Stochastic Differential Equation Package

Deniz Kenan Kılıç

Dec 24, 2015

Outline

Outline

Outline

Introduction

- Sources, binaries and documentation for R can be obtained via CRAN, the "Comprehensive R Archive Network" at https://CRAN.R-project.org/mirrors.html.
- Then RStudio can be installed as an editor.
- R is an interpreted language, not a compiled one, i.e. all commands are directly executed without requiring to build a complete program.
- When R is running, variables, data, functions, results, etc, are stored in the active memory of the computer in the form of objects which have a name.
- All the actions of R are done on objects stored in the active memory of the computer: no temporary files are used.

Brownian motion, Brownian bridge, and geometric Brownian motion simulators

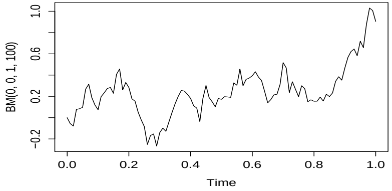
- BBridge(x=0, y=0, t0=0, T=1, N=100)
  BM(x=0, t0=0, T=1, N=100)
  GBM(x=1, r=0, sigma=1, T=1, N=100)
- **x**-initial value of the process at time t0.
  y-terminal value of the process at time T.
  t0-initial time.
  r-the interest rate of the GBM.
  sigma-the volatility of the GBM.
  T-final time.
  N-number of intervals in which to split [t0,T].

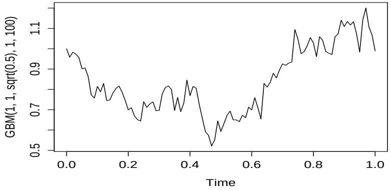
BM Algorithms

```
install.packages("sde")
library(sde)

set.seed(123)
par(mfrow=c(2,2))
plot (BM(0,0,1,100))
plot (GBM(1,1,sqrt(0.5),1,100))
plot (BBridge(0,-1,0,1,100))
```

BM Algorithms





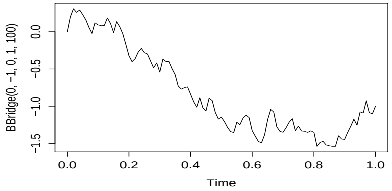


Figure: Brownian motion, Brownian bridge, and geometric Brownian motion simulators.

Simulation of stochastic differential equation

- sde.sim(t0 = 0, T = 1, X0 = 1, N = 100, delta, drift, sigma, drift.x, sigma.x, drift.xx, sigma.xx, drift.t, method = c("euler", "milstein", "KPS", "milstein2", "cdist","ozaki","shoji","EA"), alpha = 0.5, eta = 0.5, pred.corr = T, rcdist = NULL, theta = NULL, model = c("CIR", "VAS", "OU", "BS"), k1, k2, phi, max.psi = 1000, rh, A, M=1)
- **t0**-time origin. **T**-horizon of simulation. **X0**-initial value of the process.
  N-number of simulation steps.
  M-number of trajectories.
  delta-time step of the simulation.
  drift-drift coefficient: an expression of two variables t and x.
  sigma-diffusion coefficient: an expression of two variables t and x.
  drift.x-partial derivative of the drift coefficient w.r.t. x: a function of two variables t and x.
  sigma.x-partial derivative of the diffusion coefficient w.r.t. x: a function of two variables t and x.

Simulation of stochastic differential equation

drift.xx-second partial derivative of the drift coefficient w.r.t. x: a function of two variables t and x.
sigma.xx-second partial derivative of the diffusion coefficient w.r.t. x: a function of two variables t and x.
drift.t-partial derivative of the drift coefficient w.r.t. t: a function of two variables t and x.
method-method of simulation; see details.
alpha-weight alpha of the predictor-corrector scheme.
eta-weight eta of the predictor-corrector scheme.
pred.corr-boolean: whether to apply the predictor-correct adjustment; see details.
rcdist-a function that is a random number generator from the conditional distribution of the process; see details. theta-vector of parameters for cdist; see details.
model-model from which to simulate; see details.
k1-lower bound for psi(x); see details.
k2-upper bound for psi(x); see details.
phi-the function psi(x) - k1.
max.psi-upper value of the support of psi to search for its maximum.
rh-the rejection function; see details.
A-A(x) is the integral of the drift between 0 and x.

Simulation of stochastic differential equation

- The *sde.sim* function can be used to simulate paths of solutions to generic stochastic differential equations. The function can simulate trajectories with different methods. The function accepts the two coefficients drift and sigma and eventually the partial derivative of the diffusion coefficient *sigma.x* for the *Milstein* scheme.
- If the diffusion coefficient sigma is not specified, it is assumed to be unitary.
- The method of simulation can be one among: euler, KPS, milstein, milstein2, cdist, EA, ozaki, and shoji.
- The model is one among: CIR: Cox-Ingersoll-Ross, VAS: Vasicek, OU: Ornstein-Uhlenbeck, BS: Black and Scholes.

SDE Simulations

```
par(mfrow=c(2,2))
# Ornstein-Uhlenbeck process
set.seed(123)
d <- expression(-5*x)
s <- expression(3.5)
X<-sde.sim(X0=10,drift=d, sigma=s)
plot(X,main="Ornstein-Uhlenbeck")
# Multiple trajectories of the O-U process
set.seed(123)
X<-sde.sim(X0=10,drift=d, sigma=s, M=3)
plot(X,main="Multiple trajectories of O-U")
# Cox-Ingersoll-Ross process
# dXt = (6-3*Xt)*dt + 2*sqrt(Xt)*dWt
set.seed(123)
d <- expression( 6-3*x )
s <- expression( 2*sqrt(x) )
X<-sde.sim(X0=10,drift=d, sigma=s)
plot(X,main="Cox-Ingersoll-Ross")
# Exact simulation
set.seed(123)
d <- expression(sin(x))
d.x <- expression(cos(x))
A <- function(x) 1-cos(x)
X<-sde.sim(method="EA", delta=1/20, X0=0, N=500,
        drift=d, drift.x = d.x, A=A)
plot(X, main="Periodic drift")
```

SDE Simulations

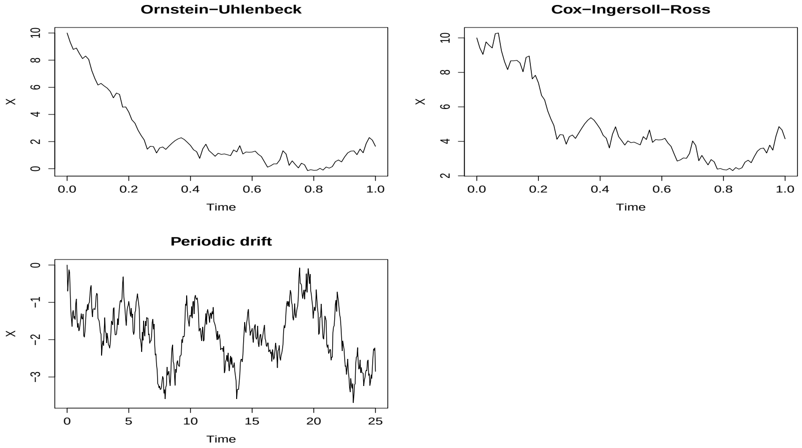


Figure: Generic interface to different methods of simulation of solutions to stochastic differential equations.

SDE Simulations

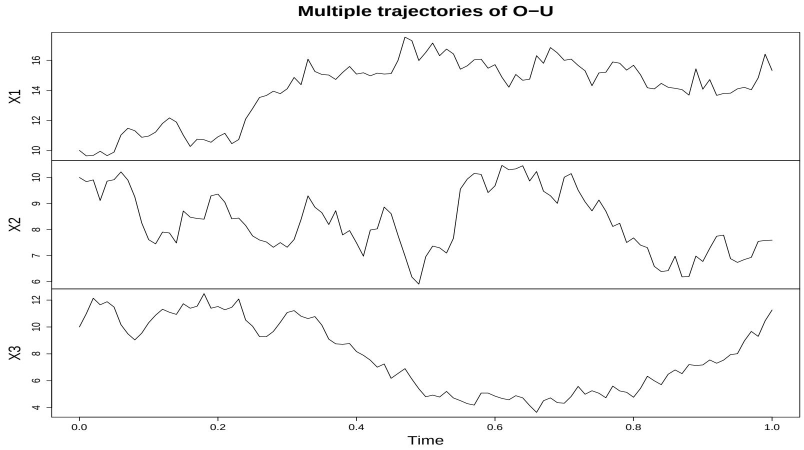


Figure: Generic interface to different methods of simulation of solutions to stochastic differential equations.

Examples

- The Ornstein-Uhlenbeck process

$$dX_t = (\theta_1 - \theta_2 X_t)\, dt + \theta_3 dW_t, X_0 = 10, (\theta_1, \theta_2, \theta_3) = (0, 5, 3.5).$$

- The Cox-Ingersoll-Ross process

$$dX_t = (\theta_1 - \theta_2 X_t)\, dt + \theta_3 \sqrt{X_t} dW_t, X_0 = 10, (\theta_1, \theta_2, \theta_3) = (6, 3, 2).$$

- The Cox-Ingersoll-Ross process with Milstein scheme
  We need the partial derivative with respect to variable x of the coefficient $\sigma(\cdot, \cdot)$,

$$\sigma_x(t, x) = \frac{\partial}{\partial x} 2\sqrt{x} = \frac{1}{\sqrt{x}}.$$

- The Cox-Ingersoll-Ross process with Euler scheme on the transformed process $Y_t = \sqrt{X_t}$

$$dY_t = \frac{1}{2Y_t}\left(\theta_1 - \theta_2 Y_t^2 - \frac{1}{4}\theta_3^2\right) dt + \frac{1}{2}\theta_3 dW_t, Y_0, (\theta_1, \theta_2, \theta_3) = (6, 3, 2).$$

SDE Simulations

```
# Ornstein - Uhlenbeck process
set.seed(123)
d <- expression (-5 * x)
s <- expression (3.5)
X<-sde.sim (X0 =10, drift=d, sigma=s)
par(mfrow=c(2,2))
plot (X, main ="Ornstein-Uhlenbeck")
# Cox-Ingersoll-Ross(CIR-1)
set.seed (123)
d <- expression (6-3*x)
s <- expression (2* sqrt(x))
X<-sde.sim(X0 =10 , drift =d, sigma =s)
plot (X, main ="Cox-Ingersoll-Ross")
# Cox-Ingersoll-Ross(CIR-2)
d <- expression(6-3*x)
s <- expression ( 2* sqrt (x) )
s.x <- expression ( 1/ sqrt (x) )
set.seed (123)
X<-sde.sim (X0=10 , drift=d, sigma=s, sigma.x=s.x,
             method ="milstein")
plot (X, main ="Cox-Ingersoll- Ross")
# Cox-Ingersoll-Ross(CIR-3)
set.seed(123)
d <- expression ((6-3*x^2-1)/(2*x))
s <- expression (1)
Y<-sde.sim (X0= sqrt(10), drift=d, sigma=s)
plot (Y^2, main ="Cox-Ingersoll-Ross")
```

SDE Simulations

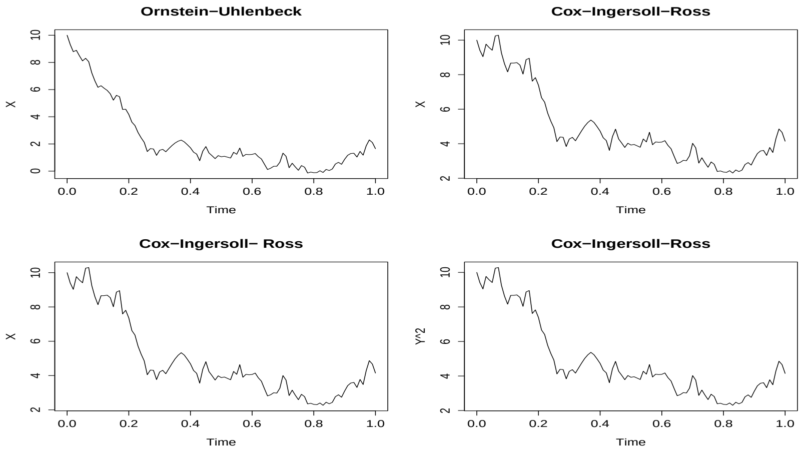


Figure: *sde.sim* function for some of the processes

SDE Simulations

Simulation for $\triangle = 0.1$ and $\triangle = 0.25$ of

$$dX_t = \left(5 - 11X_t + 6X_t^2 - X_t^3\right) dt + dW_t, X_0 = 5$$

```
bX <- expression ((5-11*x+6*x^2-x^3))
x0 <- 5
DT <- 0.1
par(mfrow=c(2,3))
set.seed (123)
X <- sde.sim(drift=bX,delta=DT,X0=x0)
plot (X, main="Euler")
set.seed (123)
Y <- sde.sim(drift=bX,method="ozaki",delta=DT,X0=x0)
plot (Y, main ="Ozaki")
set.seed (123)
Z <- sde.sim(drift=bX,method="shoji",delta=DT,X0=x0)
plot (Z,main ="Shoji-Ozaki")
DT <- 0.25
set.seed(123)
X <- sde.sim(drift=bX,delta=DT,X0=x0)
plot (X, main="Euler")
set.seed (123)
Y <- sde.sim(drift=bX,method="ozaki",delta=DT,X0=x0)
plot (Y,main ="Ozaki")
set.seed(123)
Z <- sde.sim(drift=bX,method="shoji",delta=DT,X0=x0)
plot (Z, main ="Shoji-Ozaki")
```

SDE Simulations

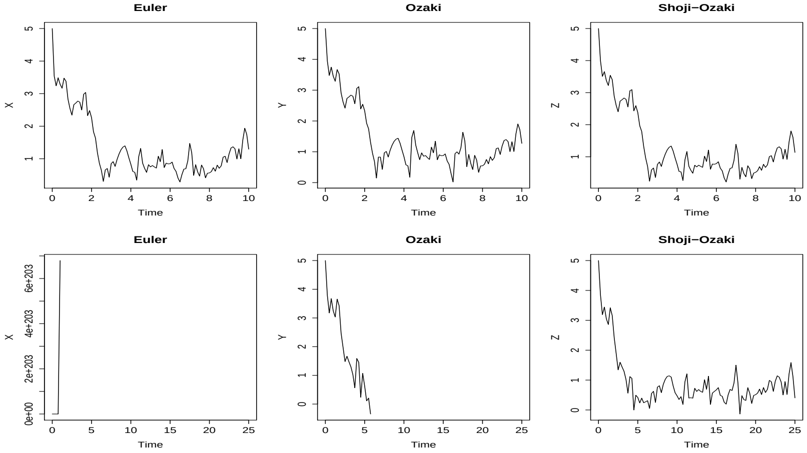


Figure: Different performance of the Euler, Ozaki, and Shoji-Ozaki methods for different values of △ (top: 0.1; bottom: 0.25). The Euler scheme explodes for high values of △.

SDE Simulations

- In the Euler method, the variance V_x is independent from the previous state of the process $X_t = x$ (inheritance of the independence of the increments of the Brownian motion).
- Conversely V_x for the Ozaki and Shoji-Ozaki methods depend on the previous state of the process.
- the Shoji-Ozaki method also takes into account the stochastic behavior of the discretization because of the Itô formula.
- Shoji-Ozaki is more stable if the time $\triangle$ is large.

SDE Simulations(Exact Algorithm)

```
set.seed(123)
d <- expression (-4*tanh(2*x))
d.x <- expression (-(4*(2/cosh(2*x)^2)))
A <- function(x) -(0.5+6/4)*log(cosh(2*x))
X0 <- rt (1, df=4)/2
F <- function(x) log(x+sqrt(1+x^2))/2
Y0 <- F(X0)
Y<-sde.sim (method="EA",delta=1/20,X0=Y0,N=500,drift=d,
               drift.x=d.x, A=A, k1=-4,k2 =8)
X <- sinh(Y)
XY<-ts( cbind (X,Y), start =0, delta =1/ 20)
plot (XY , main ="Original scale X vs transformed Y")
```

SDE Simulations(Exact Algorithm)

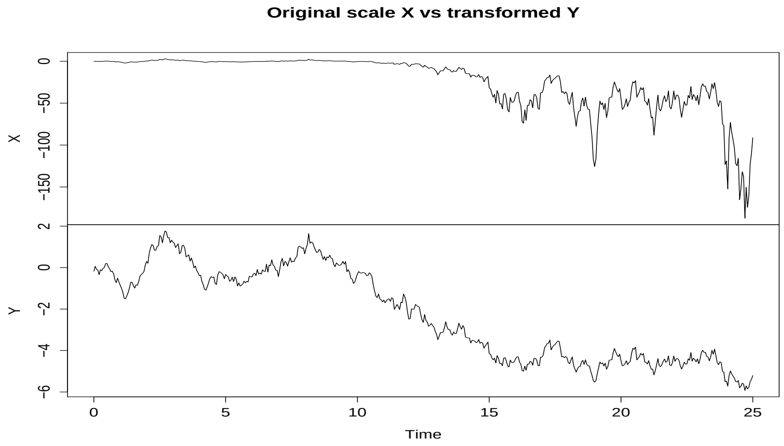


Figure: Simulation of the process Y_t and transformed version $X_t = \sinh 2Y_t$ using the EA1 algorithm.

Other Functions

cpoint(x, mu, sigma): Volatility change-point estimator for diffusion processes based on least squares. The model is assumed to be of the form

$$dX_t = b\left(X_t\right)dt + \theta\sigma\left(X_t\right)dW_t$$

where $theta = theta1$ for $t <= tau0$ and $theta = theta2$ otherwise.

- **x**- a ts object.
  mu- a function of x describing the drift coefficient.
  sigma- a function of x describing the diffusion coefficient.

Other Functions

dcElerian(x, t, x0, t0, theta, d, s, sx, log=FALSE): This function returns the value of the conditional density of $X(t)|X(t_0) = x_0$ at point x. All the functions d, s, and sx must be functions of t, x, and $theta$.

- **x**- vector of quantiles. **t**- lag or time
  x0- the value of the process at time t0. **t0**- initial time.
  theta- parameter of the process.
  log- logical; if TRUE, probabilities p are given as $\log(p)$.
  d- drift coefficient as a function. **s**- diffusion coefficient as a function.
  sx- partial derivative w.r.t. x of the diffusion coefficient.

It has different methods like Euler, Kessler, Ozaki, Shoji, Pedersen.

Other Functions

EULERloglik(X, theta, d, s, log = TRUE): Euler approximation of the likelihood of a process solution of a stochastic differential equation. These functions are useful to calculate approximated maximum likelihood estimators when the transition density of the process is not known.

- **X**- a ts object containing a sample path of an sde.
  theta- vector of parameters.
  d, s- drift and diffusion coefficients.
  log- logical; if TRUE, the log-likelihood is returned.

Other Functions

dcBS(x, Dt, x0, theta, log = FALSE)
pcBS(x, Dt, x0, theta, lower.tail = TRUE, log.p = FALSE)
qcBS(p, Dt, x0, theta, lower.tail = TRUE, log.p = FALSE)
rcBS(n=1, Dt, x0, theta)
Density, distribution function, quantile function, and random generation for the conditional law $X(t)|X(0) = x_0$ of the Black-Scholes-Merton process also known as the geometric Brownian motion process, i.e. $dX_t = \theta_1 X_t dt + \theta_2 X_t dW_t$ with $\theta_3 > 0$.

- **x, p**- vector of quantiles and probabilities
  Dt- lag or time. **x0**- the value of the process at time t.
  theta- parameter of the Black-Scholes-Merton process.
  n- number of random numbers to generate from the conditional distribution.
  log, log.p- logical; if TRUE, probabilities p are given as $\log(p)$.
  lower.tail- logical; if TRUE (default), probabilities are $P[Z <= x]$; otherwise, $P[X > x]$.

Package have same function for Cox-Ingersoll-Ross and Ornstein-Uhlenbeck(Vasicek) processes. Similar function is for stationary law. One also use sde.sim to simulate all methods.

Outline

Bibliography I

S.M. Iacus.
Simulation and Inference for Stochastic Differential Equations With R Examples
Springer Press, 2007.

Y. Luo.
Numerical Solutions for Stochastic Differential Equations and Some Examples
Brigham Young University.