Package 'sde'

October 14, 2022

Type Package

Title Simulation and Inference for Stochastic Differential Equations

Version 2.0.18

Author Stefano Maria Iacus

Depends MASS, stats4, fda, zoo

Maintainer Stefano Maria Iacus <siacus@iq.harvard.edu>

Description Companion package to the book Simulation and Inference for Stochastic Differential Equations With R Examples, ISBN 978-0-387-75838-1, Springer, NY.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-08-09 07:50:02 UTC

R topics documented:

| BM | *Brownian motion, Brownian bridge, and geometric Brownian motion simulators* |

Description

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

Usage

```
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)
```

Arguments

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].

Details

These functions return an invisible ts object containing a trajectory of the process calculated on a grid of N+1 equidistant points between t0 and T; i.e., t[i] = t0 + (T-t0)*i/N, i in 0:N. t0=0 for the geometric Brownian motion.

The function BBridge returns a trajectory of the Brownian bridge starting at x at time t0 and ending at y at time T; i.e.,

$$\{B(t), t_0 \le t \le T | B(t_0) = x, B(T) = y\}.$$

The function BM returns a trajectory of the translated Brownian motion $B(t), t \geq 0|B(t_0) = x$; i.e., $x + B(t - t_0)$ for t >= t0. The standard Brownian motion is obtained choosing x=0 and t0=0 (the default values).

The function GBM returns a trajectory of the geometric Brownian motion starting at x at time t0=0; i.e., the process

$$S(t) = x \exp\{(r - \sigma^2/2)t + \sigma B(t)\}.$$

Value

X   an invisible ts object

Author(s)

Stefano Maria Iacus

Examples

```
plot(BM())
plot(BBridge())
plot(GBM())
```

---

cpoint   *Volatility change-point estimator for diffusion processes*

Description

Volatility change-point estimator for diffusion processes based on least squares.

Usage

```
cpoint(x, mu, sigma)
```

Arguments

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

Details

The function returns a list of elements containing the discrete k0 and continuous tau0 change-point instant, the estimated volatilities before (theta1) and after (theta2) the time change. The model is assumed to be of the form

$$\mathrm{d}X_t = b(X_t)\mathrm{d}t + \theta\sigma(X_t)\mathrm{d}W_t$$

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

If the drift coefficient is unknown, the model

$$\mathrm{d}X_t = b(X_t)\mathrm{d}t + \theta\mathrm{d}W_t$$

is considered and b is estimated nonparametrically.

Value

X   a list

Author(s)

Stefano Maria Iacus

Examples

```
tau0 <- 0.6
k0 <- ceiling(1000*tau0)
set.seed(123)
X1 <- sde.sim(X0=1, N=2*k0, t0=0, T=tau0, model="CIR",
              theta=c(6,2,1))
X2 <- sde.sim(X0=X1[2*k0+1], N=2*(1000-k0), t0=tau0,
   T=1, model="CIR", theta=c(6,2,3))

Y <- ts(c(X1,X2[-1]), start=0, deltat=deltat(X1))
X <- window(Y,deltat=0.01)
DELTA <- deltat(X)
n <- length(X)

mu <- function(x) 6-2*x
sigma <- function(x) sqrt(x)

cp <- cpoint(X,mu,sigma)
cp
plot(X)
abline(v=tau0,lty=3)
abline(v=cp$tau0,col="red")

# nonparametric estimation
cpoint(X)
```

DBridge   *Simulation of diffusion bridge*

Description

Simulation of diffusion bridge.

Usage

```
DBridge(x=0, y=0, t0=0, T=1, delta, drift, sigma, ...)
```

Arguments

x	initial value of the process at time `t0`.
y	terminal value of the process at time `T`.
t0	initial time.
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`.
T	final time.
...	passed to the `sde.sim` function.

Details

The function returns a trajectory of the diffusion bridge starting at x at time `t0` and ending at y at time `T`.

The function uses the `sde.sim` function to simulate the paths internally. Refer to the `sde.sim` documentation for further information about the argument "...".

Value

X	an invisible `ts` object

Author(s)

Stefano Maria Iacus

References

Bladt, M., Soerensen, M. (2007) Simple simulation of diffusion bridges with application to likelihood inference for diffusions, mimeo.

See Also

sde.sim, BBridge

Examples

```
d <- expression((3-x))
s <- expression(1.2*sqrt(x))
par(mar=c(3,3,1,1))
par(mfrow=c(2,1))
set.seed(123)
X <- DBridge(x=1.7,y=0.5, delta=0.01, drift=d, sigma=s)
plot(X)
X <- DBridge(x=1,y=5, delta=0.01, drift=d, sigma=s)
plot(X)
```

---

dcElerian	*Approximated conditional law of a diffusion process by Elerian's method*

Description

Approximated conditional densities for $X(t)|X(t_0) = x_0$ of a diffusion process.

Usage

```
dcElerian(x, t, x0, t0, theta, d, s, sx, log=FALSE)
```

Arguments

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

Details

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.

Value

x	a numeric vector

Author(s)

Stefano Maria Iacus

References

Elerian, O. (1998) A note on the existence of a closed form conditional density for the Milstein scheme, *Working Paper, Nuffield College, Oxford University*. Available at http://www.nuff.ox.ac.uk/economics/papers/

| dcEuler | *Approximated conditional law of a diffusion process* |

Description

Approximated conditional densities for $X(t)|X(t_0) = x_0$ of a diffusion process.

Usage

```
dcEuler(x, t, x0, t0, theta, d, s, log=FALSE)
```

Arguments

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

Details

This function returns the value of the conditional density of $X(t)|X(t_0) = x_0$ at point x.

The functions d and s must be functions of t, x, and theta.

Value

x	a numeric vector

Author(s)

Stefano Maria Iacus

| dcKessler | *Approximated conditional law of a diffusion process by Kessler's method* |

Description

Approximated conditional densities for $X(t)|X(t_0) = x_0$ of a diffusion process.

Usage

```
dcKessler(x, t, x0, t0, theta, d, dx, dxx, s, sx, sxx,
          log=FALSE)
```

Arguments

x	vector of quantiles.
t	lag or time.
x0	the value of the process at time t0; see details.
t0	initial time.
theta	parameter of the process; see details.
log	logical; if TRUE, probabilities p are given as $\log(p)$.
d	drift coefficient as a function; see details.
dx	partial derivative w.r.t. x of the drift coefficient; see details.
dxx	second partial derivative wrt x^2 of the drift coefficient; see details.
s	diffusion coefficient as a function; see details.
sx	partial derivative w.r.t. x of the diffusion coefficient; see details.
sxx	second partial derivative w.r.t. x^2 of the diffusion coefficient; see details.

Details

This function returns the value of the conditional density of $X(t)|X(t_0) = x_0$ at point x.

All the functions d, dx, dxx, dt, s, sx, and sxx must be functions of t, x, and theta.

Value

x	a numeric vector

Author(s)

Stefano Maria Iacus

References

Kessler, M. (1997) Estimation of an ergodic diffusion from discrete observations, *Scand. J. Statist.*, 24, 211-229.

---

dcOzaki	*Approximated conditional law of a diffusion process by Ozaki's method*

Description

Approximated conditional densities for $X(t)|X(t_0) = x_0$ of a diffusion process.

Usage

```
dcOzaki(x, t, x0, t0, theta, d, dx, s, log=FALSE)
```

Arguments

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

Details

This function returns the value of the conditional density of $X(t)|X(t_0) = x_0$ at point x.

All the functions d, dx, and s must be functions of t, x, and theta.

Value

x	a numeric vector

Author(s)

Stefano Maria Iacus

References

Ozaki, T. (1992) A bridge between nonlinear time series models and nonlinear stochastic dynamical systems: A local linearization approach, *Statistica Sinica*, 2, 25-83.

`dcShoji`	*Approximated conditional law of a diffusion process by the Shoji-Ozaki method*

Description

Approximated conditional densities for $X(t)|X(t_0) = x_0$ of a diffusion process.

Usage

```
dcShoji(x, t, x0, t0, theta, d, dx, dxx, dt, s, log=FALSE)
```

Arguments

x	vector of quantiles.
t	lag or time.
x0	the value of the process at time t0; see details.
t0	initial time.
theta	parameter of the process; see details.
log	logical; if TRUE, probabilities p are given as $\log(p)$.
d	drift coefficient as a function; see details.
dx	partial derivative w.r.t. x of the drift coefficient; see details.
dxx	second partial derivative w.r.t. x^2 of the drift coefficient; see details.
dt	partial derivative w.r.t. t of the drift coefficient; see details.
s	diffusion coefficient as a function; see details.

Details

This function returns the value of the conditional density of $X(t)|X(t_0) = x_0$ at point x.

All the functions d, dx, dxx, dt, and s must be functions of t, x, and theta.

Value

x	a numeric vector

Author(s)

Stefano Maria Iacus

References

Shoji, L., Ozaki, T. (1998) Estimation for nonlinear stochastic differential equations by a local linearization method, *Stochastic Analysis and Applications*, 16, 733-752.

---

dcSim   *Pedersen's simulated transition density*

Description

Simulated transition density $X(t)|X(t_0) = x_0$X(t) | X(t0) = x0 of a diffusion process based on Pedersen's method.

Usage

```
dcSim(x0, x, t, d, s, theta, M=10000, N=10, log=FALSE)
```

Arguments

x0	the value of the process at time 0.
x	value in which to evaluate the conditional density.
t	lag or time.
theta	parameter of the process; see details.
log	logical; if TRUE, probabilities p are given as $\log(p)$.
d	drift coefficient as a function; see details.
s	diffusion coefficient as a function; see details.
N	number of subintervals; see details.
M	number of Monte Carlo simulations, which should be an even number; see details.

Details

This function returns the value of the conditional density of $X(t)|X(0) = x_0$ at point x.

The functions d and s, must be functions of t, x, and theta.

Value

x	a numeric vector

Author(s)

Stefano Maria Iacus

References

Pedersen, A. R. (1995) A new approach to maximum likelihood estimation for stochastic differential equations based on discrete observations, *Scand. J. Statist.*, 22, 55-71.

Examples

```
## Not run:
d1 <- function(t,x,theta) theta[1]*(theta[2]-x)
s1 <- function(t,x,theta) theta[3]*sqrt(x)

from <- 0.08
x <- seq(0,0.2, length=100)
sle10 <- NULL
sle2 <- NULL
sle5 <- NULL
true <- NULL
set.seed(123)
for(to in x){
 sle2 <- c(sle2, dcSim(from, to, 0.5, d1, s1,
    theta=c(2,0.02,0.15), M=50000,N=2))
 sle5 <- c(sle5, dcSim(from, to, 0.5, d1, s1,
    theta=c(2,0.02,0.15), M=50000,N=5))
 sle10 <- c(sle10, dcSim(from, to, 0.5, d1, s1,
    theta=c(2,0.02,0.15), M=50000,N=10))
 true <- c(true, dcCIR(to, 0.5, from, c(2*0.02,2,0.15)))
}

par(mar=c(5,5,1,1))
plot(x, true, type="l", ylab="conditional density")
lines(x, sle2, lty=4)
lines(x, sle5, lty=2)
lines(x, sle10, lty=3)
legend(0.15,20, legend=c("exact","N=2", "N=5", "N=10"),
    lty=c(1,2,4,3))

## End(Not run)
```

DWJ   *Weekly closings of the Dow-Jones industrial average*

Description

This dataset contains the weekly closings of the Dow-Jones industrial average in the period July 1971–August 1974. These data were proposed to test change-point estimators. There are 162 data, and the main evidence found by several authors is that a change in the variance occurred around the third week of March 1973.

Usage

```
data(DWJ)
```

References

Hsu, D.A. (1977) Tests for variance shift at an unknown time point, *Appl. Statist.*, 26(3), 279-284.

Hsu, D.A. (1979) Detecting shifts of parameter in gamma sequences with applications to stock price and air traffic flow analysis, *Journal American Stat. Ass.*, 74(365), 31-40.

Examples

```
data(DWJ)
ret <- diff(DWJ)/DWJ[-length(DWJ)]

par(mfrow=c(2,1))
par(mar=c(3,3,2,1))
plot(DWJ,main="Dow-Jones closings",ylab="",type="p")
plot(ret,main="Dow-Jones returns",ylab="",type="p")

cp <- cpoint(ret)
cp
abline(v=cp$tau0,lty=3)

cp <- cpoint(window(ret,end=cp$tau0))
cp
abline(v=cp$tau0,lty=3)
```

EULERloglik	*Euler approximation of the likelihood*

Description

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.

Usage

```
EULERloglik(X, theta, d, s, log = TRUE)
```

Arguments

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

Details

The function `EULERloglik` returns the Euler approximation of the log-likelihood. The functions `s` and `d` are the drift and diffusion coefficients with arguments (`t`,`x`,`theta`).

Value

x   a number

Author(s)

Stefano Maria Iacus

Examples

```
set.seed(123)
d <- expression(-1*x)
s <- expression(2)
sde.sim(drift=d, sigma=s) -> X

S <- function(t, x, theta) sqrt(theta[2])
B <- function(t, x, theta) -theta[1]*x

true.loglik <- function(theta){
 DELTA <- deltat(X)
 lik <- 0
 for(i in 2:length(X))
  lik <- lik + dnorm(X[i], mean=X[i-1]*exp(-theta[1]*DELTA),
  sd = sqrt((1-exp(-2*theta[1]*DELTA))*
            theta[2]/(2*theta[1])),TRUE)
 lik
}

xx <- seq(-3,3,length=100)
sapply(xx, function(x) true.loglik(c(x,4))) -> py
sapply(xx, function(x) EULERloglik(X,c(x,4),B,S)) -> pz

# true likelihood
plot(xx,py,type="l",xlab=expression(beta),ylab="log-likelihood")
lines(xx,pz, lty=2) # Euler
```

---

gmm   *Generalized method of moments estimator*

Description

Implementation of the estimator of the generalized method of moments by Hansen.

Usage

```
gmm(X, u, dim, guess, lower, upper, maxiter=30, tol1=1e-3,
    tol2=1e-3)
```

Arguments

`X`	a ts object containing a sample path of an sde.
`u`	a function of x, y, and `theta` and `DELTA`; see details.
`dim`	dimension of parameter space; see details.
`guess`	initial value of the parameters; see details.
`lower`	lower bounds for the parameters; see details.
`upper`	upper bounds for the parameters; see details.
`tol1`	tolerance for parameters; see details.
`tol2`	tolerance for Q1; see details.
`maxiter`	maximum number of iterations at the second stage; see details.

Details

The function gmm minimizes at the first stage the function `Q(theta) = t(Gn(theta)) * Gn(theta)` with respect to `theta`, where `Gn(theta) = mean(u(X[i+1], X[i], theta))`. Then a matrix of weights `W` is obtained by inverting an estimate of the long-run covariance and the quadratic function `Q1(theta) = t(Gn(theta)) * W * Gn(theta)` with starting value `theta1` (the solution at the first stage). The second stage is iterated until the first of these conditions verifies: (1) that the number of iterations reaches `maxiter`; (2) that the Euclidean distance between `theta1` and `theta2 < tol1`; (3) that `Q1 < tol2`.

The function u must be a function of (`u`,`y`,`theta`,`DELTA`) and should return a vector of the same length as the dimension of the parameter space. The sanity checks are left to the user.

Value

`x`	a list with parameter estimates, the value of `Q1` at the minimum, and the Hessian

Author(s)

Stefano Maria Iacus

References

Hansen, L.P. (1982) Large Sample Properties of Generalized Method of Moments Estimators, *Econometrica*, 50(4), 1029-1054.

Examples

```
## Not run:
alpha <- 0.5
beta <- 0.2
sigma <- sqrt(0.05)
true <- c(alpha, beta, sigma)
names(true) <- c("alpha", "beta", "sigma")

x0 <- rsCIR(1,theta=true)
set.seed(123)
```

```
sde.sim(X0=x0,model="CIR",theta=true,N=500000,delta=0.001) -> X
X <- window(X, deltat=0.1)
DELTA = deltat(X)
n <- length(X)
X <- window(X, start=n*DELTA*0.5)
plot(X)

u <- function(x, y, theta, DELTA){
  c.mean <- theta[1]/theta[2] +
            (y-theta[1]/theta[2])*exp(-theta[2]*DELTA)
  c.var <- ((y*theta[3]^2 *
     (exp(-theta[2]*DELTA)-exp(-2*theta[2]*DELTA))/theta[2] +
  theta[1]*theta[3]^2*
  (1-exp(-2*theta[2]*DELTA))/(2*theta[2]^2)))
  cbind(x-c.mean,y*(x-c.mean), c.var-(x-c.mean)^2,
       y*(c.var-(x-c.mean)^2))
}

CIR.lik <- function(theta1,theta2,theta3) {
 n <- length(X)
 dt <- deltat(X)
 -sum(dcCIR(x=X[2:n], Dt=dt, x0=X[1:(n-1)],
   theta=c(theta1,theta2,theta3), log=TRUE))
}

fit <- mle(CIR.lik, start=list(theta1=.1,  theta2=.1,theta3=.3),
    method="L-BFGS-B",lower=c(0.001,0.001,0.001), upper=c(1,1,1))
# maximum likelihood estimates
coef(fit)


gmm(X,u, guess=as.numeric(coef(fit)), lower=c(0,0,0),
    upper=c(1,1,1))

true

## End(Not run)
```

HPloglik	*Ait-Sahalia Hermite polynomial expansion approximation of the likelihood*

Description

Ait-Sahalia Hermite polynomial expansion and 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.

Usage

```
HPloglik(X, theta, M, F, s, log=TRUE)
```

Arguments

X	a ts object containing a sample path of an sde.
theta	vector of parameters.
M	list of derivatives; see details.
F	the transform function; see details.
s	drift and diffusion coefficient; see details.
log	logical; if TRUE, the log-likelihood is returned.

Details

The function `HPloglik` returns the Hermite polynomial approximation of the likelihood of a diffusion process transformed to have a unitary diffusion coefficient. The function `F` is the transform function, and `s` is the original diffusion coefficient. The list of functions `M` contains the transformed drift in `M[[1]]` and the subsequent six derivatives in `x` of `M[[1]]`. The functions `F`, `s`, and `M` have arguments `(t,x,theta)`.

Value

x	a number

Author(s)

Stefano Maria Iacus

References

Ait-Sahalia, Y. (1996) Testing Continuous-Time Models of the Spot Interest Rate, *Review of Financial Studies*, 9(2), 385-426.

Examples

```
set.seed(123)
d <- expression(-1*x)
s <- expression(2)
sde.sim(drift=d, sigma=s) -> X

M0 <- function(t, x, theta) -theta[1]*x
M1 <- function(t, x, theta) -theta[1]
M2 <- function(t, x, theta) 0
M3 <- function(t, x, theta) 0
M4 <- function(t, x, theta) 0
M5 <- function(t, x, theta) 0
M6 <- function(t, x, theta) 0
mu <- list(M0, M1, M2, M3, M4, M5, M6)

F <- function(t, x, theta) x/sqrt(theta[2])
S <- function(t, x, theta) sqrt(theta[2])

true.loglik <- function(theta) {
```

```
 DELTA <- deltat(X)
 lik <- 0
 for(i in 2:length(X))
  lik <- lik + dnorm(X[i], mean=X[i-1]*exp(-theta[1]*DELTA),
   sd = sqrt((1-exp(-2*theta[1]*DELTA))*theta[2]/
             (2*theta[1])),TRUE)
 lik
}

xx <- seq(-3,3,length=100)
sapply(xx, function(x) HPloglik(X,c(x,4),mu,F,S)) -> px
sapply(xx, function(x) true.loglik(c(x,4))) -> py

plot(xx,px,type="l",xlab=expression(beta),ylab="log-likelihood")
lines(xx,py, lty=3) # true
```

---

ksmooth	*Nonparametric invariant density, drift, and diffusion coefficient estimation*

Description

Implementation of simple Nadaraya-Watson nonparametric estimation of drift and diffusion coefficient, and plain kernel density estimation of the invariant density for a one-dimensional diffusion process.

Usage

```
ksdrift(x, bw, n = 512)
ksdiff(x, bw, n = 512)
ksdens(x, bw, n = 512)
```

Arguments

x	a ts object.
bw	bandwidth.
n	number of points in which to calculate the estimates.

Details

These functions return the nonparametric estimate of the drift or diffusion coefficients for data x using the Nadaraya-Watson estimator for diffusion processes.

ksdens returns the density estimates of the invariant density.

If not provided, the bandwidth bw is calculated using Scott's rule (i.e., bw = len^(-1/5)*sd(x)) where len=length(x) is the number of observed points of the diffusion path.

Value

val   an invisible list of x and y coordinates and an object of class `density` in the case
               of invariant density estimation

Author(s)

Stefano Maria Iacus

References

Ait-Sahalia, Y. (1996) Nonparametric pricing of interest rate derivative securities, *Econometrica*, 64, 527-560.

Bandi, F., Phillips, P. (2003) Fully nonparametric estimation of scalar diffusion models, *Econometrica*, 71, 241-283.

Florens-Zmirou, D. (1993) On estimating the diffusion coefficient from discrete observations, *Journal of Applied Probability*, 30, 790-804.

Examples

```
set.seed(123)
theta <- c(6,2,1)
X <- sde.sim(X0 = rsCIR(1, theta), model="CIR", theta=theta,
     N=1000,delta=0.1)

b <- function(x)
 theta[1]-theta[2]*x

sigma <- function(x)
 theta[3]*sqrt(x)

minX <- min(X)
maxX <- max(X)

par(mfrow=c(3,1))
curve(b,minX,maxX)
lines(ksdrift(X),lty=3)

curve(sigma,minX, maxX)
lines(ksdiff(X),lty=3)

f <-function(x) dsCIR(x, theta)
curve(f,minX,maxX)
lines(ksdens(X),lty=3)
```

---

linear.mart.ef   *Linear martingale estimating function*

Description

Apply a linear martingale estimating function to find estimates of the parameters of a process solution of a stochastic differential equation.

Usage

```
linear.mart.ef(X, drift, sigma, a1, a2, guess, lower, upper,
       c.mean, c.var)
```

Arguments

X	a ts object containing a sample path of an sde.
drift	an expression for the drift coefficient; see details.
sigma	an expression for the diffusion coefficient; see details.
a1, a2	weights or instruments.
c.mean	expressions for the conditional mean.
c.var	expressions for the conditional variance.
guess	initial value of the parameters; see details.
lower	lower bounds for the parameters; see details.
upper	upper bounds for the parameters; see details.

Details

The function linear.mart.ef minimizes a linear martingale estimating function that is a particular case of the polynomial martingale estimating functions.

Value

x	a vector of estimates

Author(s)

Stefano Maria Iacus

References

Bibby, B., Soerensen, M. (1995) Martingale estimating functions for discretely observed diffusion processes, *Bernoulli*, 1, 17-39.

Examples

```
set.seed(123)
d <- expression(-1 * x)
s <- expression(1)
x0 <- rnorm(1,sd=sqrt(1/2))
sde.sim(X0=x0,drift=d, sigma=s,N=1000,delta=0.1) -> X

d <- expression(-theta * x)

linear.mart.ef(X, d, s, a1=expression(-x), lower=0, upper=Inf,
  c.mean=expression(x*exp(-theta*0.1)),
  c.var=expression((1-exp(-2*theta*0.1))/(2*theta)))
```

MOdist   *Markov Operator distance for clustering diffusion processes.*

Description

Markov Operator distance for clustering diffusion processes.

Usage

```
MOdist(x, M=50, rangeval=range(x, na.rm=TRUE, finite = TRUE))
```

Arguments

x	one or multi-dimensional time series.
M	number of splines bases used to approximate the Markov Operator.
rangeval	a vector containing lower and upper limit. Default is the range of x.

Details

This function return a lower triangular dist object to be further used in cluster analysis (see examples below).

If x is a one-dimensional time series, the output is the scalar 0, not a dist object.

If x has less than 2 observations, NA is returned.

If time series x contains missing data, then x is converted to a zoo object and missing data are imputed by interpolation.

Value

X	a dist object

Author(s)

Stefano Maria Iacus

References

De Gregorio, A. Iacus, S.M. (2008) Clustering of discretely observed diffusion processes, Computational Statistics and Data Analysis, 54(12), 598-606, doi:10.1016/j.csda.2009.10.005.

Examples

```
## Not run:
data(quotes)

plot(quotes)

d <- MOdist(quotes)
cl <- hclust( d )
groups <- cutree(cl, k=4)

cmd <- cmdscale(d)
plot( cmd, col=groups)
text( cmd, labels(d) , col=groups)

plot(quotes, col=groups)

plot(quotes, col=groups,ylim=range(quotes))

## End(Not run)
```

---

quotes   *Daily closings of 20 financial time series from 2006-01-03 to 2007-12-31*

Description

This dataset contains the daily closings of 20 assets from NYSE/NASDAQ. Quotes from 2006-01-03 to 2007-12-31.

It is an object of class zoo. Original data contained missing data and interpolated. Used as example data to test the Markov Operator distance for discretely observed diffusion processes.

Usage

```
data(quotes)
```

References

De Gregorio, A. Iacus, S.M. (2008) Clustering of discretely observed diffusion processes, Computational Statistics and Data Analysis, 54(12), 598-606, doi:10.1016/j.csda.2009.10.005.

Examples

```
data(quotes)

plot(quotes)

d <- MOdist(quotes)
cl <- hclust( d )
groups <- cutree(cl, k=4)

cmd <- cmdscale(d)
plot( cmd, col=groups)
text( cmd, labels(d) , col=groups)

plot(quotes, col=groups)

plot(quotes, col=groups,ylim=range(quotes))
```

---

rcBS	*Black-Scholes-Merton or geometric Brownian motion process conditional law*

Description

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.

Usage

```
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)
```

Arguments

x	vector of quantiles.
p	vector of probabilities.
Dt	lag or time.
x0	the value of the process at time t; see details.
theta	parameter of the Black-Scholes-Merton process; see details.
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[X <= x]; otherwise, P[X > x].

Details

This function returns quantities related to the conditional law of the process solution of

$$\mathrm{d}X_t = \theta_1 X_t \mathrm{d}t + \theta_2 X_t \mathrm{d}W_t.$$

Constraints: $\theta_3 > 0$.

Value

x   a numeric vector

Author(s)

Stefano Maria Iacus

References

Black, F., Scholes, M.S. (1973) The pricing of options and corporate liabilities, *Journal of Political Economy*, 81, 637-654.

Merton, R. C. (1973) Theory of rational option pricing, *Bell Journal of Economics and Management Science*, 4(1), 141-183.

Examples

```
rcBS(n=1, Dt=0.1, x0=1, theta=c(2,1))
```

---

rcCIR   *Conditional law of the Cox-Ingersoll-Ross process*

Description

Density, distribution function, quantile function and random generation for the conditional law $X(t + D_t)|X(t) = x_0$ of the Cox-Ingersoll-Ross process.

Usage

```
dcCIR(x, Dt, x0, theta, log = FALSE)
pcCIR(x, Dt, x0, theta, lower.tail = TRUE, log.p = FALSE)
qcCIR(p, Dt, x0, theta, lower.tail = TRUE, log.p = FALSE)
rcCIR(n=1, Dt, x0, theta)
```

Arguments

x	vector of quantiles.
p	vector of probabilities.
Dt	lag or time.
x0	the value of the process at time t; see details.
theta	parameter of the Ornstein-Uhlenbeck process; see details.
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[X <= x]; otherwise P[X > x].

Details

This function returns quantities related to the conditional law of the process solution of

$$\mathrm{d}X_t = (\theta_1 - \theta_2 X_t)\mathrm{d}t + \theta_3\sqrt{X_t}\mathrm{d}W_t.$$

Constraints: $2\theta_1 > \theta_3^2$, all θ positive.

Value

x	a numeric vector

Author(s)

Stefano Maria Iacus

References

Cox, J.C., Ingersoll, J.E., Ross, S.A. (1985) A theory of the term structure of interest rates, *Econometrica*, 53, 385-408.

See Also

[rsCIR](#)

Examples

```
rcCIR(n=1, Dt=0.1, x0=1, theta=c(6,2,2))
```

---

rcOU   *Ornstein-Uhlenbeck or Vasicek process conditional law*

Description

Density, distribution function, quantile function, and random generation for the conditional law $X(t + D_t)|X(t) = x_0$ of the Ornstein-Uhlenbeck process, also known as the Vasicek process.

Usage

```
dcOU(x, Dt, x0, theta, log = FALSE)
pcOU(x, Dt, x0, theta, lower.tail = TRUE, log.p = FALSE)
qcOU(p, Dt, x0, theta, lower.tail = TRUE, log.p = FALSE)
rcOU(n=1, Dt, x0, theta)
```

Arguments

x	vector of quantiles.
p	vector of probabilities.
Dt	lag or time.
x0	the value of the process at time t; see details.
theta	parameter of the Ornstein-Uhlenbeck process; see details.
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[X <= x]; otherwise P[X > x].

Details

This function returns quantities related to the conditional law of the process solution of

$$\mathrm{d}X_t = (\theta_1 - \theta_2 X_t)\mathrm{d}t + \theta_3\mathrm{d}W_t.$$

Constraints: $\theta_2 > 0, \theta_3 > 0$.

Please note that the process is stationary only if $\theta_2 > 0$.

Value

x	a numeric vector

Author(s)

Stefano Maria Iacus

References

Uhlenbeck, G. E., Ornstein, L. S. (1930) On the theory of Brownian motion, *Phys. Rev.*, 36, 823-841.

Vasicek, O. (1977) An Equilibrium Characterization of the Term Structure, *Journal of Financial Economics*, 5, 177-188.

See Also

rsOU

Examples

```
rcOU(n=1, Dt=0.1, x0=1, theta=c(0,2,1))
```

---

rsCIR   *Cox-Ingersoll-Ross process stationary law*

Description

Density, distribution function, quantile function, and random generation of the stationary law for the Cox-Ingersoll-Ross process.

Usage

```
dsCIR(x, theta, log = FALSE)
psCIR(x, theta, lower.tail = TRUE, log.p = FALSE)
qsCIR(p, theta, lower.tail = TRUE, log.p = FALSE)
rsCIR(n=1, theta)
```

Arguments

x	vector of quantiles.
p	vector of probabilities.
theta	parameter of the Cox-Ingersoll-Ross process; see details.
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[X <= x]; otherwise P[X > x].

Details

This function returns quantities related to the stationary law of the process solution of

$$\mathrm{d}X_t = (\theta_1 - \theta_2 X_t)\mathrm{d}t + \theta_3 \sqrt{X_t}\mathrm{d}W_t.$$

Constraints: $2\theta_1 > \theta_3^2$, all θ positive.

Value

x   a numeric vector

Author(s)

Stefano Maria Iacus

References

Cox, J.C., Ingersoll, J.E., Ross, S.A. (1985) A theory of the term structure of interest rates, *Econometrica*, 53, 385-408.

See Also

rsCIR

Examples

```
rsCIR(n=1, theta=c(6,2,1))
```

---

rsOU   *Ornstein-Uhlenbeck or Vasicek process stationary law*

Description

Density, distribution function, quantile function, and random generation for the stationary law of the Ornstein-Uhlenbeck process also known as the Vasicek process.

Usage

```
dsOU(x, theta, log = FALSE)
psOU(x, theta, lower.tail = TRUE, log.p = FALSE)
qsOU(p, theta, lower.tail = TRUE, log.p = FALSE)
rsOU(n=1, theta)
```

Arguments

x	vector of quantiles.
p	vector of probabilities.
theta	parameter of the Ornstein-Uhlenbeck process; see details.
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[X <= x]; otherwise P[X > x].

Details

This function returns quantities related to the stationary law of the process solution of

$$\mathrm{d}X_t = (\theta_1 - \theta_2 X_t)\mathrm{d}t + \theta_3 \mathrm{d}W_t.$$

Contraints: $theta_2 > 0, \theta_3 > 0$.

Please note that the process is stationary only if $\theta_2 > 0$.

Value

x   a numeric vector

Author(s)

Stefano Maria Iacus

References

Uhlenbeck, G. E., Ornstein, L. S. (1930) On the theory of Brownian motion, *Phys. Rev.*, 36, 823-841.

Vasicek, O. (1977) An Equilibrium Characterization of the Term Structure, *Journal of Financial Economics*, 5, 177-188.

See Also

rcOU

Examples

```
rsOU(n=1, theta=c(0,2,1))
```

---

sde.sim   *Simulation of stochastic differential equation*

Description

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

Usage

```
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)
```

Arguments

`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`.
`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`.

Details

The function returns a `ts` object of length `N+1`; i.e., `X0` and the new `N` simulated values if `M=1`. For `M>1`, an `mts` (multidimensional `ts` object) is returned, which means that `M` independent trajectories are simulated. If the initial value `X0` is not of the length `M`, the values are recycled in order to have an initial vector of the correct length. If `delta` is not specified, then `delta = (T-t0)/N`. If `delta` is

specified, then N values of the solution of the sde are generated and the time horizon T is adjusted to be N * delta.

The function psi is psi(x) = 0.5*drift(x)^2 + 0.5*drift.x(x).

If any of drift.x, drift.xx, drift.t, sigma.x, and sigma.xx are not specified, then numerical derivation is attempted when needed.

If sigma is not specified, it is assumed to be the constant function 1.

The method of simulation can be one among: euler, KPS, milstein, milstein2, cdist, EA, ozaki, and shoji. No assumption on the coefficients or on cdist is checked: the user is responsible for using the right method for the process object of simulation.

The model is one among: CIR: Cox-Ingersoll-Ross, VAS: Vasicek, OU Ornstein-Uhlenbeck, BS: Black and Scholes. No assumption on the coefficient theta is checked: the user is responsible for using the right ones.

If the method is cdist, then the process is simulated according to its known conditional distribution. The random generator rcdist must be a function of n, the number of random numbers; dt, the time lag; x, the value of the process at time t - dt; and the vector of parameters theta.

For the exact algorithm method EA: if missing k1 and k2 as well as A, rh and phi are calculated numerically by the function.

Value

x   returns an invisible ts object

Author(s)

Stefano Maria Iacus

References

See Chapter 2 of the text.

Examples

```
# Ornstein-Uhlenbeck process
set.seed(123)
d <- expression(-5 * x)
s <- expression(3.5)
sde.sim(X0=10,drift=d, sigma=s) -> X
plot(X,main="Ornstein-Uhlenbeck")

# Multiple trajectories of the O-U process
set.seed(123)
sde.sim(X0=10,drift=d, sigma=s, M=3) -> X
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) )
```

```
sde.sim(X0=10,drift=d, sigma=s) -> X
plot(X,main="Cox-Ingersoll-Ross")

# Cox-Ingersoll-Ross using the conditional distribution "rcCIR"

set.seed(123)
sde.sim(X0=10, theta=c(6, 3, 2), rcdist=rcCIR,
        method="cdist") -> X
plot(X, main="Cox-Ingersoll-Ross")

set.seed(123)
sde.sim(X0=10, theta=c(6, 3, 2), model="CIR") -> X
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)
sde.sim(method="EA", delta=1/20, X0=0, N=500,
        drift=d, drift.x = d.x, A=A) -> X
plot(X, main="Periodic drift")
```

---

sdeAIC   *Akaike's information criterion for diffusion processes*

Description

Implementation of the AIC statistics for diffusion processes.

Usage

```
sdeAIC(X, theta, b, s, b.x, s.x, s.xx, B, B.x, H, S, guess,
       ...)
```

Arguments

X	a ts object containing a sample path of an sde.
theta	a vector or estimates of the parameters.
b	drift coefficient of the model as a function of x and theta.
s	diffusion coefficient of the model as a function of x and theta.
b.x	partial derivative of b as a function of x and theta.
s.x	partial derivative of s as a function of x and theta.
s.xx	second-order partial derivative of s as a function of x and theta.
B	initial value of the parameters; see details.
B.x	partial derivative of B as a function of x and theta.

H	function of (x,y), the integral of B/s; optional.
S	function of (x,y), the integral of 1/s; optional.
guess	initial value for the parameters to be estimated; optional.
...	passed to the optim function; optional.

Details

The sdeAIC evaluates the AIC statistics for diffusion processes using Dacunha-Castelle and Florens-Zmirou approximations of the likelihood.

The parameter theta is supposed to be the value of the true MLE estimator or the minimum contrast estimator of the parameters in the model. If missing or NULL and guess is specified, theta is estimated using the minimum contrast estimator derived from the locally Gaussian approximation of the density. If both theta and guess are missing, nothing can be calculated.

If missing, B is calculated as b/s - 0.5*s.x provided that s.x is not missing.

If missing, B.x is calculated as b.x/s - b*s.x/(s^2)-0.5*s.xx, provided that b.x, s.x, and s.xx are not missing.

If missing, both H and S are evaluated numerically.

Value

x	the value of the AIC statistics

Author(s)

Stefano Maria Iacus

References

Dacunha-Castelle, D., Florens-Zmirou, D. (1986) Estimation of the coefficients of a diffusion from discrete observations, *Stochastics*, 19, 263-284.

Uchida, M., Yoshida, N. (2005) AIC for ergodic diffusion processes from discrete observations, preprint MHF 2005-12, march 2005, *Faculty of Mathematics, Kyushu University, Fukuoka, Japan*.

Examples

```
## Not run:
set.seed(123)
# true model generating data
dri <- expression(-(x-10))
dif <- expression(2*sqrt(x))
sde.sim(X0=10,drift=dri, sigma=dif,N=1000,delta=0.1) -> X

# we test the true model against two competing models
b <- function(x,theta) -theta[1]*(x-theta[2])
b.x <- function(x,theta)  -theta[1]+0*x

s <- function(x,theta) theta[3]*sqrt(x)
s.x <- function(x,theta) theta[3]/(2*sqrt(x))
```

```
s.xx <- function(x,theta) -theta[3]/(4*x^1.5)
# AIC for the true model
sdeAIC(X, NULL, b, s, b.x, s.x, s.xx, guess=c(1,1,1),
       lower=rep(1e-3,3), method="L-BFGS-B")

s <- function(x,theta) sqrt(theta[3]*+theta[4]*x)
s.x <- function(x,theta) theta[4]/(2*sqrt(theta[3]+theta[4]*x))
s.xx <- function(x,theta) -theta[4]^2/(4*(theta[3]+theta[4]*x)^1.5)
# AIC for competing model 1
sdeAIC(X, NULL, b, s, b.x, s.x, s.xx, guess=c(1,1,1,1),
       lower=rep(1e-3,4), method="L-BFGS-B")

s <- function(x,theta) (theta[3]+theta[4]*x)^theta[5]
s.x <- function(x,theta)
          theta[4]*theta[5]*(theta[3]+theta[4]*x)^(-1+theta[5])
s.xx <- function(x,theta) (theta[4]^2*theta[5]*(theta[5]-1)
                *(theta[3]+theta[4]*x)^(-2+theta[5]))
# AIC for competing model 2
sdeAIC(X, NULL, b, s, b.x, s.x, s.xx, guess=c(1,1,1,1,1),
       lower=rep(1e-3,5), method="L-BFGS-B")

## End(Not run)
```

---

sdeDiv   *Phi-Divergences test for diffusion processes*

Description

Phi-Divergences test for diffusion processes.

Usage

```
sdeDiv(X, theta1, theta0, phi= expression( -log(x) ), C.phi, K.phi,
       b, s, b.x, s.x, s.xx, B, B.x, H, S, guess, ...)
```

Arguments

X	a ts object containing a sample path of an sde.
theta1	a vector parameters for the hypothesis H1. If not given, theta1 is estimated from the data.
theta0	a vector parameters for the hypothesis H0.
phi	an expression containing the phi function of the phi-divergence.
C.phi	the value of first derivtive of phi at point 1. If not given, it is calculated within this function.
K.phi	the value of second derivative of phi at point 1. If not given, it is calculated within this function.
b	drift coefficient of the model as a function of x and theta.

s	diffusion coefficient of the model as a function of x and theta.
b.x	partial derivative of b as a function of x and theta.
s.x	partial derivative of s as a function of x and theta.
s.xx	second-order partial derivative of s as a function of x and theta.
B	initial value of the parameters; see details.
B.x	partial derivative of B as a function of x and theta.
H	function of (x,y), the integral of B/s; optional.
S	function of (x,y), the integral of 1/s; optional.
guess	initial value for the parameters to be estimated; optional.
...	passed to the optim function; optional.

Details

The sdeDiv estimate the phi-divergence for diffusion processes defined as D(theta1, theta0) = phi(f(theta1)/f(theta0)) where f is the likelihood function of the process. This function uses the Dacunha-Castelle and Florens-Zmirou approximation of the likelihood for f.

The parameter theta1 is supposed to be the value of the true MLE estimator or the minimum contrast estimator of the parameters in the model. If missing or NULL and guess is specified, theta1 is estimated using the minimum contrast estimator derived from the locally Gaussian approximation of the density. If both theta1 and guess are missing, nothing can be calculated.

The function always calculates the likelihood ratio test and the p-value of the test statistics. In some cases, the p-value of the phi-divergence test statistics is obtained by simulation. In such a case, the out$est.pval is set to TRUE

Dy default phi is set to -log(x). In this case the phi-divergence and the likelihood ratio test are equivalent (e.g. phi-Div = LRT/2)

For more informations on phi-divergences for discretely observed diffusion processes see the references.

If missing, B is calculated as b/s - 0.5*s.x provided that s.x is not missing.

If missing, B.x is calculated as b.x/s - b*s.x/(s^2)-0.5*s.xx, provided that b.x, s.x, and s.xx are not missing.

If missing, both H and S are evaluated numerically.

Value

x	a list containing the value of the divergence, its pvalue, the likelihood ratio test statistics and its p-value

Author(s)

Stefano Maria Iacus

References

Dacunha-Castelle, D., Florens-Zmirou, D. (1986) Estimation of the coefficients of a diffusion from discrete observations, *Stochastics*, 19, 263-284.

De Gregorio, A., Iacus, S.M. (2008) Divergences Test Statistics for Discretely Observed Diffusion Processes, Journal of Statistical Planning and Inference, 140(7), 1744-1753, doi:10.1016/j.jspi.2009.12.029.

Examples

```
## Not run:
set.seed(123)
theta0 <- c(0.89218*0.09045,0.89218,sqrt(0.032742))
theta1 <- c(0.89218*0.09045/2,0.89218,sqrt(0.032742/2))

# we test the true model against two competing models
b <- function(x,theta) theta[1]-theta[2]*x
b.x <- function(x,theta)  -theta[2]

s <- function(x,theta) theta[3]*sqrt(x)
s.x <- function(x,theta) theta[3]/(2*sqrt(x))
s.xx <- function(x,theta) -theta[3]/(4*x^1.5)


X <- sde.sim(X0=rsCIR(1, theta1), N=1000, delta=1e-3, model="CIR",
 theta=theta1)

sdeDiv(X=X, theta0 = theta0,   b=b, s=s, b.x=b.x, s.x=s.x,
 s.xx=s.xx, method="L-BFGS-B",
 lower=rep(1e-3,3), guess=c(1,1,1))

sdeDiv(X=X, theta0 = theta1,   b=b, s=s, b.x=b.x, s.x=s.x,
 s.xx=s.xx, method="L-BFGS-B",
 lower=rep(1e-3,3), guess=c(1,1,1))

lambda <- -1.75
myphi <-  expression( (x^(lambda+1) -x - lambda*(x-1))/(lambda*(lambda+1)) )

sdeDiv(X=X, theta0 = theta0,   phi = myphi, b=b, s=s, b.x=b.x,
 s.x=s.x, s.xx=s.xx, method="L-BFGS-B",
 lower=rep(1e-3,3), guess=c(1,1,1))

sdeDiv(X=X, theta0 = theta1,   phi = myphi, b=b, s=s, b.x=b.x,
 s.x=s.x, s.xx=s.xx, method="L-BFGS-B",
 lower=rep(1e-3,3), guess=c(1,1,1))

## End(Not run)
```

SIMloglik   *Pedersen's approximation of the likelihood*

Description

Pedersen's approximation of the likelihood of a process solution of a stochastic differential equation. This function is useful to calculate approximated maximum likelihood estimators when the transition density of the process is not known. It is computationally intensive.

Usage

```
SIMloglik(X, theta, d, s,  M=10000, N=2, log=TRUE)
```

Arguments

X	a ts object containing a sample path of an sde.
theta	vector of parameters.
d,s	drift and diffusion coefficients; see details.
log	logical; if TRUE, the log-likelihood is returned.
N	number of subintervals; see details.
M	number of Monte Carlo simulations, which should be an even number; see details.

Details

The function SIMloglik returns the simulated log-likelihood obtained by Pedersen's method. The functions s and d are the drift and diffusion coefficients with arguments (t,x,theta).

Value

x	a number

Author(s)

Stefano Maria Iacus

References

Pedersen, A. R. (1995) A new approach to maximum likelihood estimation for stochastic differential equations based on discrete observations, *Scand. J. Statist.*, 22, 55-71.

Examples

```
## Not run:
set.seed(123)
d <- expression(-1*x)
s <- expression(2)
sde.sim(drift=d, sigma=s,N=50,delta=0.01) -> X

S <- function(t, x, theta) sqrt(theta[2])
B <- function(t, x, theta) -theta[1]*x

true.loglik <- function(theta) {
```

```
 DELTA <- deltat(X)
 lik <- 0
 for(i in 2:length(X))
  lik <- lik + dnorm(X[i], mean=X[i-1]*exp(-theta[1]*DELTA),
   sd = sqrt((1-exp(-2*theta[1]*DELTA))*
             theta[2]/(2*theta[1])),TRUE)
 lik
}

xx <- seq(-10,10,length=20)
sapply(xx, function(x) true.loglik(c(x,4))) -> py
sapply(xx, function(x) EULERloglik(X,c(x,4),B,S)) -> pz
sapply(xx, function(x) SIMloglik(X,c(x,4),B,S,M=10000,N=5)) -> pw

plot(xx,py,type="l",xlab=expression(beta),
    ylab="log-likelihood",ylim=c(0,15)) # true
lines(xx,pz, lty=2) # Euler
lines(xx,pw, lty=3) # Simulated

## End(Not run)
```

---

simple.ef   *Simple estimating functions of types I and II*

Description

Apply a simple estimating function to find estimates of the parameters of a process solution of a
stochastic differential equation.

Usage

```
simple.ef(X, f, guess, lower, upper)
```

Arguments

X	a ts object containing a sample path of an sde.
f	a list of expressions of x and/or y and the parameters to be estimated; see details.
guess	initial value of the parameters; see details.
lower	lower bounds for the parameters; see details.
upper	upper bounds for the parameters; see details.

Details

The function simple.ef minimizes a simple estimating function of the form sum_i f_i(x,y;theta)
= 0 or sum_i f_i(x;theta) as a function of theta. The index i varies in 1:length(theta).

The list f is a list of expressions in x or (x,y).

Value

x   a vector of estimates

Author(s)

Stefano Maria Iacus

References

Kessler, M. (1997) Estimation of an ergodic diffusion from discrete observations, *Scand. J. Statist.*, 24, 211-229.

Kessler, M. (2000) Simple and Explicit Estimating Functions for a Discretely Observed Diffusion Process, *Scand. J. Statist.*, 27, 65-82.

Examples

```
set.seed(123);
# Kessler's estimator for O-H process
K.est <- function(x) {
  n.obs <- length(x)
  n.obs/(2*(sum(x^2)))
}

# Least squares estimators for the O-H process
LS.est <- function(x) {
  n <- length(x) -1
  k.sum <- sum(x[1:n]*x[2:(n+1)])
  dt <- deltat(x)
  ifelse(k.sum>0, -log(k.sum/sum(x[1:n]^2))/dt, NA)
}

d <- expression(-1 * x)
s <- expression(1)
x0 <- rnorm(1,sd=sqrt(1/2))
sde.sim(X0=x0,drift=d, sigma=s,N=2500,delta=0.1) -> X

# Kessler's estimator as estimating function
f <- list(expression(2*theta*x^2-1))
simple.ef(X, f, lower=0, upper=Inf)
K.est(X)

# Least Squares estimator as estimating function
f <- list(expression(x*(y-x*exp(-0.1*theta))))
simple.ef(X, f, lower=0, upper=Inf)
LS.est(X)
```

| simple.ef2 | *Simple estimating function based on the infinitesimal generator a the diffusion process* |

Description

Apply a simple estimating function based on the infinitesimal generator of a diffusion to find estimates of the parameters of a process solution of that particular stochastic differential equation.

Usage

```
simple.ef2(X, drift, sigma, h, h.x, h.xx, guess, lower,
           upper)
```

Arguments

X	a `ts` object containing a sample path of an sde.
drift	an expression for the drift coefficient; see details.
sigma	an expression for the diffusion coefficient; see details.
h	an expression of x and the parameters to be estimated; see details.
h.x	an expression of x containing the first derivative of h; see details.
h.xx	an expression of x containing the second derivative of h; see details.
guess	initial value of the parameters; see details.
lower	lower bounds for the parameters; see details.
upper	upper bounds for the parameters; see details.

Details

The function `simple.ef2` minimizes the simple estimating function of the form `sum_i f_i(x;theta) = 0`, where `f` is the result of applying the infinitesimal generator of the diffusion to the function `h`. This involves the drift and diffusion coefficients plus the first two derivatives of `h`. If not provided by the user, the derivatives are calculated by the function.

Value

x	a vector of estimates

Author(s)

Stefano Maria Iacus

References

Kessler, M. (1997) Estimation of an ergodic diffusion from discrete observations, *Scand. J. Statist.*, 24, 211-229.

Kessler, M. (2000) Simple and Explicit Estimating Functions for a Discretely Observed Diffusion Process, *Scand. J. Statist.*, 27, 65-82.

Examples

```
set.seed(123)
d <- expression(10 - x)
s <- expression(sqrt(x))
x0 <- 10
sde.sim(X0=x0,drift=d, sigma=s,N=1500,delta=0.1) -> X

# rather difficult problem unless a good initial guess is given
d <- expression(alpha + theta*x)
s <- expression(x^gamma)
h <- list(expression(x), expression(x^2), expression(x^2))
simple.ef2(X, d, s, h, lower=c(0,-Inf,0), upper=c(Inf,0,1))
```

Index