Package: mcODE (via r-universe)

September 1, 2024

Type Package

Title Monte Carlo Solution of First Order Differential Equations

Version 1.1

Date 2024-07-01

Description Two functions for simulating the solution of initial value problems of the form g'(x) = G(x, g) with g(x0) = g0. One is an acceptance-rejection method. The other is a method based on the Mean Value Theorem.

LazyLoad true

License GPL (>= 2)

NeedsCompilation no

Author W.J. Braun [aut, cre]

Maintainer W.J. Braun <john.braun@ubc.ca>

Date/Publication 2024-07-02 15:20:02 UTC

Repository https://wjbraun.r-universe.dev

RemoteUrl https://github.com/cran/mcODE

RemoteRef HEAD

RemoteSha 38a4e200f58c592e740e9e7c2e3cba748b36bb61

Contents

mcODE-package *Monte Carlo Integration of First Order ODEs*

Description

Monte Carlo solution of first order differential equations and confidence intervals for the solutions.

Details

ODE.MVT can solve any first order differential equation of the form g' = F(x, g) with initial condition g(x0) = g0, provided conditions are satisfied for existence and uniqueness of the solution. Confidence intervals for the solution can be obtained through use of ODE.MVT.agg. An alternate method due to Akhtar et al is implemented in ODE.ADA.

Author(s)

Author: W.J. Braun

Maintainer: W.J. Braun

References

Akhtar, M. N., Durad, M. H., and Ahmed, A. (2015). Solving initial value ordinary differential equations by Monte Carlo method. Proc. IAM, 4:149-174.

Braun, W. J. (2024) Monte Carlo integration of first order ordinary differential equations. Preprint.

See Also

deSolve

Examples

```
# Solve g' = F(x, g) on (0, 1] with g(0) = -1/1000001
G <- function(x, g) {
    -1000*g + 3000 - 2000*exp(-x)
}
T <- 1
x0 <- 0
g0 <- -1/1000001
nMVT <- 5000
ghat <- ODE.MVT(G, initvalue = g0, endpoint = T, initpoint = x0, Niter = 2, npoints = nMVT)
plot(ghat, type = "l")
```

ODE.ADA *Monte Carlo ODE Solver of Akhtar et al*

Description

Given g' = G(x, g) and g(x0) = g0 satisfying conditions for existence and uniqueness of solution, a Monte Carlo approximation to the solution is found. The method is essentially a rejection method.

Usage

```
ODE.ADA(G, initvalue, endpoint, X0 = 0, npoints = 1000, M, R, N = 1e+05)
```

Arguments

G	a function having two numeric vector arguments.
initvalue	a numeric initial value
endpoint	a numeric interval endpoint value.
X0	a numeric interval starting point value.
npoints	an integer value specifying the number of subintervals to build the approximation on.
M	a numeric value specifying an upper bound for F(x, g).
R	a numeric value specifying a lower bound for F(x, g).
N	an integer value specifying the number of Monte Carlo simulations used for each subinterval.

Value

A list consisting of

x	a vector of length npoints, consisting of the x-coordinate of the solution.
y	a vector of length npoints, consisting of the y-coordinate of the solution, i.e. g(x).

References

Akhtar, M. N., Durad, M. H., and Ahmed, A. (2015). Solving initial value ordinary differential equations by Monte Carlo method. Proc. IAM, 4:149-174.

Examples

```
G <- function(x, g) {
    -1000*g + sin(x)
}
gTrue <- function(x) (1000*sin(x) - cos(x))/1000001
xF <- 1; x0 <- 0
g0 <- -1/1000001  # initial value
```

```
NAkhtar <- 800
nAkhtar <- 2500
MAkhtar <- 1e-3
RAkhtar <- 1e-6
gAkhtar <- ODE.ADA(G, initvalue = g0, endpoint = xF, X0 = x0,
    npoints = nAkhtar, M=MAkhtar, R = RAkhtar, N = NAkhtar)
plot(gAkhtar, type = "l")
```

ODE.MVT *Monte Carlo ODE Solver Based on Mean Value Theorem*

Description

Given g' = G(x, g) and g(x0) = g0 satisfying conditions for existence and uniqueness of solution, a Monte Carlo approximation to the solution is found. The method is essentially a Monte Carlo version of Euler and Runge-Kutta type methods.

Usage

```
ODE.MVT(G, initvalue, endpoint, initpoint = 0, Niter = 2, npoints = 1000)
```

Arguments

G	a function having two arguments.
initvalue	a numeric initial value
endpoint	a numeric interval endpoint value.
initpoint	a numeric interval starting point value.
Niter	an integer value specifying the number of iterations at each step.
npoints	an integer value specifying the number of subintervals to build the approximation on.

Value

A list consisting of

x	a vector of length npoints, consisting of the x-coordinate of the solution.
y	a vector of length npoints, consisting of the y-coordinate of the solution, i.e. g(x).

Author(s)

Braun, W.J.

References

Braun, W.J. (2024) Preprint.

Examples

```
# Initial Value Problem:
G <- function(x, g) {
    -1000*g + sin(x)
}
g0 <- -1/1000001; x0 <- 0 # initial condition
xF <- 1 # interval endpoint
nMVT <- 1000 # number of subintervals used
# Monte Carlo solution based on one iteration:
ghat1 <- ODE.MVT(G, initvalue = g0, endpoint = xF, initpoint = x0, Niter = 1, npoints = nMVT)
# Monte Carlo solution based on five iterations:
ghat5 <- ODE.MVT(G, initvalue = g0, endpoint = xF, initpoint = x0, Niter = 5, npoints = nMVT)
gTrue <- function(x) (1000*sin(x) - cos(x))/1000001 # true solution
oldpar <- par(mfrow=c(1,3))
curve(gTrue(x)) #
lines(ghat1, col = 2, lty = 2, ylab="g(x)")
plot(abs(gTrue(ghat1$x) - ghat1$y) ~ ghat1$x, xlab = "x",
    ylab = "Absolute Error", type = "l", ylim = c(0, 2e-6), main="1 Iteration")
plot(abs(gTrue(ghat5$x) - ghat5$y) ~ ghat5$x, xlab = "x",
    ylab = "Absolute Error", type = "l", ylim = c(0, 2e-6), main="5 Iterations")
par(oldpar)
```

`ODE.MVT.agg` *Monte Carlo ODE Solver Based on Mean Value Theorem - Replicated*

Description

Given g' = G(x, g) and g(x0) = g0 satisfying conditions for existence and uniqueness of solution, a Monte Carlo approximation to the solution is found. The method is essentially a Monte Carlo version of Euler and Runge-Kutta type methods. Repeated calls are made to the `ODE.MVT` function to obtain replicated estimates of the solution.

Usage

```
ODE.MVT.agg(G, initvalue, endpoint, initpoint = 0, Niter = 2, npoints = 1000, nsims = 10)
```

Arguments

`G`	a function having two numeric vector arguments.
`initvalue`	a numeric initial value
`endpoint`	a numeric interval endpoint value.
`initpoint`	a numeric interval starting point value.
`Niter`	an integer value specifying the number of iterations at each step.
`npoints`	an integer value specifying the number of subintervals to build the approximation on.
`nsims`	an integer value specifying the number of replicated solutions required.

Value

A list consisting of

x	a vector of length npoints, consisting of the x-coordinate of the solution.
y	a vector of length npoints, consisting of the average of replicated y-coordinates of the solution, i.e. g(x).
stdev	a vector of length npoints, consisting of the standard deviation estimates of the solution estimate at each point.

Author(s)

W.J. Braun

Examples

```
G <- function(x, g) {
    -1000*g + sin(x)
}
g0 <- -1/1000001; x0 <- 0 # initial condition
xF <- 1 # interval endpoint
nMVT <- 1000 # number of subintervals used
# 10 reps of Monte Carlo solution based on two iterations:
ghat2 <- ODE.MVT.agg(G, initvalue = g0, endpoint = xF, initpoint = x0,
    Niter = 2, npoints = nMVT, nsims = 10)
# 10 reps of Monte Carlo solution based on three iterations:
ghat3 <- ODE.MVT.agg(G, initvalue = g0, endpoint = xF, initpoint = x0,
    Niter = 3, npoints = nMVT, nsims = 10)
gTrue <- function(x) (1000*sin(x) - cos(x))/1000001 # true solution
oldpar <- par(mfrow=c(1,2))
curve(gTrue(x), from = 0.3, to = 0.3015) # graph of solution on small interval
lines(ghat2, col = 4, lty = 4, ylab="g(x)", lwd = 2)  # estimated MC solution - 2 iterations
lines(ghat2$y - 1.96*ghat2$stdev ~ ghat2$x, lty = 3, col = 4) # lower conf. limits
lines(ghat2$y + 1.96*ghat2$stdev ~ ghat2$x, lty = 3, col = 4) # upper conf. limits
curve(gTrue(x), from = 0.3, to = 0.3015) # graph of solution on small interval
lines(ghat3, col = 4, lty = 4, ylab="g(x)", lwd = 2)  # estimated MC solution - 3 iterations
lines(ghat3$y - 1.96*ghat3$stdev ~ ghat3$x, lty = 3, col = 4) # lower conf. limits
lines(ghat3$y + 1.96*ghat3$stdev ~ ghat3$x, lty = 3, col = 4) # upper conf. limits
par(oldpar)
```

Index