Package 'PBSddesolve'

November 14, 2025

Version 1.13.7

Date 2025-11-12

Title Solver for Delay Differential Equations

Author Alex Couture-Beil [aut],
Jon T. Schnute [aut],
Rowan Haigh [aut],
Simon N. Wood [aut],
Benjamin J. Cairns [aut],
Nicholas Boers [ctb],
Nick Fisch [cre]

Maintainer Nick Fisch <nick.fisch@dfo-mpo.gc.ca>

Depends R (>= 4.2.0)

Suggests PBSmodelling

NeedsCompilation yes

Description Functions for solving systems of delay differential equations by interfacing with numerical routines written by Simon N. Wood, including contributions from Benjamin J. Cairns. These numerical routines first appeared in Simon Wood's 'solv95' program. This package includes a vignette and a complete user's guide. 'PBSddesolve' originally appeared on CRAN under the name 'ddesolve'. That version is no longer supported. The current name emphasizes a close association with other 'PBS' packages, particularly 'PBSmodelling'.

License GPL (>= 2)

URL https://github.com/pbs-software/pbs-ddesolve

Repository CRAN

Date/Publication 2025-11-14 12:50:17 UTC

Contents

---

dde *Solve Delay Differential Equations*

Description

A solver for systems of delay differential equations based on numerical routines from C source code `solv95` by Simon Wood. This solver is also capable of solving systems of ordinary differential equations.

Usage

```
dde(y, times, func, parms=NULL, switchfunc=NULL, mapfunc=NULL,
   tol=1e-08, dt=0.1, hbsize=10000)
```

Arguments

`y`	`numeric` – vector of initial values of the DDE system; the size of the supplied vector determines the number of variables in the system
`times`	`numeric` – vector of specific times to solve
`func`	`function` – a user-supplied function that computes the gradients in the DDE system at time `t`. The function must be defined using the arguments: `(t,y)` or `(t,y,parms)`, where `t` is the current time in the integration, `y` is a vector of the current estimated variables of the DDE system, and `parms` is any R object representing additional parameters (optional). The argument `func` must return one of the two following return types: 1) a vector containing the calculated gradients for each variable; or 2) a list with two elements - the first a vector of calculated gradients, the second a vector (possibly named) of values for a variable specified by the user at each point in the integration.
`parms`	`list` – any constant parameters to pass to `func`, `switchfunc`, and `mapfunc`
`switchfunc`	`function` – an optional function that is used to manipulate state values at given times. The switch function takes the arguments `(t,y)` or `(t,y,parms)` and must return a numeric vector. The size of the vector determines the number of switches used by the model. As values of `switchfunc` pass through zero (from positive to negative), a corresponding call to `mapfunc` is made, which can then modify any state value.
`mapfunc`	`function` – if `switchfunc` is defined, then a map function must also be supplied with arguments `(t, y, switch_id)` or `t, y, switch_id, parms)`, where `t` is the time, `y` are the current state values, `switch_id` is the index of the triggered switch, and `parms` are additional constant parameters.

tol	numeric – maximum error tolerated at each time step (as a proportion of the state variable concerned)
dt	numeric – maximum initial time step
hbsize	numeric – history buffer size required for solving DDEs

Details

Please see the included demos ('blowflies', 'cooling', 'icecream', 'lorenz') for examples of how to use dde.

The demos can be run two ways:

1. Using the package utils, run the command:
   demo(icecream, package="PBSddesolve", ask=FALSE)
2. Using the package PBSmodelling, run the commands:
   require(PBSmodelling); runDemos()

The latter produces a GUI that shows all demos available from locally installed packages. Choose PBSddesolve. Note that the examples are run in the temporary working environment .PBSddeEnv.

The user supplied function func can access past values (lags) of y by calling the pastvalue function. Past gradients are accessible by the pastgradient function. These functions can only be called from func and can only be passed values of t greater or equal to the start time, but less than the current time of the integration point. For example, calling pastvalue(t) is not allowed, since these values are the current values which are passed in as y.

Value

A data frame with one column for t, a column for every variable in the system, and a column for every additional value that may (or may not) have been returned by func in the second element of the list.

If the initial y values parameter was named, then the solved values column will use the same names. Otherwise y1, y2, ... will be used.

If func returned a list, with a named vector as the second element, then those names will be used as the column names. If the vector was not named, then extra1, extra2, ... will be used.

Author(s)

Alex Couture-Beil – Software Engineer, Earthly Technologies, Victoria BC

Maintainer: Rowan Haigh, Program Head – Offshore Rockfish
Pacific Biological Station (PBS), Fisheries & Oceans Canada (DFO), Nanaimo BC
locus opus: remote office, Vancouver BC
Last modified Rd: 2025-06-13

See Also

In package **PBSddesolve**:
pastvalue

Examples

```
################################################
## This is just a single example of using dde.
## For more examples see demo(package="PBSddesolve")
## the demos require the package PBSmodelling
################################################

require(PBSddesolve)
local(env=.PBSddeEnv, expr={
  #create a func to return dde gradient
  yprime <- function(t,y,parms) {
    if (t < parms$tau)
      lag <- parms$initial
    else
      lag <- pastvalue(t - parms$tau)
    y1 <- parms$a * y[1] - (y[1]^3/3) + parms$m * (lag[1] - y[1])
    y2 <- y[1] - y[2]
    return(c(y1,y2))
  }

  #define initial values and parameters
  yinit <- c(1,1)
  parms <- list(tau=3, a=2, m=-10, initial=yinit)

  # solve the dde system
  yout <- dde(y=yinit,times=seq(0,30,0.1),func=yprime,parms=parms)

  # and display the results
  plot(yout$time, yout$y1, type="l", col="red", xlab="t", ylab="y",
    ylim=c(min(yout$y1, yout$y2), max(yout$y1, yout$y2)))
  lines(yout$time, yout$y2, col="blue")
  legend("topleft", legend = c("y1", "y2"),lwd=2, lty = 1,
    xjust = 1, yjust = 1, col = c("red","blue"))
})
```

---

dot-onClosePBSddeExamples

On Close Set Old WD

Description

A trivial function that sets the user's working directory to an old (previous) location before opening the Windows GUI that is now being closed.

Usage

```
.onClosePBSddeExamples()
```

Value

```
setwd(.PBSddeEnv$oldwd)
```

Author(s)

Alex Couture-Beil – Software Engineer, Earthly Technologies, Victoria BC

Maintainer: Rowan Haigh, Program Head – Offshore Rockfish
Pacific Biological Station (PBS), Fisheries & Oceans Canada (DFO), Nanaimo BC
locus opus: remote office, Vancouver BC
Last modified `Rd: 2025-06-13`

---

dot-PBSddeEnv *PBSddesolve Environment*

Description

An environment set aside for `PBSddesolve`.

Usage

```
.PBSddeEnv
```

Format

A new environment with a `.GlobalEnv` parent.

Details

The environment is created in `'zzz.r'` and can be used by `PBSmodelling` functions `'lisp'`, `'tget'`, `'tput'`, `'tprint'`, and `'tcall'`.

Source

Generated by a call to the base function `new.env()`.

See Also

In **PBSmodelling**:
`lisp`, `tget`

pastvalue *Retrieve Past Values (lags) During Gradient Calculation*

Description

These routines provides access to variable history at lagged times. The lagged time t must not be less than t_0, nor should it be greater than the current time of gradient calculation. The routine cannot be directly called by a user, and will only work during the integration process as triggered by the dde routine.

Usage

```
pastvalue(t)
pastgradient(t)
```

Arguments

t	numeric – time *t* at which history is accessed

Value

Vector of variable history at time *t*.

Author(s)

Alex Couture-Beil – Software Engineer, Earthly Technologies, Victoria BC

Maintainer: Rowan Haigh, Program Head – Offshore Rockfish
Pacific Biological Station (PBS), Fisheries & Oceans Canada (DFO), Nanaimo BC
locus opus: remote office, Vancouver BC
Last modified Rd: 2025-06-13

See Also

In package **PBSddesolve**:
dde

PBSddesolve *Package: Solver for Delay Differential Equations*

Description

A solver for systems of delay differential equations based on numerical routines from Simon Wood's programme. This solver is also capable of solving systems of ordinary differential equations.

Details

Please see the user guide `PBSddesolve-UG.pdf`, located in R's library directory `./library/PBSddesolve/doc`, for a comprehensive overview.

Author(s)

Simon Wood – Chair of Computational Statistics, School of Mathematics, University of Edinburgh
Alex Couture-Beil – Software Engineer, Earthly Technologies, Victoria BC
Jon T. Schnute – Scientist Emeritus, DFO, Nanaimo BC
Nicholas M. Boers – Senior Software Engineer, Jobber, Edmonton AB

Maintainer: Rowan Haigh, Program Head – Offshore Rockfish
Pacific Biological Station (PBS), Fisheries & Oceans Canada (DFO), Nanaimo BC
locus opus: remote office, Vancouver BC
Last modified `Rd: 2025-06-13`

References

Wood, S.N. (1999) Solv95: a numerical solver for systems of delay differential equations with switches. Mathematical Institute, North Haugh, St. Andrews, Fife KY16 9SS, U.K., 10 p.

See Also

In package **PBSddesolve**:
`dde`
In package **deSolve**:
`lsoda`

Index