

Package ‘PBSddesolve’

January 2, 2012

Version 1.08.11

Date 2010-08-27

Title Solver for Delay Differential Equations

Author Alex Couture-Beil <alex_rproject@mofo.ca>, Jon Schnute
<Jon.Schnute@dfo-mpo.gc.ca>, Rowan Haigh <Rowan.Haigh@dfo-mpo.gc.ca>

Maintainer Jon Schnute <Jon.Schnute@dfo-mpo.gc.ca>

Depends R (>= 2.6.0)

Description This package solves systems of delay differential equations. by interfacing numerical routines written by Simon N. Wood <s.wood_at_bath.ac.uk>, with contributions by Benjamin J. Cairns <ben.cairns@bristol.ac.uk>. 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)

Repository CRAN

Date/Publication 2010-08-28 06:19:36

R topics documented:

dde	2
pastvalue	4
PBSddesolve	5

Index	6
--------------	----------

Description

A solver for systems of delay differential equations based off numerical routines from Simon Wood's *sol95* program. This solver is also capable of solving systems of ordinary differential equations.

Please see the included demos for examples of how to use dde.

To view available demos run `demo(package="PBSddesolve")`.

The supplied demos require that the package **PBSmodelling** be installed.

Usage

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

Arguments

y	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	<p>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).</p> <p>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.</p>
parms	any constant parameters to pass to func, switchfunc, and mapfunc.
switchfunc	<p>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.</p>
mapfunc	<p>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.</p>

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

Details

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.

See Also

[pastvalue](#)

Examples

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

#create a func to return dde gradient
require(PBSdresolve)
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$t, yout$y1, type="l", col="red", xlab="t", ylab="y",
      ylim=c(min(yout$y1, yout$y2), max(yout$y1, yout$y2)))
lines(yout$t, yout$y2, col="blue")
legend("topleft", legend = c("y1", "y2"),lwd=2, lty = 1,
       xjust = 1, yjust = 1, col = c("red","blue"))
```

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 access history at time t .

Value

vector of variable history at time t .

See Also

[dde](#)

PBSddesolve

Package: Solver for Delay Differential Equations

Description

A solver for systems of delay differential equations based off numerical routines from Simon Wood's solv95 program. This solver is also capable of solving systems of ordinary differential equations.

Details

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

Author(s)

Alex Couture-Beil <alex_rproject@mofoc.ca>
Jon T. Schnute <Jon.Schnute@dfo-mpo.gc.ca>
Rowan Haigh <Rowan.Haigh@dfo-mpo.gc.ca>
Maintainer: Jon Schnute <Jon.Schnute@dfo-mpo.gc.ca>

References

Wood, S.N. (1999) Solv95: a numerical solver for systems of delay differential equations with switches. Saint Andrews, UK. 10 pp. URL: <http://www.maths.bath.ac.uk/~sw283/simon/dde.html>

See Also

[dde](#)

Index

*Topic **math**

dde, [2](#)

pastvalue, [4](#)

*Topic **package**

PBSddesolve, [5](#)

dde, [2](#), [4](#), [5](#)

pastgradient, [3](#)

pastgradient (pastvalue), [4](#)

pastvalue, [3](#), [4](#)

PBSddesolve, [5](#)

PBSddesolve-package (PBSddesolve), [5](#)