

Package ‘Sim.DiffProc’

October 8, 2016

Type Package

Version 3.4

Date 2016-10-08

Title Simulation of Diffusion Processes

Author Arsalane Chouaib Guidoum [cre, aut],
Kamal Boukhetala [aut]

Maintainer Arsalane Chouaib Guidoum <acguidoum@usthb.dz>

Depends R (>= 2.15.1)

Imports scatterplot3d, rgl

Suggests knitr, rmarkdown, MASS, ggplot2, sm

Encoding UTF-8

VignetteBuilder knitr

Description Provides the functions for simulating and modeling of stochastic differential equations (SDE's). Statistical analysis and simulation of the solution of SDE's enabled many searchers in different domains to use these equations to modeling practical problems, in financial and actuarial modeling and other areas of application. For example, modeling and simulate of dispersion in shallow water using the attractive center (Boukhetala K, 1996).

License GPL (>= 2)

Classification/MSC 37H10, 37M10, 60H05, 60H10, 60H35, 60J60, 68N15

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-10-08 08:48:04

R topics documented:

Sim.DiffProc-package	2
bconfint	11
BM	12
bridgesde1d	15
bridgesde2d	17

bridgesde3d	20
fitsde	23
fptsde1d	28
fptsde2d	30
fptsde3d	33
HWV	35
Irates	37
plot2d	38
rsde1d	40
snsde1d	42
snsde2d	45
snsde3d	49
st.int	53

Index	57
--------------	-----------

Sim.DiffProc-package *Simulation of Diffusion Processes*

Description

Provides the functions for simulating and modeling of stochastic differential equations (SDE's). Statistical analysis and simulation of the solution of SDE's enabled many searchers in different domains to use these equations to modeling practical problems, in financial and actuarial modeling and other areas of application. For example, modeling and simulate of dispersion in shallow water using the attractive center (Boukhetala K, 1996).

Details

Package:	Sim.DiffProc
Type:	Package
Version:	3.4
Date:	2016-10-08
License:	GPL (>= 3)
Depends:	R (>= 2.15.1)
Imports:	scatterplot3d, rgl
Suggests:	knitr, rmarkdown, MASS, ggplot2, sm
Classification/MSR:	37H10, 37M10, 60H05, 60H10, 60H35, 60J60, 68N15

There are main types of functions in this package:

1. Computing the stochastic integrals of Ito or Stratonovich type.
2. Simulation of solutions to 1,2 and 3-dim stochastic differential equations of Ito or Stratonovich type, with different methods.
3. Estimate drift and diffusion parameters by the method of maximum pseudo-likelihood of the 1-dim stochastic differential equation.

4. Simulation of solutions to 1,2 and 3-dim diffusion bridge of Ito or Stratonovich type, with different methods.
5. Random number generators (RN's) to generate 1,2 and 3-dim sde of Ito or Stratonovich type.
6. First-passage-time (f.p.t) in 1,2 and 3-dim sde of Ito or Stratonovich type.
7. Displaying an object inheriting from class "sde" (1,2 and 3 dim).

Main Features

stochastic integrals:

We consider a simple example to simulation Ito integral, used `st.int` function:

$$\int_{t_0}^t W_s^n dW_s = \frac{1}{n+1} [W_t^{n+1} - W_{t_0}^{n+1}] - \frac{n}{2} \int_{t_0}^t W_s^{n-1} ds$$

And the Stratonovich integral

$$\int_{t_0}^t W_s^n \circ dW_s = \frac{1}{n+1} [W_t^{n+1} - W_{t_0}^{n+1}]$$

```
R> fexpr <- expression( w^2 )
R> ito <- st.int(fexpr,type="ito",M=1,lower=0,upper=1)
R> ito
Ito integral:
  | X(t)  = integral (f(s,w) * dw(s))
  | f(t,w) = w^2
Summary:
  | Number of subintervals      = 1000.
  | Number of simulations       = 1.
  | Limits of integration       = [0,1].
  | Discretization              = 0.001.
R> str <- st.int(fexpr,type="str",M=1,lower=0,upper=1)
R> str
Stratonovich integral:
  | X(t)  = integral (f(s,w) o dw(s))
  | f(t,w) = w^2
Summary:
  | Number of subintervals      = 1000.
  | Number of simulations       = 1.
  | Limits of integration       = [0,1].
  | Discretization              = 0.001.
```

SDE's 1,2 and 3-dim:

There are thus two widely used types of stochastic calculus, Stratonovich and Ito, differing in respect of the stochastic integral used. Modelling issues typically dictate which version is appropriate, but once one has been chosen a corresponding equation of the other type with the same solutions

can be determined. Thus it is possible to switch between the two stochastic calculi. Specifically, the processes $\{X_t, t \geq 0\}$ solution to the Ito SDE:

$$dX_t = f(t, X_t)dt + g(t, X_t)dW_t$$

where $\{W_t, t \geq 0\}$ is the standard Wiener process or standard Brownian motion, the drift $f(t, X_t)$ and diffusion $g(t, X_t)$ are known functions that are assumed to be sufficiently regular (Lipschitz, bounded growth) for existence and uniqueness of solution; has the same solutions as the Stratonovich SDE:

$$dX_t = \underline{f}(t, X_t)dt + g(t, X_t) \circ dW_t$$

with the modified drift coefficient

$$\underline{f}(t, X_t) = f(t, X_t) - \frac{1}{2}g(t, X_t)\frac{\partial g}{\partial x}(t, X_t)$$

The following examples for different methods of simulation of SDEs (1,2 and 3-dim) use the [snssde1d](#), [snssde2d](#) and [snssde3d](#) functions.

```
R> ## 1-dim sde
R> f <- expression(2*(3-x) )
R> g <- expression(2*x)
R> res1 <- snssde1d(drift=f,diffusion=g,M=10,x0=1,N=1000)
R> res1
Ito Sde 1D:
| dX(t) = 2 * (3 - X(t)) * dt + 2 * X(t) * dW(t)
Method:
| Euler scheme of order 0.5
Summary:
| Size of process      | N = 1000.
| Number of simulation | M = 10.
| Initial value        | x0 = 1.
| Time of process      | t in [0,1].
| Discretization       | Dt = 0.001.
R> res2 <- snssde1d(drift=f,diffusion=g,M=10,x0=1,N=1000,type="str")
R> res2
Stratonovich Sde 1D:
| dX(t) = 2 * (3 - X(t)) * dt + 2 * X(t) o dW(t)
Method:
| Euler scheme of order 0.5
Summary:
| Size of process      | N = 1000.
| Number of simulation | M = 10.
| Initial value        | x0 = 1.
| Time of process      | t in [0,1].
| Discretization       | Dt = 0.001.

R> ## 2-dim sde
R> fx <- expression(x-y)
R> gx <- expression(2*y)
R> fy <- expression(y-x)
```

```

R> gy <- expression(2*x)
R> res2d <- snssde2d(driftx=fx,diffx=gx,drifty=fy,diffy=gy,M=1,N=1000,x0=1,y0=1)
R> res2d
Ito Sde 2D:
  | dX(t) = X(t) - Y(t) * dt + 2 * Y(t) * dW1(t)
  | dY(t) = Y(t) - X(t) * dt + 2 * X(t) * dW2(t)
Method:
  | Euler scheme of order 0.5
Summary:
  | Size of process          | N = 1000.
  | Number of simulation     | M = 1.
  | Initial values          | (x0,y0) = (1,1).
  | Time of process         | t in [0,1].
  | Discretization          | Dt = 0.001.
R> plot2d(res2d)

R> ## 3-dim sde
R> fx <- expression(y)
R> gx <- expression(z)
R> fy <- expression(0)
R> gy <- expression(1)
R> fz <- expression(0)
R> gz <- expression(1)
R> res3d <- snssde3d(driftx=fx,diffx=gx,drifty=fy,diffy=gy,driftz=fz,
+                   diffz=gz,N=1000,M=100)
R> res3d
Ito Sde 3D:
  | dX(t) = Y(t) * dt + Z(t) * dW1(t)
  | dY(t) = 0 * dt + 1 * dW2(t)
  | dZ(t) = 0 * dt + 1 * dW3(t)
Method:
  | Euler scheme of order 0.5
Summary:
  | Size of process          | N = 1000.
  | Number of simulation     | M = 100.
  | Initial values          | (x0,y0,z0) = (0,0,0).
  | Time of process         | t in [0,1].
  | Discretization          | Dt = 0.001.
plot3D(res3d)

```

Estimate the parameters of 1-dim sde:

Consider a process solution of the general stochastic differential equation:

$$dX_t = f(t, X_t, \theta)dt + g(t, X_t, \theta)dW_t$$

The package Sim.DiffProc implements the function `fitsde` of estimate drift and diffusion parameters $\theta = (\theta_1, \theta_2, \dots, \theta_p)$ with different methods of maximum pseudo-likelihood of the 1-dim stochastic differential equation.

An example we use a real data, fit with the CKLS model:

$$dX_t = (\theta_1 + \theta_2 X_t)dt + \theta_3 X_t^{\theta_4} dW_t$$

we estimate the vector of parameters $\underline{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4)$, using Euler pseudo-likelihood.

```
R> ## 1-dim fitsde
R> data(Irates)
R> rates <- Irates[,"r1"]
R> rates <- window(rates, start=1964.471, end=1989.333)
R> fx <- expression(theta[1]+theta[2]*x)
R> gx <- expression(theta[3]*x^theta[4])
R> ## theta = (theta1,theta2,theta3,theta4), p=4
R> fitmod <- fitsde(rates,drift=fx,diffusion=gx,pmle="euler",start = list(theta1=1,
      theta2=1,theta3=1,theta4=1),optim.method = "L-BFGS-B")

R> fitmod
Call:
fitsde(data = rates, drift = fx, diffusion = gx, pmle = "euler",
      start = list(theta1 = 1, theta2 = 1, theta3 = 1, theta4 = 1),
      optim.method = "L-BFGS-B")
Coefficients:
      theta1      theta2      theta3      theta4
 2.0769516 -0.2631871  0.1302158  1.4513173
R> summary(fitmod)
Pseudo maximum likelihood estimation
Method: Euler
Call:
fitsde(data = rates, drift = fx, diffusion = gx, pmle = "euler",
      start = list(theta1 = 1, theta2 = 1, theta3 = 1, theta4 = 1),
      optim.method = "L-BFGS-B")
Coefficients:
      Estimate Std. Error
theta1  2.0769516  0.98838467
theta2 -0.2631871  0.19544290
theta3  0.1302158  0.02523105
theta4  1.4513173  0.10323740

-2 log L: 475.7572
R> coef(fitmod)
      theta1      theta2      theta3      theta4
2.0769516 -0.2631871  0.1302158  1.4513173
R> logLik(fitmod)
[1] -237.8786
R> AIC(fitmod)
[1] 483.7572
R> BIC(fitmod)
[1] 487.1514
R> vcov(fitmod)
      theta1      theta2      theta3      theta4
```

```

theta1 0.9769042534 -1.843596e-01 -2.714334e-04 0.0011374342
theta2 -0.1843595796 3.819793e-02 5.169849e-05 -0.0002165286
theta3 -0.0002714334 5.169849e-05 6.366061e-04 -0.0025457493
theta4 0.0011374342 -2.165286e-04 -2.545749e-03 0.0106579616
R> confint(fitmod,level=0.95)
      2.5 %    97.5 %
theta1 0.13975321 4.0141499
theta2 -0.64624812 0.1198740
theta3 0.08076388 0.1796678
theta4 1.24897569 1.6536589

```

Random number generators (RN's) to generate 1,2 and 3-dim sde:

Simulation M-sample for the random variable X_{at} at time $t = at$ by a simulated 1, 2 and 3-dim sde, using the functions `rsde1d`, `rsde2d` and `rsde3d`.

```

R> f <- expression(-2*(x<=0)+2*(x>=0))
R> g <- expression(0.5)
R> res1 <- snssde1d(drift=f,diffusion=g,M=50,type="str",T=10)
R> x <- rsde1d(res1,at=10)
R> x
 [1] -17.64115  21.67111 -19.00162 -20.21546  20.65829  19.59535
 [7] -20.00676 -18.75649 -19.04453 -15.55535 -18.75077  18.89528
[13] -22.99474 -19.66526 -19.75898  22.02310 -19.68301 -19.08581
[19] -19.15081 -19.24476 -22.24332  17.74989  19.88449 -18.17091
[25] -18.65697  19.08473 -17.81218  19.58453  19.27531 -21.88292
[31]  19.03283 -19.29196  21.99163  20.12123  21.09657 -20.20252
[37]  20.85097 -19.41987 -18.67530 -19.36289  19.50057  16.30538
[43]  19.34247 -17.97358  22.81003 -18.40051 -18.47490 -21.86839
[49] -21.32638 -18.96264
R> summary(x)
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-22.990 -19.350 -18.440  -4.436  19.330  22.810

R> ## 2 and 3-dim rsde
R> example(rsde2d)
R> example(rsde3d)

```

First-passage-time (f.p.t) in 1,2 and 3-dim sde

The functions `fptsde1d` (`fptsde2d` and `fptsde3d` for 2 and 3-dim) returns a random variable $\tau_{(X(t),S(t))}$ "first passage time", is defined as:

$$\tau_{(X(t),S(t))} = \{t \geq 0; X_t \geq S(t)\}, \quad \text{if } X(t_0) < S(t_0)$$

$$\tau_{(X(t),S(t))} = \{t \geq 0; X_t \leq S(t)\}, \quad \text{if } X(t_0) > S(t_0)$$

with $S(t)$ is through a continuous boundary (barrier).

```

R> f <- expression( 0.5*x*t )
R> g <- expression( sqrt(1+x^2) )
R> St <- expression(-0.5*sqrt(t)+exp(t^2))
R> mod <- snssde1d(drift=f,diffusion=g,x0=2,M=40,type="srt")
R> fptmod <- fptsde1d(mod,boundary=St)
R> fptmod
$SDE
Stratonovich Sde 1D:
  | dX(t) = 0.5 * X(t) * t * dt + sqrt(1 + X(t)^2) o dW(t)
Method:
  | Euler scheme of order 0.5
Summary:
  | Size of process          | N = 1000.
  | Number of simulation     | M = 40.
  | Initial value           | x0 = 2.
  | Time of process         | t in [0,1].
  | Discretization          | Dt = 0.001.

$boundary
-0.5 * sqrt(t) + exp(t^2)

$fpt
[1] 0.82242466 0.45358835 0.88674689 0.14106847 0.31739211 0.13865566
[7] 0.10290387 0.51083349 0.15448491 0.07300526 0.18760955 0.34355537
[13] 0.38265480 0.21179391 0.25347172 0.47189179 0.22873932 0.30373205
[19] 0.90971226 0.73081590 0.31632793 0.06987074 0.25727990 0.50231425
[25]          NA 0.04375905          NA          NA 0.65789666          NA
[31] 0.68149376 0.21715404 0.77533196 0.97967078 0.42262475          NA
[37] 0.05659770          NA 0.08594537 0.63574506

attr(,"class")
[1] "fptsde1d"
R> summary(fptmod)

      Monte-Carlo Statistics for the F.P.T of X(t)
      | T(S,X) = inf{t >= 0 : X(t) <= -0.5 * sqrt(t) + exp(t^2)}
      T(S,X)
NA's          6
Mean          0.39197
Variance      0.07718
Median        0.31686
First quartile 0.16277
Third quartile 0.60452
Skewness      0.58208
Kurtosis      2.0585
Moment of order 3 0.01248
Moment of order 4 0.01226
Moment of order 5 0.00424

```



```
Int.conf Inf (95
Int.conf Sup (95
R> ## fpt in 2 and 3-dim sde
R> example(fptsde2d)
R> example(fptsde3d)
```

For other examples see `demo(Sim.DiffProc)`, and for an overview of this package, see `vignette("SDEs")`, `vignette("FitSDE")`.

Requirements

R version \geq 2.15.1

Licence

This package and its documentation are usable under the terms of the "GNU General Public License", a copy of which is distributed with the package.

Author(s)

A.C. Guidoum <acguidoum@usthb.dz> and K. Boukhetala <kboukhetala@usthb.dz> (Dept. Probability and Statistics, USTHB, Algeria).

Please send comments, error reports, etc. to the author via the addresses email.

References

- Argyrakisa, P. and G.H. Weiss (2006). A first-passage time problem for many random walkers. *Physica A*, **363**, 343–347.
- Aytug H., G. J. Koehler (2000). New stopping criterion for genetic algorithms. *European Journal of Operational Research*, **126**, 662–674.
- Boukhetala, K. (1994). Simulation study of a dispersion about an attractive centre. In proceedings of 11th Symposium Computational Statistics, edited by R.Dutter and W.Grossman, Wien, Austria, 128–130.
- Boukhetala, K. (1996). Modelling and simulation of a dispersion pollutant with attractive centre. ed by Computational Mechanics Publications, Southampton, U.K and Computational Mechanics Inc, Boston, USA, 245–252.
- Boukhetala, K. (1998a). Estimation of the first passage time distribution for a simulated diffusion process. *Maghreb Math.Rev*, **7**(1), 1–25.
- Boukhetala, K. (1998b). Kernel density of the exit time in a simulated diffusion. *les Annales Maghrebines De L ingenieur*, **12**, 587–589.
- Ding, M. and G. Rangarajan. (2004). First Passage Time Problem: A Fokker-Planck Approach. *New Directions in Statistical Physics*. ed by L. T. Wille. Springer. 31–46.
- Ait-Sahalia, Y. (1999). Transition densities for interest rate and other nonlinear diffusions. *The Journal of Finance*, **54**, 1361–1395.
- Ait-Sahalia, Y. (2002). Maximum likelihood estimation of discretely sampled diffusions: a closed-form approximation approach. *Econometrica*. **70**, 223–262.

- Roman, R.P., Serrano, J. J., Torres, F. (2008). First-passage-time location function: Application to determine first-passage-time densities in diffusion processes. *Computational Statistics and Data Analysis*, **52**, 4132–4146.
- Roman, R.P., Serrano, J. J., Torres, F. (2012). An R package for an efficient approximation of first-passage-time densities for diffusion processes based on the FPTL function. *Applied Mathematics and Computation*, **218**, 8408–8428.
- Kessler, M. (1997). Estimation of an ergodic diffusion from discrete observations. *Scand. J. Statist.*, **24**, 211-229.
- Gardiner, C. W. (1997). *Handbook of Stochastic Methods*. Springer-Verlag, New York.
- Friedman, A. (1975). *Stochastic differential equations and applications*. Volume 1, ACADEMIC PRESS.
- Henderson, D. and Plaschko, P. (2006). *Stochastic differential equations in science and engineering*. World Scientific.
- Croissant, Y. (2014). Ecdat: Data sets for econometrics. R package version 0.2-5.
- Vasicek, O. (1977). An Equilibrium Characterization of the Term Structure. *Journal of Financial Economics*, **5**, 177–188.
- Allen, E. (2007). *Modeling with Ito stochastic differential equations*. Springer-Verlag.
- Jedrzejewski, F. (2009). *Modeles aleatoires et physique probabiliste*. Springer-Verlag.
- Iacus, S.M. (2008). *Simulation and inference for stochastic differential equations: with R examples*. Springer-Verlag, New York.
- Iacus, S.M. (2014). sde: Simulation and Inference for Stochastic Differential Equations. R package version 2.0.13.
- Brouste, A. et al. (2014). The yuima Project: A Computational Framework for Simulation and Inference of Stochastic Differential Equations. *Journal of Statistical Software*, **57**(4).
- Kloeden, P.E, and Platen, E. (1989). A survey of numerical methods for stochastic differential equations. *Stochastic Hydrology and Hydraulics*, **3**, 155–178.
- Kloeden, P.E, and Platen, E. (1991a). Relations between multiple ito and stratonovich integrals. *Stochastic Analysis and Applications*, **9**(3), 311–321.
- Kloeden, P.E, and Platen, E. (1991b). Stratonovich and ito stochastic taylor expansions. *Mathematische Nachrichten*, **151**, 33–50.
- Kloeden, P.E, and Platen, E. (1995). *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, New York.
- Oksendal, B. (2000). *Stochastic Differential Equations: An Introduction with Applications*. 5th edn. Springer-Verlag, Berlin.
- B.L.S. Prakasa Rao. (1999). *Statistical Inference for Diffusion Type Processes*. Arnold, London and Oxford University press, New York.
- Kutoyants, Y.A. (2004). *Statistical Inference for Ergodic Diffusion Processes*. Springer, London.
- Sorensen, H. (2000). Inference for Diffusion Processes and Stochastic Volatility Models. Ph.D. thesis, Department of Mathematical Sciences, University of Copenhagen.
- Sorensen, H. (2002). Estimation of diffusion parameters for discretely observed diffusion processes. *Bernoulli*, **8**, 491–508.

- Sorensen, H. (2004). Parametric inference for diffusion processes observed at discrete points in time: a survey. *International Statistical Review*, 72, 337–354.
- Platen, E. (1980). Weak convergence of approximations of ito integral equations. *Z Angew Math Mech.* **60**, 609–614.
- Platen, E. and Bruti-Liberati, N. (2010). *Numerical Solution of Stochastic Differential Equations with Jumps in Finance*. Springer-Verlag, New York.
- Saito, Y, and Mitsui, T. (1993). Simulation of Stochastic Differential Equations. *The Annals of the Institute of Statistical Mathematics*, **3**, 419–432.
- Risken, H. (2001). *The Fokker Planck Equation : Methods of Solutions and Applications*. 2nd edition, Springer Series in Synergetics.
- Dacunha, D.C. and Florens, D.Z. (1986). Estimation of the Coefficients of a Diffusion from Discrete Observations. *Stochastics*. 19, 263–284.
- Dohnal, G. (1987). On estimating the diffusion coefficient. *J. Appl.Prob.*, 24, 105–114.
- Genon, V.C. (1990). Maximum constrast estimation for diffusion processes from discrete observation. *Statistics*, 21, 99–116.
- Protter, P. (2005). *Stochastic Integration and Differential Equations*. 2nd edn. Springer-Verlag, New York.
- Bladt, M. and Sorensen, M. (2007). Simple simulation of diffusion bridges with application to likelihood inference for diffusions. *Working Paper, University of Copenhagen*.
- Ozaki, T. (1992). A bridge between nonlinear time series models and nonlinear stochastic dynamical systems: A local linearization approach. *Statistica Sinica*, 2, 25-83.
- Shoji, L., Ozaki, T. (1998). Estimation for nonlinear stochastic differential equations by a local linearization method. *Stochastic Analysis and Applications*, 16, 733-752.
- Nicolau, J. (2004). Introduction to the estimation of stochastic differential equations based on discrete observations. *Autumn School and International Conference, Stochastic Finance*.
- F C Klebaner, F.C. (2005). *Introduction to stochastic calculus with application*. 2nd edn. Imperial College Press (ICP).
- Henderson, D. and Plaschko, P. (2006). *Stochastic differential equations in science and engineering*. World Scientific.

See Also

sde, yumia, fptdApprox, PSM.

bconfint

Kurtosis, Skewness, Moment and Confidence Bands

Description

Generic function for compute the kurtosis, skewness, moment and confidence bands of class "sde".

Usage

```
## Default S3 method:
bconfint(x, level = 0.95, ...)
## Default S3 method:
kurtosis(x, ...)
## Default S3 method:
moment(x, order = 2, ...)
## Default S3 method:
skewness(x, ...)
```

Arguments

x	an object inheriting from class "sde".
order	order of moment.
level	the confidence level required.
...	further arguments for (non-default) methods.

Author(s)

A.C. Guidoum, K. Boukhetala.

Examples

```
## Example 1:
## dX(t) = 2*(3-X(t)) *dt + dW(t)
set.seed(1234)

f <- expression( 2*(3-x) )
g <- expression( 1 )
res <- snssde1d(drift=f,diffusion=g,M=25)
x <- rsde1d(res,at=1)
kurtosis(res)
skewness(res)
bconfint(res,level = 0.95)
moment(res,order=4)
```

 BM

Brownian motion, Brownian bridge, geometric Brownian motion, and arithmetic Brownian motion simulators

Description

The (S3) generic function for simulation of brownian motion, brownian bridge, geometric brownian motion, and arithmetic brownian motion.

Usage

```

BM(N, ...)
BB(N, ...)
GBM(N, ...)
ABM(N, ...)

## Default S3 method:
BM(N =100,M=1,x0=0,t0=0,T=1,Dt, ...)
## Default S3 method:
BB(N =100,M=1,x0=0,y=1,t0=0,T=1,Dt, ...)
## Default S3 method:
GBM(N =100,M=1,x0=1,t0=0,T=1,Dt,theta=1,sigma=1, ...)
## Default S3 method:
ABM(N =100,M=1,x0=0,t0=0,T=1,Dt,theta=1,sigma=1, ...)

```

Arguments

N	number of simulation steps.
M	number of trajectories.
x0	initial value of the process at time t_0 .
y	terminal value of the process at time T of the BB.
t0	initial time.
T	final time.
Dt	time step of the simulation (discretization). If it is <code>missing</code> a default $\Delta t = \frac{T-t_0}{N}$.
theta	the interest rate of the ABM and GBM.
sigma	the volatility of the ABM and GBM.
...	further arguments for (non-default) methods.

Details

The function BM returns a trajectory of the **standard Brownian motion** (Wiener process) in the time interval $[t_0, T]$. Indeed, for $W(dt)$ it holds true that $W(dt) \rightarrow W(dt) - W(0) \rightarrow \mathcal{N}(0, dt)$, where $\mathcal{N}(0, 1)$ is normal distribution [Normal](#).

The function BB returns a trajectory of the **Brownian bridge** starting at x_0 at time t_0 and ending at y at time T ; i.e., the diffusion process solution of stochastic differential equation:

$$dX_t = \frac{y - X_t}{T - t} dt + dW_t$$

The function GBM returns a trajectory of the **geometric Brownian motion** starting at x_0 at time t_0 ; i.e., the diffusion process solution of stochastic differential equation:

$$dX_t = \theta X_t dt + \sigma X_t dW_t$$

The function ABM returns a trajectory of the **arithmetic Brownian motion** starting at x_0 at time t_0 ; i.e., the diffusion process solution of stochastic differential equation:

$$dX_t = \theta dt + \sigma dW_t$$

Value

X an visible ts object.

Author(s)

A.C. Guidoum, K. Boukhetala.

References

Allen, E. (2007). *Modeling with Ito stochastic differential equations*. Springer-Verlag, New York.
 Jedrzejewski, F. (2009). *Modeles aleatoires et physique probabiliste*. Springer-Verlag, New York.
 Henderson, D and Plaschko, P. (2006). *Stochastic differential equations in science and engineering*. World Scientific.

See Also

This functions [BM](#), [BBridge](#) and [GBM](#) are available in other packages such as "sde".

Examples

```
op <- par(mfrow = c(2, 2))

## Brownian motion
set.seed(1234)
X <- BM(N = 1000, M = 50)
plot(X,plot.type="single")
lines(as.vector(time(X)),rowMeans(X),col="red")

## Brownian bridge
set.seed(1234)
X <- BB(N = 1000, M =50)
plot(X,plot.type="single")
lines(as.vector(time(X)),rowMeans(X),col="red")

## Geometric Brownian motion
set.seed(1234)
X <- GBM(N = 1000, M = 50)
plot(X,plot.type="single")
lines(as.vector(time(X)),rowMeans(X),col="red")

## Arithmetic Brownian motion
set.seed(1234)
X <- ABM(N = 1000, M = 50)
plot(X,plot.type="single")
lines(as.vector(time(X)),rowMeans(X),col="red")

par(op)
```

Description

The (S3) generic function `bridgesde1d` for simulation of 1-dim diffusion bridge.

Usage

```
bridgesde1d(N, ...)
## Default S3 method:
bridgesde1d(N = 1000, M=1, x0 = 0, y = 0, t0 = 0, T = 1, Dt,
  drift, diffusion, alpha = 0.5, mu = 0.5, type = c("ito", "str"),
  method = c("euler", "milstein", "predcorr", "smilstein", "taylor",
    "heun", "rk1", "rk2", "rk3"), ...)

## S3 method for class 'bridgesde1d'
time(x, ...)
## S3 method for class 'bridgesde1d'
mean(x, ...)
## S3 method for class 'bridgesde1d'
median(x, ...)
## S3 method for class 'bridgesde1d'
quantile(x, ...)
## S3 method for class 'bridgesde1d'
kurtosis(x, ...)
## S3 method for class 'bridgesde1d'
skewness(x, ...)
## S3 method for class 'bridgesde1d'
moment(x, order = 2, ...)
## S3 method for class 'bridgesde1d'
bconfint(x, level=0.95, ...)
## S3 method for class 'bridgesde1d'
plot(x, ...)
## S3 method for class 'bridgesde1d'
lines(x, ...)
## S3 method for class 'bridgesde1d'
points(x, ...)
```

Arguments

<code>N</code>	number of simulation steps.
<code>M</code>	number of trajectories.
<code>x0</code>	initial value of the process at time <code>t0</code> .
<code>y</code>	terminal value of the process at time <code>T</code> .

t0	initial time.
T	final time.
Dt	time step of the simulation (discretization). If it is <code>missing</code> a default $\Delta t = \frac{T-t_0}{N}$.
drift	drift coefficient: an <code>expression</code> of two variables t and x.
diffusion	diffusion coefficient: an <code>expression</code> of two variables t and x.
alpha, mu	weight of the predictor-corrector scheme; the default alpha = 0.5 and mu = 0.5.
type	if type="ito" simulation diffusion bridge of Ito type, else type="str" simulation diffusion bridge of Stratonovich type; the default type="ito".
method	numerical methods of simulation, the default method = "euler"; see <code>snsde1d</code> .
x	an object inheriting from class "bridgesde1d".
order	order of moment.
level	the confidence level required.
...	further arguments for (non-default) methods.

Details

The function `bridgesde1d` returns a trajectory of the diffusion bridge starting at x at time t0 and ending at y at time T.

The methods of approximation are classified according to their different properties. Mainly two criteria of optimality are used in the literature: the strong and the weak (orders of) convergence. The method of simulation can be one among: Euler-Maruyama Order 0.5, Milstein Order 1, Milstein Second-Order, Predictor-Corrector method, Ito-Taylor Order 1.5, Heun Order 2 and Runge-Kutta Order 1, 2 and 3.

For more details see `vignette("SDEs")`.

Value

`bridgesde1d` returns an object inheriting from `class` "bridgesde1d".

X	an invisible <code>ts</code> object.
drift	drift coefficient.
diffusion	diffusion coefficient.
C	nombre of crossing realized.
type	type of sde.
method	the numerical method used.

Author(s)

A.C. Guidoum, K. Boukhetala.

References

Bladt, M. and Sorensen, M. (2007). Simple simulation of diffusion bridges with application to likelihood inference for diffusions. *Working Paper, University of Copenhagen*.

Iacus, S.M. (2008). *Simulation and inference for stochastic differential equations: with R examples*. Springer-Verlag, New York

See Also

[bridgesde2d](#) and [bridgesde3d](#) for 2 and 3-dim.

[DBridge](#) in package "sde".

Examples

```
## Ito Bridge sde
## dX(t) = 2*(1-X(t)) *dt + dW(t)
## x0 = 0 at time t0=0 , and y = 1 at time T=1
set.seed(1234)

f <- expression( 2*(1-x) )
g <- expression( 1 )
X <- bridgesde1d(drift=f,diffusion=g,y=1,N=1000,M=100)
X
plot(X,plot.type="single")
lines(time(X),mean(X),col=2)
```

bridgesde2d

Simulation of 2-Dim Diffusion Bridge

Description

The (S3) generic function `bridgesde2d` for simulation of 2-dim diffusion bridge.

Usage

```
bridgesde2d(N, ...)
## Default S3 method:
bridgesde2d(N = 1000, M = 1, x0 = c(0, 0), y = c(1, 1), t0 = 0, T = 1, Dt,
  driftx, diffx, drifty, diffy, alpha = 0.5, mu = 0.5, type = c("ito", "str"),
  method = c("euler", "milstein", "predcorr", "smilstein", "taylor",
    "heun", "rk1", "rk2", "rk3"), ...)
```

```
## S3 method for class 'bridgesde2d'
time(x, ...)
## S3 method for class 'bridgesde2d'
mean(x, ...)
## S3 method for class 'bridgesde2d'
```

```

median(x, ...)
## S3 method for class 'bridgesde2d'
quantile(x, ...)
## S3 method for class 'bridgesde2d'
kurtosis(x, ...)
## S3 method for class 'bridgesde2d'
skewness(x, ...)
## S3 method for class 'bridgesde2d'
moment(x, order = 2, ...)
## S3 method for class 'bridgesde2d'
bconfint(x, level=0.95, ...)
## S3 method for class 'bridgesde2d'
plot(x, ...)
## S3 method for class 'bridgesde2d'
lines(x, ...)
## S3 method for class 'bridgesde2d'
points(x, ...)
## S3 method for class 'bridgesde2d'
plot2d(x, ...)
## S3 method for class 'bridgesde2d'
lines2d(x, ...)
## S3 method for class 'bridgesde2d'
points2d(x, ...)

```

Arguments

N	number of simulation steps.
M	number of trajectories.
x0	initial value (numeric vector of length 2) of the process X_t and Y_t at time t_0 .
y	terminal value (numeric vector of length 2) of the process X_t and Y_t at time T .
t0	initial time.
T	final time.
Dt	time step of the simulation (discretization). If it is <code>missing</code> a default $\Delta t = \frac{T-t_0}{N}$.
driftx, drifty	drift coefficient: an <code>expression</code> of three variables t, x and y for process X_t and Y_t .
diffx, diffy	diffusion coefficient: an <code>expression</code> of three variables t, x and y for process X_t and Y_t .
alpha, mu	weight of the predictor-corrector scheme; the default alpha = 0.5 and mu = 0.5.
type	if type="ito" simulation diffusion bridge of Ito type, else type="str" simulation diffusion bridge of Stratonovich type; the default type="ito".
method	numerical methods of simulation, the default method = "euler"; see <code>snsde2d</code> .
x	an object inheriting from class "bridgesde2d".
order	order of moment.
level	the confidence level required.
...	further arguments for (non-default) methods.

Details

The function `bridgesde2d` returns a `mts` of the diffusion bridge starting at x at time t_0 and ending at y at time T .

The methods of approximation are classified according to their different properties. Mainly two criteria of optimality are used in the literature: the strong and the weak (orders of) convergence. The method of simulation can be one among: Euler-Maruyama Order 0.5, Milstein Order 1, Milstein Second-Order, Predictor-Corrector method, Ito-Taylor Order 1.5, Heun Order 2 and Runge-Kutta Order 1, 2 and 3.

For more details see `vignette("SDEs")`.

Value

`bridgesde2d` returns an object inheriting from `class "bridgesde2d"`.

<code>X, Y</code>	an invisible <code>mts</code> (2-dim) object $(X(t), Y(t))$.
<code>driftx, drifty</code>	drift coefficient of $X(t)$ and $Y(t)$.
<code>diffx, diffy</code>	diffusion coefficient of $X(t)$ and $Y(t)$.
<code>Cx, Cy</code>	nombre of crossing realized of $X(t)$ ($Y(t)$).
<code>type</code>	type of sde.
<code>method</code>	the numerical method used.

Author(s)

A.C. Guidoum, K. Boukhetala.

References

Bladt, M. and Sorensen, M. (2007). Simple simulation of diffusion bridges with application to likelihood inference for diffusions. *Working Paper, University of Copenhagen*.

Iacus, S.M. (2008). *Simulation and inference for stochastic differential equations: with R examples*. Springer-Verlag, New York

See Also

[bridgesde1d](#) for simulation of 1-dim diffusion bridge.

[DBridge](#) in package "sde".

Examples

```
## dX(t) = 4*(-1-X(t))*Y(t) dt + 0.2 dW1(t)
## dY(t) = 4*(1-Y(t))*X(t) dt + 0.2 dW2(t)
## x01 = 0 , y01 = 1
## x02 = -1, y02 = 0
## W1(t) and W2(t) two independent Brownian motion
set.seed(1234)
```

```

fx <- expression(4*(-1-x)*y)
gx <- expression(0.2)
fy <- expression(4*(1-y)*x)
gy <- expression(0.2)

res <- bridgesde2d(x0=c(0,-1),y=c(1,0),driftx=fx,diffx=gx,drifty=fy,diffy=gy,M=30)
res
plot(res)
dev.new()
plot2d(res,type="n")
points2d(res,col=rgb(0,100,0,50,maxColorValue=255), pch=16)

```

bridgesde3d

Simulation of 3-Dim Diffusion Bridge

Description

The (S3) generic function `bridgesde3d` for simulation of 3-dim diffusion bridge.

Usage

```

bridgesde3d(N, ...)
## Default S3 method:
bridgesde3d(N=1000,M=1, x0=c(0,0, 0), y=c(1,-1, 2),
  t0 = 0, T = 1, Dt, driftx, diffx, drifty, diffy, driftz, diffz,
  alpha = 0.5, mu = 0.5, type = c("ito", "str"), method = c("euler",
  "milstein","predcorr", "smilstein", "taylor", "heun", "rk1", "rk2",
  "rk3"), ...)

## S3 method for class 'bridgesde3d'
time(x, ...)
## S3 method for class 'bridgesde3d'
mean(x, ...)
## S3 method for class 'bridgesde3d'
median(x, ...)
## S3 method for class 'bridgesde3d'
quantile(x, ...)
## S3 method for class 'bridgesde3d'
kurtosis(x, ...)
## S3 method for class 'bridgesde3d'
skewness(x, ...)
## S3 method for class 'bridgesde3d'
moment(x, order = 2, ...)
## S3 method for class 'bridgesde3d'
bconfint(x, level=0.95, ...)
## S3 method for class 'bridgesde3d'
plot(x, ...)
## S3 method for class 'bridgesde3d'

```

```

lines(x, ...)
## S3 method for class 'bridgesde3d'
points(x, ...)
## S3 method for class 'bridgesde3d'
plot3D(x, display = c("persp", "rgl"), ...)

```

Arguments

N	number of simulation steps.
M	number of trajectories.
x0	initial value (numeric vector of length 3) of the process X_t , Y_t and Z_t at time t_0 .
y	terminal value (numeric vector of length 3) of the process X_t , Y_t and Z_t at time T .
t0	initial time.
T	final time.
Dt	time step of the simulation (discretization). If it is missing a default $\Delta t = \frac{T-t_0}{N}$.
driftx	drift coefficient: an expression of four variables t, x, y and z for process X_t .
diffx	diffusion coefficient: an expression of four variables t, x, y and z for process X_t .
drifty	drift coefficient: an expression of four variables t, x, y and z for process Y_t .
diffy	diffusion coefficient: an expression of four variables t, x, y and z for process Y_t .
driftz	drift coefficient: an expression of four variables t, x, y and z for process Z_t .
diffz	diffusion coefficient: an expression of four variables t, x, y and z for process Z_t .
alpha	weight alpha of the predictor-corrector scheme; the default alpha = 0.5.
mu	weight mu of the predictor-corrector scheme; the default mu = 0.5.
type	if type="ito" simulation diffusion bridge of Ito type, else type="str" simulation diffusion bridge of Stratonovich type; the default type="ito".
method	numerical methods of simulation, the default method = "euler"; see snsde3d .
x	an object inheriting from class "bridgesde3d".
order	order of moment.
level	the confidence level required.
display	"persp" perspective or "rgl" plots.
...	further arguments for (non-default) methods.

Details

The function `bridgesde3d` returns a `mts` of the diffusion bridge starting at x at time t_0 and ending at y at time T .

The methods of approximation are classified according to their different properties. Mainly two criteria of optimality are used in the literature: the strong and the weak (orders of) convergence. The method of simulation can be one among: Euler-Maruyama Order 0.5, Milstein Order 1, Milstein Second-Order, Predictor-Corrector method, Ito-Taylor Order 1.5, Heun Order 2 and Runge-Kutta Order 1, 2 and 3.

For more details see `vignette("SDEs")`.

Value

`bridgesde3d` returns an object inheriting from `class "bridgesde3d"`.

<code>X, Y, Z</code>	an invisible <code>mts</code> (3-dim) object $(X(t), Y(t), Z(t))$.
<code>driftx, drifty, driftz</code>	drift coefficient of $X(t)$, $Y(t)$ and $Z(t)$.
<code>diffx, diffy, diffz</code>	diffusion coefficient of $X(t)$, $Y(t)$ and $Z(t)$.
<code>Cx, Cy, Cz</code>	nombre of crossing realized of $X(t)$ ($Y(t)$)($Z(t)$).
<code>type</code>	type of sde.
<code>method</code>	the numerical method used.

Author(s)

A.C. Guidoum, K. Boukhetala.

References

Bladt, M. and Sorensen, M. (2007). Simple simulation of diffusion bridges with application to likelihood inference for diffusions. *Working Paper, University of Copenhagen*.

Iacus, S.M. (2008). *Simulation and inference for stochastic differential equations: with R examples*. Springer-Verlag, New York

See Also

`bridgesde1d` for simulation of 1-dim diffusion bridge. `DBridge` in package "sde".

Examples

```
## dX(t) = 4*(-1-X(t))*Y(t) dt + 0.2 * dW1(t) ; x01 = 0 and y01 = 0
## dY(t) = 4*(1-Y(t)) *X(t) dt + 0.2 * dW2(t) ; x02 = -1 and y02 = -2
## dZ(t) = 4*(1-Z(t)) *Y(t) dt + 0.2 * dW3(t) ; x03 = 0.5 and y03 = 0.5
## W1(t), W2(t) and W3(t) three independent Brownian motion
set.seed(1234)

fx <- expression(4*(-1-x)*y)
gx <- expression(0.2)
```

```

fy <- expression(4*(1-y)*x)
gy <- expression(0.2)
fz <- expression(4*(1-z)*y)
gz <- expression(0.2)

res <- bridgesde3d(x0=c(0,-1,0.5),y=c(0,-2,0.5),driftx=fx,diffx=gx,
                  driftz=fz,diffz=gz,M=10)
res
plot(res,union=TRUE)
dev.new()
plot3D(res,display = "persp",main="3-dim bridge sde")

```

fitsde

Maximum Pseudo-Likelihood Estimation of 1-Dim SDE

Description

The (S3) generic function "fitsde" of estimate drift and diffusion parameters by the method of maximum pseudo-likelihood of the 1-dim stochastic differential equation.

Usage

```

fitsde(data, ...)
## Default S3 method:
fitsde(data, drift, diffusion, start = list(), pmle = c("euler","kessler",
              "ozaki", "shoji"), optim.method = "L-BFGS-B",
        lower = -Inf, upper = Inf, ...)

## S3 method for class 'fitsde'
summary(object, ...)
## S3 method for class 'fitsde'
coef(object, ...)
## S3 method for class 'fitsde'
vcov(object, ...)
## S3 method for class 'fitsde'
logLik(object, ...)
## S3 method for class 'fitsde'
AIC(object, ...)
## S3 method for class 'fitsde'
BIC(object, ...)
## S3 method for class 'fitsde'
confint(object,parm, level=0.95, ...)

```

Arguments

data	a univariate time series (ts class).
drift	drift coefficient: an expression of two variables t, x and theta a vector of parameters of sde. See Examples.

diffusion	diffusion coefficient: an expression of two variables t , x and θ a vector of parameters of <code>sde</code> . See Examples.
start	named list of starting values for optimizer. See Examples.
pmle	a character string specifying the method; can be either: "euler" (Euler pseudo-likelihood), "ozaki" (Ozaki pseudo-likelihood), "shoji" (Shoji pseudo-likelihood), and "kessler" (Kessler pseudo-likelihood).
optim.method	the method for optim .
lower, upper	bounds on the variables for the "Brent" or "L-BFGS-B" method.
object	an object inheriting from class "fitsde".
parm	a specification of which parameters are to be given confidence intervals, either a vector of names (example <code>parm='theta1'</code>). If missing, all parameters are considered.
level	the confidence level required.
...	further arguments to pass to optim .

Details

The function `fitsde` returns a pseudo-likelihood estimators of the drift and diffusion parameters in 1-dim stochastic differential equation. The [optim](#) optimizer is used to find the maximum of the negative log pseudo-likelihood. An approximate covariance matrix for the parameters is obtained by inverting the Hessian matrix at the optimum.

The `pmle` of pseudo-likelihood can be one among: "euler": Euler pseudo-likelihood, "ozaki": Ozaki pseudo-likelihood, "shoji": Shoji pseudo-likelihood, and "kessler": Kessler pseudo-likelihood.

For more details see `vignette("FitSDE")`.

Value

`fitsde` returns an object inheriting from [class](#) "fitsde".

Author(s)

A.C. Guidoum, K. Boukhetala.

References

- Kessler, M. (1997). Estimation of an ergodic diffusion from discrete observations. *Scand. J. Statist.*, 24, 211-229.
- Iacus, S.M. (2008). *Simulation and inference for stochastic differential equations: with R examples*. Springer-Verlag, New York.
- Iacus, S.M. (2009). `sde`: Simulation and Inference for Stochastic Differential Equations. *R package version 2.0.10*.
- Iacus, S.M. and all. (2014). The `yuima` Project: A Computational Framework for Simulation and Inference of Stochastic Differential Equations. *Journal of Statistical Software*, 57(4).

- Ozaki, T. (1992). A bridge between nonlinear time series models and nonlinear stochastic dynamical systems: A local linearization approach. *Statistica Sinica*, 2, 25-83.
- Shoji, L., Ozaki, T. (1998). Estimation for nonlinear stochastic differential equations by a local linearization method. *Stochastic Analysis and Applications*, 16, 733-752.
- Dacunha, D.C. and Florens, D.Z. (1986). Estimation of the Coefficients of a Diffusion from Discrete Observations. *Stochastics*. 19, 263–284.
- Dohnal, G. (1987). On estimating the diffusion coefficient. *J. Appl.Prob.*, 24, 105–114.
- Genon, V.C. (1990). Maximum constrast estimation for diffusion processes from discrete observation. *Statistics*, 21, 99–116.
- Nicolau, J. (2004). Introduction to the estimation of stochastic differential equations based on discrete observations. *Autumn School and International Conference, Stochastic Finance*.
- Ait-Sahalia, Y. (1999). Transition densities for interest rate and other nonlinear diffusions. *The Journal of Finance*, 54, 1361–1395.
- Ait-Sahalia, Y. (2002). Maximum likelihood estimation of discretely sampled diffusions: a closed-form approximation approach. *Econometrica*. 70, 223–262.
- B.L.S. Prakasa Rao. (1999). *Statistical Inference for Diffusion Type Processes*. Arnold, London and Oxford University press, New York.
- Kutoyants, Y.A. (2004). *Statistical Inference for Ergodic Diffusion Processes*. Springer, London.

See Also

[dcEuler](#), [dcElerian](#), [dcOzaki](#), [dcShoji](#), [dcKessler](#) and [dcSim](#) for approximated conditional law of a diffusion process. [gmm](#) estimator of the generalized method of moments by Hansen, and [HPloglik](#) these functions are useful to calculate approximated maximum likelihood estimators when the transition density of the process is not known, in package "sde".

[qmle](#) in package "yuima" calculate quasi-likelihood and ML estimator of least squares estimator.

[PSM.estimate](#) in package "PSM" for estimation of linear and non-linear mixed-effects models using stochastic differential equations.

Examples

```
##### Example 1:

## Modele GBM (BS)
## dX(t) = theta1 * X(t) * dt + theta2 * x * dw(t)
## Simulation of data
set.seed(1234)

X <- GBM(N =1000,theta=4,sigma=1)
## Estimation: true theta=c(4,1)
fx <- expression(theta[1]*x)
gx <- expression(theta[2]*x)

fres <- fitsde(data=X,drift=fx,diffusion=gx,start = list(theta1=1,theta2=1),
              lower=c(0,0))

fres
summary(fres)
```

```

coef(fres)
logLik(fres)
AIC(fres)
BIC(fres)
vcov(fres)
confint(fres,level=0.95)

##### Example 2:

## Nonlinear mean reversion (Ait-Sahalia) modele
##  $dX(t) = (\theta_1 + \theta_2 x + \theta_3 x^2) * dt + \theta_4 * x^{\theta_5} * dW(t)$ 
## Simulation of the process X(t)
set.seed(1234)

f <- expression(1 - 11*x + 2*x^2)
g <- expression(x^0.5)
res <- snssde1d(drift=f,diffusion=g,M=1,N=1000,Dt=0.001,x0=5)
mydata1 <- res$X

## Estimation
## true param theta= c(1,-11,2,1,0.5)
true <- c(1,-11,2,1,0.5)
pmle <- eval(formals(fitsde.default)$pmle)

fx <- expression(theta[1] + theta[2]*x + theta[3]*x^2)
gx <- expression(theta[4]*x^theta[5])

fres <- lapply(1:4, function(i) fitsde(mydata1,drift=fx,diffusion=gx,
                                     pmle=pmle[i],start = list(theta1=1,theta2=1,theta3=1,theta4=1,
                                                                  theta5=1),optim.method = "L-BFGS-B"))
Coef <- data.frame(true,do.call("cbind",lapply(1:4,function(i) coef(fres[[i]]))))
names(Coef) <- c("True",pmle)
Summary <- data.frame(do.call("rbind",lapply(1:4,function(i) logLik(fres[[i]]))),
                     do.call("rbind",lapply(1:4,function(i) AIC(fres[[i]]))),
                     do.call("rbind",lapply(1:4,function(i) BIC(fres[[i]]))),
                     row.names=pmle)
names(Summary) <- c("logLik","AIC","BIC")
Coef
Summary

##### Example 3:

##  $dX(t) = (\theta_1 x * t + \theta_2 * \tan(x)) * dt + \theta_3 * t * dW(t)$ 
## Simulation of data
set.seed(1234)

f <- expression(2*x*t-tan(x))
g <- expression(1.25*t)
sim <- snssde1d(drift=f,diffusion=g,M=1,N=1000,Dt=0.001,x0=10)
mydata2 <- sim$X

```

```

## Estimation
## true param theta= c(2,-1,1.25)
true <- c(2,-1,1.25)

fx <- expression(theta[1]*x*t+theta[2]*tan(x))
gx <- expression(theta[3]*t)

fres <- lapply(1:4, function(i) fitsde(mydata2,drift=fx,diffusion=gx,
  pmle=pmle[i],start = list(theta1=1,theta2=1,theta3=1),
  optim.method = "L-BFGS-B"))
Coef <- data.frame(true,do.call("cbind",lapply(1:4,function(i) coef(fres[[i]]))))
names(Coef) <- c("True",pmle)
Summary <- data.frame(do.call("rbind",lapply(1:4,function(i) logLik(fres[[i]]))),
  do.call("rbind",lapply(1:4,function(i) AIC(fres[[i]]))),
  do.call("rbind",lapply(1:4,function(i) BIC(fres[[i]]))),
  row.names=pmle)
names(Summary) <- c("logLik","AIC","BIC")
Coef
Summary

##### Example 4:

## Application to real data
## CKLS modele vs CIR modele
## CKLS (mod1):  $dX(t) = (\theta_1 + \theta_2 * X(t)) * dt + \theta_3 * X(t)^{\theta_4} * dW(t)$ 
## CIR (mod2):  $dX(t) = (\theta_1 + \theta_2 * X(t)) * dt + \theta_3 * \sqrt{X(t)} * dW(t)$ 
set.seed(1234)

data(Irates)
rates <- Irates[, "r1"]
rates <- window(rates, start=1964.471, end=1989.333)

fx1 <- expression(theta[1]+theta[2]*x)
gx1 <- expression(theta[3]*x^theta[4])
gx2 <- expression(theta[3]*sqrt(x))

fitmod1 <- fitsde(rates,drift=fx1,diffusion=gx1,pmle="euler",start = list(theta1=1,theta2=1,
  theta3=1,theta4=1),optim.method = "L-BFGS-B")
fitmod2 <- fitsde(rates,drift=fx1,diffusion=gx2,pmle="euler",start = list(theta1=1,theta2=1,
  theta3=1),optim.method = "L-BFGS-B")

summary(fitmod1)
summary(fitmod2)
coef(fitmod1)
coef(fitmod2)
confint(fitmod1,parm=c('theta2','theta3'))
confint(fitmod2,parm=c('theta2','theta3'))
AIC(fitmod1)
AIC(fitmod2)

## Display
## CKLS Modele

```

```

op <- par(mfrow = c(1, 2))
theta <- coef(fitmod1)
N <- length(rates)
res <- snssde1d(drift=fx1,diffusion=gx1,M=200,t0=time(rates)[1],T=time(rates)[N],
               Dt=deltat(rates),x0=rates[1],N)
plot(res,plot.type="single",ylim=c(0,40))
lines(rates,col=2,lwd=2)
legend("topleft",c("real data","CKLS modele"),inset = .01,col=c(2,1),lwd=2,cex=0.8)

## CIR Modele
theta <- coef(fitmod2)
res <- snssde1d(drift=fx1,diffusion=gx2,M=200,t0=time(rates)[1],T=time(rates)[N],
               Dt=deltat(rates),x0=rates[1],N)
plot(res,plot.type="single",ylim=c(0,40))
lines(rates,col=2,lwd=2)
legend("topleft",c("real data","CIR modele"),inset = .01,col=c(2,1),lwd=2,cex=0.8)
par(op)

```

fptsde1d

First Passage Time in 1-Dim SDE

Description

The (S3) generic function `fptsde1d` for simulate first-passage-time (f.p.t) in 1-dim stochastic differential equations.

Usage

```

fptsde1d(object, ...)
## Default S3 method:
fptsde1d(object, boundary, ...)
## S3 method for class 'fptsde1d'
summary(object, digits=5, ...)
## S3 method for class 'fptsde1d'
plot(x, ...)

```

Arguments

<code>object</code>	an object inheriting from class <code>snssde1d</code> .
<code>boundary</code>	an expression of a constant or time-dependent boundary.
<code>digits</code>	integer, used for number formatting.
<code>x</code>	an object inheriting from class <code>fptsde1d</code> .
<code>...</code>	further arguments for (non-default) methods.

Details

The function `fptsde1d` returns a random variable $\tau_{(X(t),S(t))}$ "first passage time", is defined as :

$$\tau_{(X(t),S(t))} = \{t \geq 0; X_t \geq S(t)\}, \quad \text{if } X(t_0) < S(t_0)$$

$$\tau_{(X(t),S(t))} = \{t \geq 0; X_t \leq S(t)\}, \quad \text{if } X(t_0) > S(t_0)$$

with $S(t)$ is through a continuous boundary (barrier).

Value

`fptsde1d` returns an object inheriting from `class "fptsde1d"`.

`fpt` numeric vector of fpt (first passage time).

Author(s)

A.C. Guidoum, K. Boukhetala.

References

- Argyrakisa, P. and G.H. Weiss (2006). A first-passage time problem for many random walkers. *Physica A*. **363**, 343–347.
- Aytug H., G. J. Koehler (2000). New stopping criterion for genetic algorithms. *European Journal of Operational Research*, **126**, 662–674.
- Boukhetala, K. (1996) Modelling and simulation of a dispersion pollutant with attractive centre. ed by Computational Mechanics Publications, Southampton ,U.K and Computational Mechanics Inc, Boston, USA, 245–252.
- Boukhetala, K. (1998a). Estimation of the first passage time distribution for a simulated diffusion process. *Maghreb Math.Rev*, **7**(1), 1–25.
- Boukhetala, K. (1998b). Kernel density of the exit time in a simulated diffusion. *les Annales Maghrebines De L ingénieur*, **12**, 587–589.
- Ding, M. and G. Rangarajan. (2004). First Passage Time Problem: A Fokker-Planck Approach. *New Directions in Statistical Physics*. ed by L. T. Wille. Springer. 31–46.
- Roman, R.P., Serrano, J. J., Torres, F. (2008). First-passage-time location function: Application to determine first-passage-time densities in diffusion processes. *Computational Statistics and Data Analysis*. **52**, 4132–4146.
- Roman, R.P., Serrano, J. J., Torres, F. (2012). An R package for an efficient approximation of first-passage-time densities for diffusion processes based on the FPTL function. *Applied Mathematics and Computation*, **218**, 8408–8428.
- Gardiner, C. W. (1997). *Handbook of Stochastic Methods*. Springer-Verlag, New York.

See Also

[fptsde2d](#) and [fptsde3d](#) simulation fpt for 2 and 3-dim SDE.

[FPTL](#) for computes values of the first passage time location (FPTL) function, and [Approx.fpt.density](#) for approximate first-passage-time (f.p.t.) density in package "fptdApprox".

Examples

```
## Example 1: Ito SDE
## dX(t) = -4*X(t) *dt + 0.5*dW(t)
## S(t) = 0 (constant boundary)
set.seed(1234)

f <- expression( -4*x )
g <- expression( 0.5 )
St <- expression(0)
mod <- snssde1d(drift=f,diffusion=g,x0=2,M=50)
fptmod <- fptsde1d(mod, boundary=St)
fptmod
summary(fptmod)
plot(fptmod)
plot(density(fptmod$fpt[!is.na(fptmod$fpt)]),main="Kernel Density of a First-Passage-Time")

## Example 2: Ito SDE
## X(t) Brownian motion
## S(t) = 0.3+0.2*t (time-dependent boundary)
set.seed(1234)

f <- expression( 0 )
g <- expression( 1 )
St <- expression(0.5-0.5*t)
mod1 <- snssde1d(drift=f,diffusion=g,M=40)
fptmod1 <- fptsde1d(mod1,boundary=St)
summary(fptmod1)
plot(fptmod1)
plot(density(fptmod1$fpt[!is.na(fptmod1$fpt)]),main="Kernel Density of a First-Passage-Time")

## Example 3: Stratonovich SDE
## dX(t) = 0.5*X(t)*t *dt + sqrt(1+X(t)^2) o dW(t)
## S(t) = -0.5*sqrt(t) + exp(t^2) (time-dependent boundary)
set.seed(1234)

f <- expression( 0.5*x*t )
g <- expression( sqrt(1+x^2) )
St <- expression(-0.5*sqrt(t)+exp(t^2))
mod2 <- snssde1d(drift=f,diffusion=g,x0=2,M=40,type="srt")
fptmod2 <- fptsde1d(mod2,boundary=St)
summary(fptmod2)
plot(density(fptmod2$fpt[!is.na(fptmod2$fpt)]),main="Kernel Density of a First-Passage-Time")
```

fptsde2d

First Passage Time in 2-Dim SDE

Description

The (S3) generic function `fptsde2d` for simulate first-passage-time (f.p.t) in 2-dim stochastic differential equations.

Usage

```
fptsde2d(object, ...)
## Default S3 method:
fptsde2d(object, boundary, ...)
## S3 method for class 'fptsde2d'
summary(object, digits=5, ...)
## S3 method for class 'fptsde2d'
plot(x, ...)
```

Arguments

object an object inheriting from class `snssde2d`.

boundary an `expression` of a constant or time-dependent boundary.

digits integer, used for number formatting.

x an object inheriting from class `fptsde2d`.

... further arguments for (non-default) methods.

Details

The function `fptsde1d` returns a random variable $(\tau_{(X(t),S(t))}, \tau_{(Y(t),S(t))})$ "first passage time", is defined as :

$$\tau_{(X(t),S(t))} = \{t \geq 0; X_t \geq S(t)\}, \quad \text{if } X(t_0) < S(t_0)$$

$$\tau_{(Y(t),S(t))} = \{t \geq 0; Y_t \geq S(t)\}, \quad \text{if } Y(t_0) < S(t_0)$$

and:

$$\tau_{(X(t),S(t))} = \{t \geq 0; X_t \leq S(t)\}, \quad \text{if } X(t_0) > S(t_0)$$

$$\tau_{(Y(t),S(t))} = \{t \geq 0; Y_t \leq S(t)\}, \quad \text{if } Y(t_0) > S(t_0)$$

with $S(t)$ is through a continuous boundary (barrier).

Value

`fptsde2d` returns an object inheriting from class `"fptsde2d"`.

`fptx`, `fpty` a vector of couple 'fpt' $(\tau_{(X(t),S(t))}, \tau_{(Y(t),S(t))})$.

Author(s)

A.C. Guidoum, K. Boukhetala.

References

Argyrakisa, P. and G.H. Weiss (2006). A first-passage time problem for many random walkers. *Physica A*. **363**, 343–347.

Aytug H., G. J. Koehler (2000). New stopping criterion for genetic algorithms. *European Journal of Operational Research*, **126**, 662–674.

Boukhetala, K. (1996) Modelling and simulation of a dispersion pollutant with attractive centre. ed by Computational Mechanics Publications, Southampton ,U.K and Computational Mechanics Inc, Boston, USA, 245–252.

Boukhetala, K. (1998a). Estimation of the first passage time distribution for a simulated diffusion process. *Maghreb Math.Rev*, **7**(1), 1–25.

Boukhetala, K. (1998b). Kernel density of the exit time in a simulated diffusion. *les Annales Maghrebines De L ingénieur*, **12**, 587–589.

Ding, M. and G. Rangarajan. (2004). First Passage Time Problem: A Fokker-Planck Approach. *New Directions in Statistical Physics*. ed by L. T. Wille. Springer. 31–46.

Roman, R.P., Serrano, J. J., Torres, F. (2008). First-passage-time location function: Application to determine first-passage-time densities in diffusion processes. *Computational Statistics and Data Analysis*. **52**, 4132–4146.

Roman, R.P., Serrano, J. J., Torres, F. (2012). An R package for an efficient approximation of first-passage-time densities for diffusion processes based on the FPTL function. *Applied Mathematics and Computation*, **218**, 8408–8428.

Gardiner, C. W. (1997). *Handbook of Stochastic Methods*. Springer-Verlag, New York.

See Also

[fptsde1d](#) for simulation fpt in sde 1-dim. [fptsde3d](#) for simulation fpt in sde 3-dim.

[FPTL](#) for computes values of the first passage time location (FPTL) function, and [Approx.fpt.density](#) for approximate first-passage-time (f.p.t.) density in package "fptdApprox".

Examples

```
## dX(t) = 5*(-1-Y(t))*X(t) * dt + 0.5 * dW1(t)
## dY(t) = 5*(-1-X(t))*Y(t) * dt + 0.5 * dW2(t)
## x0 = 2, y0 = -2, and barrier -3+5*t.
## W1(t) and W2(t) two independent Brownian motion
set.seed(1234)

fx <- expression(5*(-1-y)*x)
gx <- expression(0.5)
fy <- expression(5*(-1-x)*y)
gy <- expression(0.5)
St <- expression(-1+5*t)

mod2d <- snssde2d(driftx=fx,diffx=gx,drifty=fy,diffy=gy,x0=2,y0=-2,M=50)
fptmod2d <- fptsde2d(mod2d,boundary=St)
fptmod2d
summary(fptmod2d)
plot(fptmod2d )

##

X1 <- cbind(fptmod2d$fptx,fptmod2d$fpty)
## library(sm)
## sm.density(X1,display="persp")
```


Description

The (S3) generic function `fptsde3d` for simulate first-passage-time (f.p.t) in 3-dim stochastic differential equations.

Usage

```
fptsde3d(object, ...)
## Default S3 method:
fptsde3d(object, boundary, ...)
## S3 method for class 'fptsde3d'
summary(object, digits=5, ...)
## S3 method for class 'fptsde3d'
plot(x, ...)
```

Arguments

<code>object</code>	an object inheriting from class <code>snssde3d</code> .
<code>boundary</code>	an expression of a constant or time-dependent boundary.
<code>digits</code>	integer, used for number formatting.
<code>x</code>	an object inheriting from class <code>fptsde3d</code> .
<code>...</code>	further arguments for (non-default) methods.

Details

The function `fptsde3d` returns a random variable $(\tau_{(X(t),S(t))}, \tau_{(Y(t),S(t))}, \tau_{(Z(t),S(t))})$ "first passage time", is defined as :

$$\tau_{(X(t),S(t))} = \{t \geq 0; X_t \geq S(t)\}, \quad \text{if } X(t_0) < S(t_0)$$

$$\tau_{(Y(t),S(t))} = \{t \geq 0; Y_t \geq S(t)\}, \quad \text{if } Y(t_0) < S(t_0)$$

$$\tau_{(Z(t),S(t))} = \{t \geq 0; Z_t \geq S(t)\}, \quad \text{if } Z(t_0) < S(t_0)$$

and:

$$\tau_{(X(t),S(t))} = \{t \geq 0; X_t \leq S(t)\}, \quad \text{if } X(t_0) > S(t_0)$$

$$\tau_{(Y(t),S(t))} = \{t \geq 0; Y_t \leq S(t)\}, \quad \text{if } Y(t_0) > S(t_0)$$

$$\tau_{(Z(t),S(t))} = \{t \geq 0; Z_t \leq S(t)\}, \quad \text{if } Z(t_0) > S(t_0)$$

with $S(t)$ is through a continuous boundary (barrier).

Value

fptsde3d returns an object inheriting from `class "fptsde3d"`.

fptx, fpty, fptz

a vector of triplet 'fpt' $(\tau_{(X(t),S(t))}, \tau_{(Y(t),S(t))}, \tau_{(Z(t),S(t))})$.

Author(s)

A.C. Guidoum, K. Boukhetala.

References

- Argyrakisa, P. and G.H. Weiss (2006). A first-passage time problem for many random walkers. *Physica A*, **363**, 343–347.
- Aytug H., G. J. Koehler (2000). New stopping criterion for genetic algorithms. *European Journal of Operational Research*, **126**, 662–674.
- Boukhetala, K. (1996) Modelling and simulation of a dispersion pollutant with attractive centre. ed by Computational Mechanics Publications, Southampton ,U.K and Computational Mechanics Inc, Boston, USA, 245–252.
- Boukhetala, K. (1998a). Estimation of the first passage time distribution for a simulated diffusion process. *Maghreb Math.Rev*, **7**(1), 1–25.
- Boukhetala, K. (1998b). Kernel density of the exit time in a simulated diffusion. *les Annales Maghrebines De L ingénieur*, **12**, 587–589.
- Ding, M. and G. Rangarajan. (2004). First Passage Time Problem: A Fokker-Planck Approach. *New Directions in Statistical Physics*. ed by L. T. Wille. Springer. 31–46.
- Roman, R.P., Serrano, J. J., Torres, F. (2008). First-passage-time location function: Application to determine first-passage-time densities in diffusion processes. *Computational Statistics and Data Analysis*. **52**, 4132–4146.
- Roman, R.P., Serrano, J. J., Torres, F. (2012). An R package for an efficient approximation of first-passage-time densities for diffusion processes based on the FPTL function. *Applied Mathematics and Computation*, **218**, 8408–8428.
- Gardiner, C. W. (1997). *Handbook of Stochastic Methods*. Springer-Verlag, New York.

See Also

[fptsde1d](#) for simulation fpt in sde 1-dim. [fptsde2d](#) for simulation fpt in sde 2-dim.

[FPTL](#) for computes values of the first passage time location (FPTL) function, and [Approx.fpt.density](#) for approximate first-passage-time (f.p.t.) density in package "fptdApprox".

Examples

```
## dX(t) = 4*(-1-X(t))*Y(t) dt + 0.2 * dW1(t)
## dY(t) = 4*(1-Y(t)) *X(t) dt + 0.2 * dW2(t)
## dZ(t) = 4*(1-Z(t)) *Y(t) dt + 0.2 * dW3(t)
## x0 = 0, y0 = -2, z0 = 0, and barrier -3+5*t.
## W1(t), W2(t) and W3(t) three independent Brownian motion
```

```

set.seed(1234)

fx <- expression(4*(-1-x)*y)
gx <- expression(0.2)
fy <- expression(4*(1-y)*x)
gy <- expression(0.2)
fz <- expression(4*(1-z)*y)
gz <- expression(0.2)

St <- expression(-3+5*t)

mod3d <- snssde3d(driftx=fx,diffx=gx,drifty=fy,diffy=gy,driftz=fz,diffz=gz,
                 x0=2,y0=-2,z0=0,M=50)
fptmod3d <- fptsde3d(mod3d,boundary=St)
fptmod3d
summary(fptmod3d)
plot(fptmod3d,union=TRUE)
dev.new()
plot(fptmod3d,union=FALSE)
##

X <- cbind(fptmod3d$fptx,fptmod3d$fpty,fptmod3d$fptz)
## library(sm)
## sm.density(X,display="rgl")

```

HWV

Hull-White/Vasicek, Ornstein-Uhlenbeck process

Description

The (S3) generic function for simulation of Hull-White/Vasicek or gaussian diffusion models, and Ornstein-Uhlenbeck process.

Usage

```

HWV(N, ...)
OU(N, ...)

## Default S3 method:
HWV(N = 100, M = 1, x0 = 2, t0 = 0, T = 1, Dt, mu = 4, theta = 1,
     sigma = 0.1, ...)
## Default S3 method:
OU(N = 100, M = 1, x0 = 2, t0 = 0, T = 1, Dt, mu = 4, sigma = 0.2, ...)

```

Arguments

N	number of simulation steps.
M	number of trajectories.
x0	initial value of the process at time t_0 .

t0	initial time.
T	final time.
Dt	time step of the simulation (discretization). If it is <code>missing</code> a default $\Delta t = \frac{T-t_0}{N}$.
mu	parameter of the HWV and OU; see details.
theta	parameter of the HWV; see details.
sigma	the volatility of the HWV and OU.
...	further arguments for (non-default) methods.

Details

The function HWV returns a trajectory of the **Hull-White/Vasicek process** starting at x_0 at time t_0 ; i.e., the diffusion process solution of stochastic differential equation:

$$dX_t = \mu(\theta - X_t)dt + \sigma dW_t$$

The function OU returns a trajectory of the **Ornstein-Uhlenbeck** starting at x_0 at time t_0 ; i.e., the diffusion process solution of stochastic differential equation:

$$dX_t = -\mu X_t dt + \sigma dW_t$$

Constraints: $\mu, \sigma > 0$.

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

Value

X an visible ts object.

Author(s)

A.C. Guidoum, K. Boukhetala.

References

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

See Also

`rcOU` and `rsOU` for conditional and stationary law of Vasicek process are available in "sde".

Examples

```
## Hull-White/Vasicek Models
##  $dX(t) = 4 * (2.5 - X(t)) * dt + 1 * dW(t)$ ,  $X_0=10$ 
set.seed(1234)

X <- HWV(N=1000,M=50,mu = 4, theta = 2.5,sigma = 1,x0=10)
plot(X,plot.type="single")
lines(as.vector(time(X)),rowMeans(X),col="red")

## Ornstein-Uhlenbeck Process
##  $dX(t) = -4 * X(t) * dt + 1 * dW(t)$ ,  $X_0=2$ 
set.seed(1234)

X <- OU(N=1000,M=50,mu = 4,sigma = 1,x0=10)
plot(X,plot.type="single")
lines(as.vector(time(X)),rowMeans(X),col="red")
```

Irates

Monthly Interest Rates

Description

monthly observations from 1946–12 to 1991–02

number of observations : 531

observation : country

country : United–States

Usage

data(Irates)

Format

A time serie containing :

r1 interest rate for a maturity of 1 months (% per year).

r2 interest rate for a maturity of 2 months (% per year).

r3 interest rate for a maturity of 3 months (% per year).

r5 interest rate for a maturity of 5 months (% per year).

r6 interest rate for a maturity of 6 months (% per year).

r11 interest rate for a maturity of 11 months (% per year).

r12 interest rate for a maturity of 12 months (% per year).

r36 interest rate for a maturity of 36 months (% per year).

r60 interest rate for a maturity of 60 months (% per year).

r120 interest rate for a maturity of 120 months (% per year).

Source

McCulloch, J.H. and Kwon, H.C. (1993). U.S. term structure data, 1947–1991, Ohio State Working Paper 93–6, Ohio State University, Columbus

These datasets `Irates` are in package "Ecdat".

References

Croissant, Y. (2014). Ecdat: Data sets for econometrics. R package version 0.2–5.

Examples

```
data(Irates)
rates <- Irates[, "r1"]
rates <- window(rates, start=1964.471, end=1989.333)

## CKLS modele vs CIR modele
## CKLS :  $dX(t) = (\theta_1 + \theta_2 * X(t)) * dt + \theta_3 * X(t)^{\theta_4} * dW(t)$ 

fx <- expression(theta[1]+theta[2]*x)
gx <- expression(theta[3]*x^theta[4])
fitmod <- fitsde(rates,drift=fx,diffusion=gx,pmle="euler",start = list(theta1=1,theta2=1,
theta3=1,theta4=1),optim.method = "L-BFGS-B")
theta <- coef(fitmod)

N <- length(rates)
res <- snssde1d(drift=fx,diffusion=gx,M=200,t0=time(rates)[1],T=time(rates)[N],
Dt=deltat(rates),x0=rates[1],N)
plot(res,plot.type="single",ylim=c(0,50))
lines(rates,col=2,lwd=2)
legend("topleft",c("real data","CKLS modele"),inset = .01,col=c(2,1),lwd=2,cex=0.8)

dev.new()

plot(res,plot.type="single",type="n",ylim=c(0,35))
lines(rates,col=2,lwd=2)
lines(time(res),mean(res),col=3,lwd=2)
lines(time(res),bconfint(res,level=0.95)[,1],col=4,lwd=2)
lines(time(res),bconfint(res,level=0.95)[,2],col=4,lwd=2)
legend("topleft",c("real data","mean path",paste("bound of", 95,"percent confidence")),
inset = .01,col=2:4,lwd=2,cex=0.8)
```

Description

Generic function for plotting.

Usage

```
## Default S3 method:
plot2d(x, ...)
## Default S3 method:
lines2d(x, ...)
## Default S3 method:
points2d(x, ...)
## Default S3 method:
plot3D(x, display = c("persp", "rgl"), ...)
```

Arguments

x an object inheriting from class `snsde2d`, `snsde3d`, `bridgesde2d` and `bridgesde3d`.
display "persp" perspective or "rgl" plots.
... other graphics parameters, see `par` in package "graphics".

Details

The 2 and 3-dim plot of class `sde`.

Author(s)

A.C. Guidoum, K. Boukhetala.

Examples

```
## Example 1:
set.seed(1234)

fx <- expression(0)
gx <- expression(1)
fy <- expression(0)
gy <- expression(1)

res <- snsde2d(driftx=fx,diffx=gx,drifty=fy,diffy=gy,N=10000)
plot2d(res,type="l")

## Example 2:
set.seed(1234)

fx <- expression(0)
gx <- expression(1)
fy <- expression(0)
gy <- expression(1)
fz <- expression(0)
gz <- expression(1)

res <- snsde3d(driftx=fx,diffx=gx,drifty=fy,diffy=gy,driftz=fz,diffz=gz,N=10000)
plot3D(res,display="persp")
```

rsde1d *Random Number Generators for SDE's at time t*

Description

The (S3) generic function `rsde1d`, `rsde2d` and `rsde3d` for random number generated of the process at time at .

Usage

```
rsde1d(object, ...)
rsde2d(object, ...)
rsde3d(object, ...)
## Default S3 method:
rsde1d(object, at, ...)
## Default S3 method:
rsde2d(object, at, ...)
## Default S3 method:
rsde3d(object, at, ...)
```

Arguments

`object` an object inheriting from class `snssde1d`, `snssde2d` and `snssde3d`.
`at` time between t_0 and T . Random number generated of the process at time at . The default $at = T$.
`...` further arguments for (non-default) methods.

Details

The function `codersde1d`, `rsde2d` and `rsde3d` returns a random variable $x_{t=at}$ realize at time $t = at$ defined by :

$$x_{t=at} = \{t \geq 0; x = X_{t=at}\}$$

with at is a fixed time between t_0 and T .

Value

`rsde1d` returns an object inheriting from class `"rsde1d"`. `rsde2d` returns an object inheriting from class `"rsde2d"`. `rsde3d` returns an object inheriting from class `"rsde3d"`.

Author(s)

A.C. Guidoum, K. Boukhetala.

See Also

`rng` random number generators in "yuima" package.
`rcBS`, `rcCIR`, `rcOU` and `rsOU` in package "sde".

Examples

```

## 1-dim SDE
## Example 1: Stratonovich sde
##  $dX(t) = (-2*(X(t) \leq 0) + 2*(X(t) > 0)) * dt + 0.5 \circ dW(t)$ 
set.seed(1234)

f <- expression(-2*(x<=0)+2*(x>=0))
g <- expression(0.5)
res1 <- snssde1d(drift=f,diffusion=g,M=25,type="str",T=10)
x <- rsde1d(res1)
summary(x)
plot(density(x))

## 2-dim SDE's
## Example 2:
## random numbers of two standard Brownian motion W1(t) and W2(t) at time = 0.5
set.seed(1234)

fx <- expression(0)
gx <- expression(1)
fy <- expression(0)
gy <- expression(1)
res <- snssde2d(driftx=fx,diffx=gx,drifty=fy,diffy=gy,M=25)

Data2d <- rsde2d(res,at=0.5)
summary(Data2d)
## library(sm)
## sm.density(Data2d,display="persp")

## 3-dim SDE's

## Example 3: Ito sde 3-dim
##  $dX(t) = 4*(-1-X(t))*Y(t) dt + 0.2 * dW1(t)$ 
##  $dY(t) = 4*(1-Y(t)) * X(t) dt + 0.2 * dW2(t)$ 
##  $dZ(t) = 4*(1-Z(t)) * Y(t) dt + 0.2 * dW3(t)$ 
## W1(t), W2(t) and W3(t) three independent Brownian motion
set.seed(1234)

fx <- expression(4*(-1-x)*y)
gx <- expression(0.2)
fy <- expression(4*(1-y)*x)
gy <- expression(0.2)
fz <- expression(4*(1-z)*y)
gz <- expression(0.2)

res <- snssde3d(driftx=fx,diffx=gx,drifty=fy,diffy=gy,driftz=fz,diffz=gz,
               x0=2,y0=-2,z0=0,M=25)
Data3d <- rsde3d(res,at=0.75)
summary(Data3d)
## library(sm)
## sm.density(Data3d,display="rgl")

```

snssde1d

*Simulation of 1-Dim Stochastic Differential Equation***Description**

The (S3) generic function snssde1d of simulation of solutions to 1-dim stochastic differential equations of Ito or Stratonovich type, with different methods.

Usage

```
snssde1d(N, ...)
## Default S3 method:
snssde1d(N = 1000, M = 1, x0 = 0, t0 = 0, T = 1, Dt,
  drift, diffusion, alpha = 0.5, mu = 0.5, type = c("ito", "str"),
  method = c("euler", "milstein", "predcorr", "smilstein", "taylor",
    "heun", "rk1", "rk2", "rk3"), ...)

## S3 method for class 'snssde1d'
summary(object, at ,digits=5, ...)
## S3 method for class 'snssde1d'
time(x, ...)
## S3 method for class 'snssde1d'
mean(x, ...)
## S3 method for class 'snssde1d'
median(x, ...)
## S3 method for class 'snssde1d'
quantile(x, ...)
## S3 method for class 'snssde1d'
kurtosis(x, ...)
## S3 method for class 'snssde1d'
skewness(x, ...)
## S3 method for class 'snssde1d'
moment(x, order = 2, ...)
## S3 method for class 'snssde1d'
bconfint(x, level=0.95, ...)
## S3 method for class 'snssde1d'
plot(x, ...)
## S3 method for class 'snssde1d'
lines(x, ...)
## S3 method for class 'snssde1d'
points(x, ...)
```

Arguments

N number of simulation steps.
M number of trajectories.

<code>x0</code>	initial value of the process at time <code>t0</code> .
<code>t0</code>	initial time.
<code>T</code>	ending time.
<code>Dt</code>	time step of the simulation (discretization). If it is <code>missing</code> a default $\Delta t = \frac{T-t_0}{N}$.
<code>drift</code>	drift coefficient: an <code>expression</code> of two variables <code>t</code> and <code>x</code> .
<code>diffusion</code>	diffusion coefficient: an <code>expression</code> of two variables <code>t</code> and <code>x</code> .
<code>alpha, mu</code>	weight of the predictor-corrector scheme; the default <code>alpha = 0.5</code> and <code>mu = 0.5</code> .
<code>type</code>	if <code>type="ito"</code> simulation sde of Ito type, else <code>type="str"</code> simulation sde of Stratonovich type; the default <code>type="ito"</code> .
<code>method</code>	numerical methods of simulation, the default <code>method = "euler"</code> .
<code>x, object</code>	an object inheriting from class <code>"snssde1d"</code> .
<code>at</code>	time between <code>t0</code> and <code>T</code> . Monte-Carlo statistics of the solution at time <code>at</code> . The default <code>at = T</code> .
<code>order</code>	order of moment.
<code>level</code>	the confidence level required.
<code>digits</code>	integer, used for number formatting.
<code>...</code>	further arguments for (non-default) methods.

Details

The function `snssde1d` returns a `ts` `x` of length `N+1`; i.e. solution of the sde of Ito or Stratonovich types; If `Dt` is not specified, then the best discretization $\Delta t = \frac{T-t_0}{N}$.

The Ito stochastic differential equation is:

$$dX(t) = a(t, X(t))dt + b(t, X(t))dW(t)$$

Stratonovich sde :

$$dX(t) = a(t, X(t))dt + b(t, X(t)) \circ dW(t)$$

The methods of approximation are classified according to their different properties. Mainly two criteria of optimality are used in the literature: the strong and the weak (orders of) convergence. The method of simulation can be one among: Euler-Maruyama Order 0.5, Milstein Order 1, Milstein Second-Order, Predictor-Corrector method, Ito-Taylor Order 1.5, Heun Order 2 and Runge-Kutta Order 1, 2 and 3.

For more details see `vignette("SDEs")`.

Value

`snssde1d` returns an object inheriting from `class "snssde1d"`.

<code>X</code>	an invisible <code>ts</code> object.
<code>drift</code>	drift coefficient.
<code>diffusion</code>	diffusion coefficient.
<code>type</code>	type of sde.
<code>method</code>	the numerical method used.

Author(s)

A.C. Guidoum, K. Boukhetala.

References

- Friedman, A. (1975). *Stochastic differential equations and applications*. Volume 1, ACADEMIC PRESS.
- Henderson, D. and Plaschko, P. (2006). *Stochastic differential equations in science and engineering*. World Scientific.
- Allen, E. (2007). *Modeling with Ito stochastic differential equations*. Springer-Verlag.
- Jedrzejewski, F. (2009). *Modeles aleatoires et physique probabiliste*. Springer-Verlag.
- Iacus, S.M. (2008). *Simulation and inference for stochastic differential equations: with R examples*. Springer-Verlag, New York.
- Kloeden, P.E. and Platen, E. (1989). A survey of numerical methods for stochastic differential equations. *Stochastic Hydrology and Hydraulics*, **3**, 155–178.
- Kloeden, P.E. and Platen, E. (1991a). Relations between multiple Ito and Stratonovich integrals. *Stochastic Analysis and Applications*, **9**(3), 311–321.
- Kloeden, P.E. and Platen, E. (1991b). Stratonovich and Ito stochastic Taylor expansions. *Mathematische Nachrichten*, **151**, 33–50.
- Kloeden, P.E. and Platen, E. (1995). *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, New York.
- Oksendal, B. (2000). *Stochastic Differential Equations: An Introduction with Applications*. 5th edn. Springer-Verlag, Berlin.
- Platen, E. (1980). Weak convergence of approximations of Ito integral equations. *Z Angew Math Mech*. **60**, 609–614.
- Platen, E. and Bruti-Liberati, N. (2010). *Numerical Solution of Stochastic Differential Equations with Jumps in Finance*. Springer-Verlag, New York.
- Saito, Y. and Mitsui, T. (1993). Simulation of Stochastic Differential Equations. *The Annals of the Institute of Statistical Mathematics*, **3**, 419–432.

See Also

[snssde2d](#) and [snssde3d](#) for 2 and 3-dim sde.
[sde.sim](#) in package "sde".
[simulate](#) in package "yuima".

Examples

```
## Example 1: Ito sde
## dX(t) = 2*(3-X(t)) dt + 2*X(t) dW(t)
set.seed(1234)

f <- expression(2*(3-x) )
g <- expression(2*x)
```

```

res <- snssde1d(drift=f,diffusion=g,M=50,x0=1,N=1000)
res
## Sim <- res$X
summary(res)
plot(res,plot.type="single")
lines(time(res),mean(res),col=2,lwd=2)
lines(time(res),bconfint(res,level=0.95)[,1],col=4,lwd=2)
lines(time(res),bconfint(res,level=0.95)[,2],col=4,lwd=2)
legend("topleft",c("mean path",paste("bound of", 95,"percent confidence")),
      inset = .01,col=c(2,4),lwd=2,cex=0.8)

## Example 2: Stratonovich sde
##  $dX(t) = ((2-X(t))/(2-t)) dt + X(t) \circ dW(t)$ 
set.seed(1234)

f <- expression((2-x)/(2-t))
g <- expression(x)
res1 <- snssde1d(type="str",drift=f,diffusion=g,M=50,x0=1,N=1000,
                method="milstein")
res1
summary(res1,at = 0.89475)
plot(res1,plot.type="single")
lines(time(res1),mean(res1),col=2,lwd=2)
lines(time(res1),bconfint(res1,level=0.95)[,1],col=4,lwd=2)
lines(time(res1),bconfint(res1,level=0.95)[,2],col=4,lwd=2)
legend("topleft",c("mean path",paste("bound of", 95,"percent confidence")),
      inset = .01,col=c(2,4),lwd=2,cex=0.8)

```

snssde2d

Simulation of 2-Dim Stochastic Differential Equation

Description

The (S3) generic function `snssde2d` of simulation of solutions to 2-dim stochastic differential equations of Ito or Stratonovich type, with different methods.

Usage

```

snssde2d(N, ...)
## Default S3 method:
snssde2d(N = 1000, M = 1, x0 = 0, y0 = 0, t0 = 0, T = 1, Dt,
         driftx, diffx, drifty, diffy, alpha = 0.5, mu = 0.5,
         type = c("ito", "str"), method = c("euler", "milstein",
         "predcorr", "smilstein", "taylor", "heun", "rk1", "rk2",
         "rk3"), ...)

## S3 method for class 'snssde2d'
summary(object, at , digits=5,...)

```

```

## S3 method for class 'snssde2d'
time(x, ...)
## S3 method for class 'snssde2d'
mean(x, ...)
## S3 method for class 'snssde2d'
median(x, ...)
## S3 method for class 'snssde2d'
quantile(x, ...)
## S3 method for class 'snssde2d'
kurtosis(x, ...)
## S3 method for class 'snssde2d'
skewness(x, ...)
## S3 method for class 'snssde2d'
moment(x, order = 2, ...)
## S3 method for class 'snssde2d'
bconfint(x, level=0.95, ...)
## S3 method for class 'snssde2d'
plot(x, ...)
## S3 method for class 'snssde2d'
lines(x, ...)
## S3 method for class 'snssde2d'
points(x, ...)
## S3 method for class 'snssde2d'
plot2d(x, ...)
## S3 method for class 'snssde2d'
lines2d(x, ...)
## S3 method for class 'snssde2d'
points2d(x, ...)

```

Arguments

N	number of simulation steps.
M	number of trajectories.
x_0, y_0	initial value of the process X_t and Y_t at time t_0 .
t_0	initial time.
T	ending time.
Dt	time step of the simulation (discretization). If it is <code>missing</code> a default $\Delta t = \frac{T-t_0}{N}$.
driftx, drifty	drift coefficient: an expression of three variables t, x and y for process X_t and Y_t .
diffx, diffy	diffusion coefficient: an expression of three variables t, x and y for process X_t and Y_t .
alpha, mu	weight of the predictor-corrector scheme; the default alpha = 0.5 and mu = 0.5.
type	if type="ito" simulation sde of Ito type, else type="str" simulation sde of Stratonovich type; the default type="ito".
method	numerical methods of simulation, the default method = "euler".

x, object	an object inheriting from class "snssde2d".
at	time between t_0 and T. Monte-Carlo statistics of the solutions at time at. The default at = T.
order	order of moment.
level	the confidence level required.
digits	integer, used for number formatting.
...	further arguments for (non-default) methods.

Details

The function snssde2d returns a mts x of length N+1; i.e. solution of the 2-dim sde (X_t, Y_t) of Ito or Stratonovich types; If Dt is not specified, then the best discretization $\Delta t = \frac{T-t_0}{N}$.

The 2-dim Ito stochastic differential equation is:

$$dX(t) = a(t, X(t), Y(t))dt + b(t, X(t), Y(t))dW_1(t)$$

$$dY(t) = a(t, X(t), Y(t))dt + b(t, X(t), Y(t))dW_2(t)$$

2-dim Stratonovich sde :

$$dX(t) = a(t, X(t), Y(t))dt + b(t, X(t), Y(t)) \circ dW_1(t)$$

$$dY(t) = a(t, X(t), Y(t))dt + b(t, X(t), Y(t)) \circ dW_2(t)$$

$W_1(t), W_2(t)$ two standard Brownian motion independent.

The methods of approximation are classified according to their different properties. Mainly two criteria of optimality are used in the literature: the strong and the weak (orders of) convergence. The method of simulation can be one among: Euler-Maruyama Order 0.5, Milstein Order 1, Milstein Second-Order, Predictor-Corrector method, Ito-Taylor Order 1.5, Heun Order 2 and Runge-Kutta Order 1, 2 and 3.

For more details see vignette("SDEs").

Value

snssde2d returns an object inheriting from class "snssde2d".

X, Y	an invisible mts (2-dim) object (X(t),Y(t)).
driftx, drifty	drift coefficient of X(t) and Y(t).
diffx, diffy	diffusion coefficient of X(t) and Y(t).
type	type of sde.
method	the numerical method used.

Author(s)

A.C. Guidoum, K. Boukhetala.

References

- Friedman, A. (1975). *Stochastic differential equations and applications*. Volume 1, ACADEMIC PRESS.
- Henderson, D. and Plaschko, P. (2006). *Stochastic differential equations in science and engineering*. World Scientific.
- Allen, E. (2007). *Modeling with Ito stochastic differential equations*. Springer-Verlag.
- Jedrzejewski, F. (2009). *Modeles aleatoires et physique probabiliste*. Springer-Verlag.
- Iacus, S.M. (2008). *Simulation and inference for stochastic differential equations: with R examples*. Springer-Verlag, New York.
- Kloeden, P.E, and Platen, E. (1989). A survey of numerical methods for stochastic differential equations. *Stochastic Hydrology and Hydraulics*, **3**, 155–178.
- Kloeden, P.E, and Platen, E. (1991a). Relations between multiple ito and stratonovich integrals. *Stochastic Analysis and Applications*, **9**(3), 311–321.
- Kloeden, P.E, and Platen, E. (1991b). Stratonovich and ito stochastic taylor expansions. *Mathematische Nachrichten*, **151**, 33–50.
- Kloeden, P.E, and Platen, E. (1995). *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, New York.
- Oksendal, B. (2000). *Stochastic Differential Equations: An Introduction with Applications*. 5th edn. Springer-Verlag, Berlin.
- Platen, E. (1980). Weak convergence of approximations of ito integral equations. *Z Angew Math Mech*. **60**, 609–614.
- Platen, E. and Bruti-Liberati, N. (2010). *Numerical Solution of Stochastic Differential Equations with Jumps in Finance*. Springer-Verlag, New York
- Saito, Y, and Mitsui, T. (1993). Simulation of Stochastic Differential Equations. *The Annals of the Institute of Statistical Mathematics*, **3**, 419–432.

See Also

- [snssde1d](#) for 1-dim sde.
- [sde.sim](#) in package "sde". [simulate](#) in package "yuima".

Examples

```
## Example 1: Ito sde
## dX(t) = 4*(-1-X(t))*Y(t) dt + 0.2 dW1(t)
## dY(t) = 4*(1-Y(t))*X(t) dt + 0.2 dW2(t)
set.seed(1234)

fx <- expression(4*(-1-x)*y)
gx <- expression(0.2)
fy <- expression(4*(1-y)*x)
gy <- expression(0.2)

res <- snssde2d(driftx=fx,diffx=gx,drifty=fy,diffy=gy,x0=1,y0=-1,M=50)
```



```

res
summary(res, at = res$T)
plot(res)
dev.new()
plot2d(res) ## in plane (0,X,Y)

## Example 2: Stratonovich sde
## dX(t) = Y(t) dt + 0 o dW1(t)
## dY(t) = (4*(1-X(t)^2)*Y(t) - X(t) ) dt + 0.2 o dW2(t)
set.seed(1234)

fx <- expression( y )
gx <- expression( 0 )
fy <- expression( (4*( 1-x^2 ))* y - x )
gy <- expression( 0.2)

res1 <- snssde2d(driftx=fx,diffx=gx,drifty=fy,diffy=gy,type="str",T=100,
                ,N=10000)

res1
plot(res1,pos=2)
dev.new()
plot(res1,union = FALSE)
dev.new()
plot2d(res1,type="n") ## in plane (0,X,Y)
points2d(res1,col=rgb(0,100,0,50,maxColorValue=255), pch=16)

```

snssde3d

Simulation of 3-Dim Stochastic Differential Equation

Description

The (S3) generic function `snssde3d` of simulation of solutions to 3-dim stochastic differential equations of Ito or Stratonovich type, with different methods.

Usage

```

snssde3d(N, ...)
## Default S3 method:
snssde3d(N = 1000, M = 1, x0 = 0, y0 = 0, z0 = 0, t0 = 0, T = 1, Dt,
         driftx, diffx, drifty, diffy, driftz, diffz, alpha = 0.5, mu = 0.5,
         type = c("ito", "str"), method = c("euler", "milstein", "predcorr",
         "smilstein", "taylor", "heun", "rk1", "rk2", "rk3"), ...)

## S3 method for class 'snssde3d'
summary(object, at, digits=5,...)
## S3 method for class 'snssde3d'
time(x, ...)
## S3 method for class 'snssde3d'

```

```

mean(x, ...)
## S3 method for class 'snssde3d'
median(x, ...)
## S3 method for class 'snssde3d'
quantile(x, ...)
## S3 method for class 'snssde3d'
kurtosis(x, ...)
## S3 method for class 'snssde3d'
skewness(x, ...)
## S3 method for class 'snssde3d'
moment(x, order = 2, ...)
## S3 method for class 'snssde3d'
bconfint(x, level=0.95, ...)
## S3 method for class 'snssde3d'
plot(x, ...)
## S3 method for class 'snssde3d'
lines(x, ...)
## S3 method for class 'snssde3d'
points(x, ...)
## S3 method for class 'snssde3d'
plot3D(x, display = c("persp","rgl"), ...)

```

Arguments

N	number of simulation steps.
M	number of trajectories.
x0, y0, z0	initial value of the process X_t, Y_t and Z_t at time t_0 .
t0	initial time.
T	ending time.
Dt	time step of the simulation (discretization). If it is <code>missing</code> a default $\Delta t = \frac{T-t_0}{N}$.
driftx, drifty, driftz	drift coefficient: an <code>expression</code> of four variables t, x, y and z for process X_t, Y_t and Z_t .
diffx, diffy, diffz	diffusion coefficient: an <code>expression</code> of four variables t, x, y and z for process X_t, Y_t and Z_t .
alpha, mu	weight of the predictor-corrector scheme; the default alpha = 0.5 and mu = 0.5.
type	if type="ito" simulation sde of Ito type, else type="str" simulation sde of Stratonovich type; the default type="ito".
method	numerical methods of simulation, the default method = "euler".
x, object	an object inheriting from class "snssde3d".
at	time between t_0 and T. Monte-Carlo statistics of the solutions at time at. The default at = T.
order	order of moment.

level	the confidence level required.
digits	integer, used for number formatting.
display	"persp" perspective or "rgl" plots.
...	further arguments for (non-default) methods.

Details

The function `snssde3d` returns a matrix `x` of length $N+1$; i.e. solution of the 3-dim sde (X_t, Y_t, Z_t) of Ito or Stratonovich types; If Δt is not specified, then the best discretization $\Delta t = \frac{T-t_0}{N}$.

The 3-dim Ito stochastic differential equation is:

$$dX(t) = a(t, X(t), Y(t), Z(t))dt + b(t, X(t), Y(t), Z(t))dW_1(t)$$

$$dY(t) = a(t, X(t), Y(t), Z(t))dt + b(t, X(t), Y(t), Z(t))dW_2(t)$$

$$dZ(t) = a(t, X(t), Y(t), Z(t))dt + b(t, X(t), Y(t), Z(t))dW_3(t)$$

3-dim Stratonovich sde :

$$dX(t) = a(t, X(t), Y(t), Z(t))dt + b(t, X(t), Y(t), Z(t)) \circ dW_1(t)$$

$$dY(t) = a(t, X(t), Y(t), Z(t))dt + b(t, X(t), Y(t), Z(t)) \circ dW_2(t)$$

$$dZ(t) = a(t, X(t), Y(t), Z(t))dt + b(t, X(t), Y(t), Z(t)) \circ dW_3(t)$$

$W_1(t), W_2(t), W_3(t)$ three standard Brownian motion independent.

The methods of approximation are classified according to their different properties. Mainly two criteria of optimality are used in the literature: the strong and the weak (orders of) convergence. The method of simulation can be one among: Euler-Maruyama Order 0.5, Milstein Order 1, Milstein Second-Order, Predictor-Corrector method, Ito-Taylor Order 1.5, Heun Order 2 and Runge-Kutta Order 1, 2 and 3.

For more details see `vignette("SDEs")`.

Value

`snssde3d` returns an object inheriting from `class "snssde3d"`.

`X, Y, Z` an invisible matrix (3-dim) object $(X(t), Y(t), Z(t))$.

`driftx, drifty, driftz`
drift coefficient of $X(t), Y(t)$ and $Z(t)$.

`diffx, diffy, diffz`
diffusion coefficient of $X(t), Y(t)$ and $Z(t)$.

`type` type of sde.

`method` the numerical method used.

Author(s)

A.C. Guidoum, K. Boukhetala.

References

- Friedman, A. (1975). *Stochastic differential equations and applications*. Volume 1, ACADEMIC PRESS.
- Henderson, D. and Plaschko, P. (2006). *Stochastic differential equations in science and engineering*. World Scientific.
- Allen, E. (2007). *Modeling with Ito stochastic differential equations*. Springer-Verlag.
- Jedrzejewski, F. (2009). *Modeles aleatoires et physique probabiliste*. Springer-Verlag.
- Iacus, S.M. (2008). *Simulation and inference for stochastic differential equations: with R examples*. Springer-Verlag, New York.
- Kloeden, P.E, and Platen, E. (1989). A survey of numerical methods for stochastic differential equations. *Stochastic Hydrology and Hydraulics*, **3**, 155–178.
- Kloeden, P.E, and Platen, E. (1991a). Relations between multiple ito and stratonovich integrals. *Stochastic Analysis and Applications*, **9**(3), 311–321.
- Kloeden, P.E, and Platen, E. (1991b). Stratonovich and ito stochastic taylor expansions. *Mathematische Nachrichten*, **151**, 33–50.
- Kloeden, P.E, and Platen, E. (1995). *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, New York.
- Oksendal, B. (2000). *Stochastic Differential Equations: An Introduction with Applications*. 5th edn. Springer-Verlag, Berlin.
- Platen, E. (1980). Weak convergence of approximations of ito integral equations. *Z Angew Math Mech*. **60**, 609–614.
- Platen, E. and Bruti-Liberati, N. (2010). *Numerical Solution of Stochastic Differential Equations with Jumps in Finance*. Springer-Verlag, New York
- Saito, Y, and Mitsui, T. (1993). Simulation of Stochastic Differential Equations. *The Annals of the Institute of Statistical Mathematics*, **3**, 419–432.

See Also

[snssde1d](#) and [snssde2d](#) for 1- and 2-dim sde.
[sde.sim](#) in package "sde". [simulate](#) in package "yuima".

Examples

```
## Example 1: Ito sde
## dX(t) = 4*(-1-X(t))*Y(t) dt + 0.2 * dW1(t)
## dY(t) = 4*(1-Y(t)) *X(t) dt + 0.2 * dW2(t)
## dZ(t) = 4*(1-Z(t)) *Y(t) dt + 0.2 * dW3(t)
## W1(t), W2(t) and W3(t) three independent Brownian motion
set.seed(1234)

fx <- expression(4*(-1-x)*y)
gx <- expression(0.2)
fy <- expression(4*(1-y)*x)
gy <- expression(0.2)
```

```

fz <- expression(4*(1-z)*y)
gz <- expression(0.2)

res <- snssde3d(x0=2,y0=-2,z0=-2,driftx=fx,diffx=gx,drifty=fy,diffy=gy,
               driftz=fz,diffz=gz,N=1000,M=50)

res
summary(res)
dev.new()
plot(res,pos=2)
dev.new()
plot(res,union = FALSE)
dev.new()
plot3D(res,display="persp") ## in space (0,X,Y,Z)

## Example 2: Stratonovich sde
## dX(t) = Y(t)* dt
## dY(t) = (4*( 1-X(t)^2 )* Y(t) - X(t))* dt + 0.2 o dW2(t)
## dZ(t) = (4*( 1-X(t)^2 )* Z(t) - X(t))* dt + 0.2 o dW3(t)
set.seed(1234)

fx <- expression( y )
gx <- expression( 0 )
fy <- expression( (4*( 1-x^2 )* y - x) )
gy <- expression( 0.2 )
fz <- expression( (4*( 1-x^2 )* z - x) )
gz <- expression( 0.2 )

res <- snssde3d(driftx=fx,diffx=gx,drifty=fy,diffy=gy,driftz=fz,diffz=gz,
               ,N=5000,T=50,type="str")

res
dev.new()
plot(res,pos=2)
dev.new()
plot(res,union = FALSE)
dev.new()
plot3D(res,display="persp") ## in space (0,X,Y,Z)

```

st.int

Stochastic Integrals

Description

The (S3) generic function `st.int` of simulation of stochastic integrals of Ito or Stratonovich type.

Usage

```

st.int(expr, ...)
## Default S3 method:
st.int(expr, lower = 0, upper = 1, M = 1, subdivisions = 1000L,
       type = c("ito", "str"), ...)

```

```

## S3 method for class 'st.int'
summary(object, ...)
## S3 method for class 'st.int'
time(x, ...)
## S3 method for class 'st.int'
mean(x, ...)
## S3 method for class 'st.int'
median(x, ...)
## S3 method for class 'st.int'
quantile(x, ...)
## S3 method for class 'st.int'
kurtosis(x, ...)
## S3 method for class 'st.int'
skewness(x, ...)
## S3 method for class 'st.int'
moment(x, order = 2, ...)
## S3 method for class 'st.int'
bconfint(x, level=0.95, ...)
## S3 method for class 'st.int'
plot(x, ...)
## S3 method for class 'st.int'
lines(x, ...)
## S3 method for class 'st.int'
points(x, ...)

```

Arguments

expr	an expression of two variables t (time) and w (w: standard Brownian motion).
lower, upper	the lower and upper end points of the interval to be integrate.
M	number of trajectories.
subdivisions	the maximum number of subintervals.
type	Ito or Stratonovich integration.
x, object	an object inheriting from class "st.int".
order	order of moment.
level	the confidence level required.
...	further arguments for (non-default) methods.

Details

The function `st.int` returns a `ts` `x` of length $N+1$; i.e. simulation of stochastic integrals of Ito or Stratonovich type.

The Ito interpretation is:

$$\int_{t_0}^t f(s) dW_s = \lim_{N \rightarrow \infty} \sum_{i=1}^N f(t_{i-1})(W_{t_i} - W_{t_{i-1}})$$

The Stratonovich interpretation is:

$$\int_{t_0}^t f(s) \circ dW_s = \lim_{N \rightarrow \infty} \sum_{i=1}^N f\left(\frac{t_i + t_{i-1}}{2}\right) (W_{t_i} - W_{t_{i-1}})$$

For more details see vignette("SDEs").

Value

st.int returns an object inheriting from class "st.int".

X	the final simulation of the integral, an invisible <code>ts</code> object.
fun	function to be integrated.
type	type of stochastic integral.
subdivisions	the number of subintervals produced in the subdivision process.

Author(s)

A.C. Guidoum, K. Boukhetala.

References

Ito, K. (1944). Stochastic integral. *Proc. Jap. Acad, Tokyo*, **20**, 19–529.

Kloeden, P.E, and Platen, E. (1995). *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, New York.

Oksendal, B. (2000). *Stochastic Differential Equations: An Introduction with Applications*. 5th edn. Springer-Verlag, Berlin.

See Also

[snssde1d](#), [snssde2d](#) and [snssde3d](#) for 1,2 and 3-dim sde.

Examples

```
## Example 1: Ito integral
## f(t,w(t)) = int(exp(w(t) - 0.5*t) * dw(s)) with t in [0,1]
set.seed(1234)

fexpr <- expression( exp(w-0.5*t) )
res <- st.int(fexpr,type="ito",M=10,lower=0,upper=1)
res
## res$X
summary(res)
## Display
plot(res,plot.type="single")
lines(time(res),mean(res),col=2,lwd=2)
lines(time(res),bconfint(res,level=0.95)[,1],col=4,lwd=2)
lines(time(res),bconfint(res,level=0.95)[,2],col=4,lwd=2)
```

```
legend("topleft",c("mean path",paste("bound of", 95," confidence")),
      inset = .01,col=c(2,4),lwd=2,cex=0.8)

## Example 2: Stratonovich integral
## f(t,w(t)) = int(w(s) o dw(s)) with t in [0,1]
set.seed(1234)

fexpr <- expression( w )
res1 <- st.int(fexpr,type="str",M=10,lower=0,upper=1)
res1
## res1$X
summary(res1)
## Display
plot(res1,plot.type="single")
lines(time(res1),mean(res1),col=2,lwd=2)
lines(time(res1),bconfint(res1,level=0.95)[,1],col=4,lwd=2)
lines(time(res1),bconfint(res1,level=0.95)[,2],col=4,lwd=2)
legend("topleft",c("mean path",paste("bound of", 95," confidence")),
      inset = .01,col=c(2,4),lwd=2,cex=0.8)
```


Index

*Topic **BM**

BM, [12](#)

*Topic **datasets**

Irates, [37](#)

*Topic **fit**

fitsde, [23](#)

*Topic **fpt**

fptsde1d, [28](#)

fptsde2d, [30](#)

fptsde3d, [33](#)

*Topic **mts**

bridgesde1d, [15](#)

bridgesde2d, [17](#)

bridgesde3d, [20](#)

fptsde2d, [30](#)

fptsde3d, [33](#)

rsde1d, [40](#)

snssde1d, [42](#)

snssde2d, [45](#)

snssde3d, [49](#)

*Topic **package**

Sim.DiffProc-package, [2](#)

*Topic **random generators**

rsde1d, [40](#)

*Topic **sde**

BM, [12](#)

bridgesde1d, [15](#)

bridgesde2d, [17](#)

bridgesde3d, [20](#)

fitsde, [23](#)

fptsde1d, [28](#)

fptsde2d, [30](#)

fptsde3d, [33](#)

HWV, [35](#)

rsde1d, [40](#)

snssde1d, [42](#)

snssde2d, [45](#)

snssde3d, [49](#)

st.int, [53](#)

*Topic **ts**

BM, [12](#)

bridgesde1d, [15](#)

bridgesde2d, [17](#)

bridgesde3d, [20](#)

fitsde, [23](#)

fptsde1d, [28](#)

fptsde2d, [30](#)

fptsde3d, [33](#)

HWV, [35](#)

rsde1d, [40](#)

snssde1d, [42](#)

snssde2d, [45](#)

snssde3d, [49](#)

st.int, [53](#)

ABM (BM), [12](#)

AIC.fitsde (fitsde), [23](#)

Approx.fpt.density, [29](#), [32](#), [34](#)

BB (BM), [12](#)

BBridge, [14](#)

bconfint, [11](#)

bconfint.bridgesde1d (bridgesde1d), [15](#)

bconfint.bridgesde2d (bridgesde2d), [17](#)

bconfint.bridgesde3d (bridgesde3d), [20](#)

bconfint.snssde1d (snssde1d), [42](#)

bconfint.snssde2d (snssde2d), [45](#)

bconfint.snssde3d (snssde3d), [49](#)

bconfint.st.int (st.int), [53](#)

BIC.fitsde (fitsde), [23](#)

BM, [12](#), [14](#)

bridgesde1d, [15](#), [19](#), [22](#)

bridgesde2d, [17](#), [17](#), [39](#)

bridgesde3d, [17](#), [20](#), [39](#)

character, [24](#)

class, [16](#), [19](#), [22](#), [24](#), [29](#), [31](#), [34](#), [40](#), [43](#), [47](#),
[51](#), [55](#)

coef.fitsde (fitsde), [23](#)

- confint.fitsde (fitsde), 23
- DBridge, 17, 19, 22
- dcElerian, 25
- dcEuler, 25
- dcKessler, 25
- dcOzaki, 25
- dcShoji, 25
- dcSim, 25
- expression, 16, 18, 21, 23, 24, 28, 31, 33, 43, 46, 50, 54
- fitsde, 5, 23
- FPTL, 29, 32, 34
- fptsde1d, 7, 28, 28, 32, 34
- fptsde2d, 7, 29, 30, 31, 34
- fptsde3d, 7, 29, 32, 33, 33
- GBM, 14
- GBM (BM), 12
- gmm, 25
- HPloglik, 25
- HWV, 35
- Irates, 37, 38
- kurtosis (bconfint), 11
- kurtosis.bridgesde1d (bridgesde1d), 15
- kurtosis.bridgesde2d (bridgesde2d), 17
- kurtosis.bridgesde3d (bridgesde3d), 20
- kurtosis.snssde1d (snssde1d), 42
- kurtosis.snssde2d (snssde2d), 45
- kurtosis.snssde3d (snssde3d), 49
- kurtosis.st.int (st.int), 53
- lines.bridgesde1d (bridgesde1d), 15
- lines.bridgesde2d (bridgesde2d), 17
- lines.bridgesde3d (bridgesde3d), 20
- lines.snssde1d (snssde1d), 42
- lines.snssde2d (snssde2d), 45
- lines.snssde3d (snssde3d), 49
- lines.st.int (st.int), 53
- lines2d (plot2d), 38
- lines2d.bridgesde2d (bridgesde2d), 17
- lines2d.snssde2d (snssde2d), 45
- logLik.fitsde (fitsde), 23
- mean.bridgesde1d (bridgesde1d), 15
- mean.bridgesde2d (bridgesde2d), 17
- mean.bridgesde3d (bridgesde3d), 20
- mean.snssde1d (snssde1d), 42
- mean.snssde2d (snssde2d), 45
- mean.snssde3d (snssde3d), 49
- mean.st.int (st.int), 53
- median.bridgesde1d (bridgesde1d), 15
- median.bridgesde2d (bridgesde2d), 17
- median.bridgesde3d (bridgesde3d), 20
- median.snssde1d (snssde1d), 42
- median.snssde2d (snssde2d), 45
- median.snssde3d (snssde3d), 49
- median.st.int (st.int), 53
- missing, 13, 16, 18, 21, 36, 43, 46, 50
- moment (bconfint), 11
- moment.bridgesde1d (bridgesde1d), 15
- moment.bridgesde2d (bridgesde2d), 17
- moment.bridgesde3d (bridgesde3d), 20
- moment.snssde1d (snssde1d), 42
- moment.snssde2d (snssde2d), 45
- moment.snssde3d (snssde3d), 49
- moment.st.int (st.int), 53
- Normal, 13
- optim, 24
- OU (HWV), 35
- par, 39
- plot.bridgesde1d (bridgesde1d), 15
- plot.bridgesde2d (bridgesde2d), 17
- plot.bridgesde3d (bridgesde3d), 20
- plot.fptsde1d (fptsde1d), 28
- plot.fptsde2d (fptsde2d), 30
- plot.fptsde3d (fptsde3d), 33
- plot.snssde1d (snssde1d), 42
- plot.snssde2d (snssde2d), 45
- plot.snssde3d (snssde3d), 49
- plot.st.int (st.int), 53
- plot2d, 38
- plot2d.bridgesde2d (bridgesde2d), 17
- plot2d.snssde2d (snssde2d), 45
- plot3D (plot2d), 38
- plot3D.bridgesde3d (bridgesde3d), 20
- plot3D.snssde3d (snssde3d), 49
- points.bridgesde1d (bridgesde1d), 15
- points.bridgesde2d (bridgesde2d), 17
- points.bridgesde3d (bridgesde3d), 20
- points.snssde1d (snssde1d), 42

- points.snssde2d (snssde2d), 45
- points.snssde3d (snssde3d), 49
- points.st.int (st.int), 53
- points2d (plot2d), 38
- points2d.bridgesde2d (bridgesde2d), 17
- points2d.snssde2d (snssde2d), 45
- print.bridgesde1d (bridgesde1d), 15
- print.bridgesde2d (bridgesde2d), 17
- print.bridgesde3d (bridgesde3d), 20
- print.fitsde (fitsde), 23
- print.snssde1d (snssde1d), 42
- print.snssde2d (snssde2d), 45
- print.snssde3d (snssde3d), 49
- print.st.int (st.int), 53
- PSM.estimate, 25

- quantile.bridgesde1d (bridgesde1d), 15
- quantile.bridgesde2d (bridgesde2d), 17
- quantile.bridgesde3d (bridgesde3d), 20
- quantile.snssde1d (snssde1d), 42
- quantile.snssde2d (snssde2d), 45
- quantile.snssde3d (snssde3d), 49
- quantile.st.int (st.int), 53

- rcBS, 40
- rcCIR, 40
- rcOU, 36, 40
- rng, 40
- rsde1d, 7, 40
- rsde2d, 7
- rsde2d (rsde1d), 40
- rsde3d, 7
- rsde3d (rsde1d), 40
- rsOU, 36, 40

- sde.sim, 44, 48, 52
- Sim.DiffProc (Sim.DiffProc-package), 2
- Sim.DiffProc-package, 2
- skewness (bconfint), 11
- skewness.bridgesde1d (bridgesde1d), 15
- skewness.bridgesde2d (bridgesde2d), 17
- skewness.bridgesde3d (bridgesde3d), 20
- skewness.snssde1d (snssde1d), 42
- skewness.snssde2d (snssde2d), 45
- skewness.snssde3d (snssde3d), 49
- skewness.st.int (st.int), 53
- snssde1d, 4, 16, 28, 40, 42, 48, 52, 55
- snssde2d, 4, 18, 31, 39, 40, 44, 45, 52, 55
- snssde3d, 4, 21, 33, 39, 40, 44, 49, 55

- st.int, 3, 53
- summary.fitsde (fitsde), 23
- summary.fptsde1d (fptsde1d), 28
- summary.fptsde2d (fptsde2d), 30
- summary.fptsde3d (fptsde3d), 33
- summary.snssde1d (snssde1d), 42
- summary.snssde2d (snssde2d), 45
- summary.snssde3d (snssde3d), 49
- summary.st.int (st.int), 53

- time.bridgesde1d (bridgesde1d), 15
- time.bridgesde2d (bridgesde2d), 17
- time.bridgesde3d (bridgesde3d), 20
- time.snssde1d (snssde1d), 42
- time.snssde2d (snssde2d), 45
- time.snssde3d (snssde3d), 49
- time.st.int (st.int), 53
- ts, 16, 23, 43, 54, 55

- vcov.fitsde (fitsde), 23