Package: mcODE (via r-universe)

November 22, 2026

Type Package

Title Monte Carlo Solution of First Order Differential Equations

Version 1.2

Date 2026-11-19

Description 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 others are based on the Mean Value Theorem, with options for antithetic variates, replicated solutions and confidence intervals, and systems of equations.

LazyLoad true

License GPL (>= 2)

NeedsCompilation no

Author W.J. Braun [aut, cre]

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

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

Date/Publication 2026-11-21 18:50:02 UTC

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

RemoteRef HEAD

RemoteSha ef9dc80504ed1b19ada7d6a1da3da67fd4283698

Contents

mcODE-package 2
ODE.ADA 3
ODE.MVT 4
ODE.MVT.agg 6
ODE.MVT.anti 7
ODE.MVT.sys 9

Index **11**

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. The method is based on the mean value theorem (MVT). ODE.MVT.anti adds antithetic variates, which reduce the random error by several orders of magnitude for non-stiff problems. ODE.MVT.sys solves systems of equations, such as the Lotka-Volterra and SIR models. 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. (2026). Monte Carlo integration of initial value problems. Preprint.

See Also

deSolve

Examples

```
# Solve g' = F(x, g) on (0, 1] with g(0) = 0
G <- function(x, g) {
    -1000*g + 3000 - 2000*exp(-x)
}
T <- 1
x0 <- 0
g0 <- 0
nMVT <- 5000
ghat <- ODE.MVT(G, initvalue = g0, endpoint = T, initpoint = x0, Niter = 2, npoints = nMVT)
plot(ghat, type = "l")
# exact solution
curve(3 - (997/999)*exp(-1000*x) - (2000/999)*exp(-x), add = TRUE, col = 2, lty = 2)
```

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 equally spaced mesh points (that is, npoints - 1 subintervals).
M	a numeric value specifying an upper bound for G(x, g); it must be at least the maximum of G along the solution.
R	a non-negative numeric value; -R is a lower bound for G(x, g).
N	an integer value specifying the number of uniform random variates. A single set of N variates is generated and re-used on every 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 of two numeric arguments, x and g.
initvalue	a numeric initial value
endpoint	a numeric interval endpoint value.
initpoint	a numeric interval starting point value.
Niter	an integer value M specifying the number of iterations, i.e. the number of nested uniform variates and evaluations of G, used on each subinterval. Niter = 1 gives an Euler step with the slope evaluated at a random point.
npoints	an integer value specifying the number of equally spaced mesh points (that is, npoints - 1 subintervals).

Details

If ξ is uniformly distributed on $[x_0, x]$, then $g(x_0) + (x - x_0)G(\xi, g(\xi))$ is an unbiased estimator of $g(x)$. The unknown $g(\xi)$ is approximated by applying the same argument recursively, with M = Niter nested uniform variates, starting from the crude prediction $g(x_0)$. The predictor is applied on each subinterval of an equally spaced mesh with spacing Δ, using a single realization per subinterval.

The bias per subinterval is of order Δ^{M+1}, with leading term $-\Delta^{M+1} G G_g^M/(M+1)!$ (where G_g is the partial derivative of G with respect to g), so the accumulated bias over a fixed interval is of order Δ^M. The bias is driven by ΔG_g: for stiff problems, where this is not small, more iterations or a finer mesh are needed. The accumulated random error is of order $\Delta^{3/2}$, regardless of M; for non-stiff problems, 2 or 3 iterations are therefore usually sufficient. The random error can be reduced with antithetic variates (ODE.MVT.anti), and replicated solutions give confidence bands (ODE.MVT.agg).

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. (2026). Monte Carlo integration of initial value problems. Preprint.

See Also

ODE.MVT.anti, ODE.MVT.agg, ODE.MVT.sys

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 (or ODE.MVT.anti) to obtain replicated estimates of the solution.

Usage

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

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 M specifying the number of iterations, i.e. the number of nested uniform variates and evaluations of G, used on each subinterval. Niter = 1 gives an Euler step with the slope evaluated at a random point.
npoints	an integer value specifying the number of equally spaced mesh points (that is, npoints - 1 subintervals).
nsims	an integer value specifying the number of replicated solutions required.
antithetic	logical; if TRUE, each replicate is computed with antithetic variates, using ODE.MVT.anti. The default, FALSE, uses ODE.MVT.

Details

Approximate pointwise $1 - \alpha$ confidence intervals are given by y $\pm z_{\alpha/2}$ stdev/sqrt(nsims). Strictly, these are confidence intervals for the expected value of the Monte Carlo solution; they are confidence intervals for $g(x)$ only to the extent that the bias of the solution is negligible compared with stdev/sqrt(nsims). The more replicates are used, the narrower the intervals, and the more iterations are needed to keep the bias small relative to their width, especially for stiff problems.

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 of the replicated solutions at each point; the standard error of y is stdev/sqrt(nsims).

Author(s)

W.J. Braun

References

Braun, W.J. (2026). Monte Carlo integration of initial value problems. Preprint.

See Also

ODE.MVT, ODE.MVT.anti

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/sqrt(10) ~ ghat2$x, lty = 3, col = 4) # lower conf. limits
lines(ghat2$y + 1.96*ghat2$stdev/sqrt(10) ~ ghat2$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/sqrt(10) ~ ghat3$x, lty = 3, col = 4) # lower conf. limits
lines(ghat3$y + 1.96*ghat3$stdev/sqrt(10) ~ ghat3$x, lty = 3, col = 4) # upper conf. limits
par(oldpar)
```

ODE.MVT.anti *Monte Carlo ODE Solver Based on Mean Value Theorem, with Antithetic Variates*

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 using the Niter-step mean value theorem (MVT) predictor of ODE.MVT, optionally with antithetic variates.

Usage

```
ODE.MVT.anti(G, initvalue, endpoint, initpoint = 0, Niter = 2, npoints = 1000,
    method = c("anti", "plain", "pair"))
```

Arguments

G	a function of two numeric arguments, x and g.
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, i.e. the number of nested uniform variates and evaluations of G, used by each predictor on each subinterval.
npoints	an integer value specifying the number of equally spaced mesh points (that is, npoints - 1 subintervals).
method	"anti" (default): each increment is the average of two predictors computed from fractions V and $1-V$; "plain": a single predictor, with the same distribution as ODE.MVT; "pair": the average of two predictors computed from independent fractions (for comparison with "anti").

Details

On each subinterval $[x_{j-1}, x_j]$ of length Δ, the nested uniform variates are written as $\xi^{[k]} - x_{j-1} = \Delta V_1 \cdots V_k$, $k = 1, \ldots, Niter$, where $V_1, \ldots, V_{Niter}$ are independent $U(0,1)$. The antithetic predictor averages the predictor computed from $V_1, \ldots, V_{Niter}$ with that computed from $1 - V_1, \ldots, 1 - V_{Niter}$.

Both predictors have the same distribution, so the antithetic predictor has the same bias as the plain one, at twice the number of evaluations of G. The leading term of the random error cancels, so that the variance of each step is reduced from order Δ^4 to order Δ^6, and the accumulated random error over a fixed interval from order $\Delta^{3/2}$ to order $\Delta^{5/2}$. Since the accumulated bias is of order Δ^{Niter}, at least 3 iterations are needed to benefit fully; for stiff problems more may be 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 y-coordinate of the solution, i.e. g(x).

Author(s)

Braun, W.J.

References

Braun, W.J. (2026). Monte Carlo integration of initial value problems. Preprint.

See Also

ODE.MVT, ODE.MVT.agg

Examples

```
# Riccati equation g' = 1 + x^2 - 2xg + g^2, g(0) = 1, with solution x - 1/(x - 1)
G <- function(x, g) 1 + x^2 - 2*x*g + g^2
gTrue <- function(x) x - 1/(x - 1)
ghat <- ODE.MVT.anti(G, initvalue = 1, endpoint = 0.5, initpoint = 0,
    Niter = 3, npoints = 1000, method = "plain")
ghatAnti <- ODE.MVT.anti(G, initvalue = 1, endpoint = 0.5, initpoint = 0,
    Niter = 3, npoints = 1000, method = "anti")
plot(ghat$x, log10(abs(ghat$y - gTrue(ghat$x)) + 1e-16), type = "l",
    ylim = c(-16, 0), xlab = "x", ylab = "log10 absolute error")
lines(ghatAnti$x, log10(abs(ghatAnti$y - gTrue(ghatAnti$x)) + 1e-16), col = 2)
legend("topleft", legend = c("plain", "antithetic"), col = 1:2, lty = 1, bty = "n")
```

ODE.MVT.sys *Monte Carlo Solver for Systems of ODEs Based on Mean Value Theorem*

Description

Given a system g' = G(x, g), where g is a vector, and g(x0) = g0 satisfying conditions for existence and uniqueness of solution, a Monte Carlo approximation to the solution is found using the mean value theorem (MVT) predictor of ODE.MVT.

Usage

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

Arguments

G	a function of two arguments, a numeric value x and a numeric vector g of the same length as initvalue, returning a numeric vector of that length.
initvalue	a numeric vector of initial values; its names, if any, are used as column names of the result.
endpoint	a numeric interval endpoint value.
initpoint	a numeric interval starting point value.
Niter	an integer value specifying the number of iterations, i.e. the number of nested uniform variates and evaluations of G, used on each subinterval.
npoints	an integer value specifying the number of equally spaced mesh points (that is, npoints - 1 subintervals).

Details

The method is that of ODE.MVT, applied without modification to a vector valued solution: on each subinterval, the same uniform variates are used for all components, and all components of each intermediate prediction are updated simultaneously from the same previous estimates. For a single equation, the result is identical to that of ODE.MVT with the same random number seed.

Higher order equations, and some integro-differential equations, can be solved by first rewriting them as first order systems.

Value

A list consisting of

x	a vector of length npoints, consisting of the x-coordinate of the solution.
y	a matrix with npoints rows and one column for each component of the solution.

Author(s)

Braun, W.J.

References

Braun, W.J. (2026). Monte Carlo integration of initial value problems. Preprint.

See Also

ODE.MVT

Examples

```
# Lotka-Volterra predator-prey model
LV <- function(t, g) {
    c(0.6*g[1] - 0.3*g[1]*g[2], -0.4*g[2] + 0.65*g[1]*g[2])
}
out <- ODE.MVT.sys(LV, initvalue = c(prey = 2, predator = 1), endpoint = 50,
    Niter = 2, npoints = 5000)
oldpar <- par(mfrow = c(1, 2))
matplot(out$x, out$y, type = "l", lty = 1, xlab = "t", ylab = "population")
plot(out$y, type = "l", main = "phase plane")
par(oldpar)

# SIR epidemic model
SIR <- function(t, g, beta = 0.5, gamma = 0.15, N = 1000) {
    c(-beta*g[1]*g[2]/N, beta*g[1]*g[2]/N - gamma*g[2], gamma*g[2])
}
out <- ODE.MVT.sys(SIR, initvalue = c(S = 990, I = 10, R = 0), endpoint = 52)
matplot(out$x, out$y, type = "l", lty = 1, xlab = "t", ylab = "number")
```

Index

∗ **datagen**
ODE.ADA, 3
ODE.MVT, 4
ODE.MVT.agg, 6
ODE.MVT.anti, 7
ODE.MVT.sys, 9
∗ **package**
mcODE-package, 2

mcODE *(*mcODE-package*)*, 2
mcODE-package, 2

ODE.ADA, *2*, 3
ODE.MVT, *2*, 4, *6–10*
ODE.MVT.agg, *2*, *5*, 6, *9*
ODE.MVT.anti, *2*, *5*, *6*, 7, *7*
ODE.MVT.sys, *2*, *5*, 9