force
bd <- 15 ;
  # lange beobachtungsdauer vorgeben
N <- 25 ;
  # anzahl der zu simulierenden OU pfade
sw <- 10^(-3) ;
  # schrittweite fuer euler-schema


legmain <- "%%%% simulierte OU pfade mit parameter theta =  (rekurrenter fall)" ;
legmain <- gsub( "", th, legmain ) ;
legmain <- gsub( "%%%%", N, legmain ) ;

M <- ceiling( bd / sw ) ;
  # schrittzahl fuer euler-schema
```

```
start_scale <- 20 ;
  # als skalierung fuer zufaellig zu waehlende startwerte
startwerte <- runif( N, -start_scale, start_scale) ;
xx <- sw * 0:M ;
length(xx) ; # das gitter von zeitpunkten



 # jetzt zu den simulierten startwerten
 # je einen OU pfad berechnen
feld <- matrix( 0, nrow = M+1, ncol=N ) ;
for( i in 1:N) feld[,i] <- euler_OU( bd, sw, startwerte[i], th ) ;


 # einen festen bilderrahmen vorgeben (rekurrenter fall)
c1 <- -1.1*start_scale ;
c2 <-  1.1*start_scale ;
plot( c( 0, bd ), c( c1, c2 ), type="n", xlab="zeit", ylab="ort", main=legmain) ;
  # liefert einen rechteckigen rahmen mit koordinaten
  # und legenden an der x-achse bzw der y-achse fuer ein zu zeichnendes bild
abline(h=0, lty=2) ;
i <- 1 ;

  # jetzt sukzessive in i
lines( xx, feld[,i], col=i ) ;
i <- i+1 ;

  # vollstaendiges bild
plot( c( 0, bd ), c( c1, c2 ), type="n", xlab="zeit", ylab="ort", main=legmain) ;
abline(h=0, lty=2) ;
for (i in 1:N ) lines( xx, feld[,i], col=i ) ;




#####################################################
#
#   transienter Fall  theta > 0  :
#
#   zeichnen von mehreren o-u pfaden mit startwert = 0
#
###########################################


th <- 1.5 ;
# th <- 0.5 ;
# th <- 0.1 ;
  # waehle staerke der zentrifugalen kraft
bd <- 15 ;
  # beobachtungsdauer vorgeben
N <- 25 ;
  # anzahl der simulationslaeufe
sw <- 10^(-3) ;
  # schrittweite fuer euler-schema


M <- ceiling( bd / sw ) ;
  # schrittzahl fuer euler-schema
startwerte <- rep( 0, N ) ;
  # alle startwerte fuer die zu simulierenden OU pfade auf null setzen

legmain <- "%%%% simulierte OU pfade mit parameter theta =  (transienter fall)" ;
legmain <- gsub( "", th, legmain ) ;
legmain <- gsub( "%%%%", N, legmain ) ;

M <- ceiling( bd / sw ) ;
  # schrittzahl fuer euler-schema
```

```
xx <- sw * 0:M ;
length(xx) ; # das gitter von zeitpunkten
feld <- matrix( 0, nrow = M+1, ncol=N ) ;
for( i in 1:N) feld[,i] <- euler_OU( bd, sw, startwerte[i], th ) ;



 # einen festen bilderrahmen (transienter fall) fuer alle pfade vorgeben
c3 <- 3 * exp( th * bd ) ;
plot( c( 0, bd ), c( -c3, c3 ), type="n", xlab="zeit", ylab="ort", main=legmain) ;
  # liefert einen rechteckigen rahmen mit koordinaten
  # und legenden an der x-achse bzw der y-achse fuer ein zu zeichnendes bild
abline(h=0, lty=2) ;
for (i in 1:N ) lines( xx, feld[,i], col=i ) ;


  # dieselben pfade : zoom ins anfangsstueck
  # waehle prozentsatz der gesamtbeaobachtungsdauer
prozent <- 0.05 ;
# prozent <- 0.1 ;
# prozent <- 0.25 ;
c4 <- 3 * exp( th * prozent * bd ) ;
plot( c( 0, prozent*bd ), c( -c4, c4 ), type="n", xlab="zeit", ylab="ort", main=legmain) ;
abline(h=0, lty=2) ;
for (i in 1:N ) lines( xx, feld[,i], col=i ) ;




################################################
#
#   exponentielle wachstumsgeschwindigkeit
#   im transienten fall (theta > 0) sichtbar machen
#
#   zeichne log-plots (aber vorsicht mit dem vorzeichen des OU pfades
#   und mit besuchen des OU pfades in der naehe der null)
#
################################################


  # uebernehme situation von oben: bd, sw, th, startwerte, feld ....
c3 <- 3 * exp( th * bd ) ;
plot( c( 0, bd ), c( -c3, c3 ), type="n", xlab="zeit", ylab="ort", main=legmain) ;
abline(h=0, lty=2) ;
for (i in 1:N ) lines( xx, feld[,i], col=i ) ;



  # und logarithmiere in geeigneter weise :
legmain <- "%%%% simulierte ornstein-uhlenbeck pfade mit parameter theta =  (transienter fall)
\nlogarithmierter absolutbetrag der pfade sofern diese ausserhalb einer 1-kugel um 0 sind" ;
legmain <- gsub( "", th, legmain ) ;
legmain <- gsub( "%%%%", N, legmain ) ;

logfeld <- ifelse( abs(feld)<1 , 0, log(abs(feld)) ) ;
  # vorsichtsmassnahme bei den besuchen des pfades in der naehe der null
  # pfade, die gegen + infty streben, werden logarithmiert
  # pfade, die gegen - infty streben, werden
  # im vorzeichen umgedreht und logarithmiert

c6 <- max ( logfeld[M,] ) ; c6 ;
  # den absolutbetrag der letzten erreichten zustaende als masstab nehmen


  # startphase anschauen
prozent <- 0.1 ;
plot( c( 0, prozent*bd ), c(0,prozent*c6), type="n", xlab="zeit", ylab="log(ort)",
```

```
main=legmain) ;
abline(h=0, lty=2) ;
for( i in 1:N) lines( xx, logfeld[,i], col=i ) ;

  # ueber gesamte beobachtungsdauer anschauen
plot( c( 0, bd ), c(0,c6), type="n", xlab="zeit", ylab="log(ort)", main=legmain) ;
abline(h=0, lty=2) ;
for( i in 1:N) lines( xx, logfeld[,i], col=i ) ;



 # rueckschluss auf den wahren parameter  theta > 0  aus der beobachteten steigung
M2 <- M ;
M1 <- round(3*M/4) ;
for ( i in 1:N ) {
  hilf <- logfeld[M2,i] - logfeld[M1,i] ;
  cat( hilf / (sw*(M2-M1)) , "\n" ) ;
  } ;
cat( "als ad-hoc schaetzung fuer den wahren parameter theta = ", th, "\n" ) ;




 # statistik stochastischer prozesse:
 # die theorie kennt einen 'optimalen' schaetzer
 # fuer das modell 'OU mit unbekanntem parameter  theta'
 # (hier: der maximum likelihood schaetzer)
for ( i in 1:N ) {
  yy <- feld[,i] ;
  schaetzer <- ( sum( yy[-length(yy)] * ( yy[-1] - yy[-length(yy)] ) ) ) / ( sw * sum( yy[-
length(yy)]^2 ) ) ;
  cat( schaetzer , "\n" ) ;
  } ; # berechnet aus jedem pfad ML schaetzwerte fuer theta
cat( "ML schaetzung fuer den wahren parameter theta = ", th, "\n" ) ;



############################  21.01.19  ######################
```