Package ‘PBSddesolve’

February 1, 2023

Version 1.13.3
Date 2023-02-01
Title Solver for Delay Differential Equations
Author Alex Couture-Beil [aut],
   Jon T. Schnute [aut],
   Rowan Haigh [aut, cre],
   Simon N. Wood [aut],
   Benjamin J. Cairns [aut],
   Nicholas Boers [ctb]
Maintainer Rowan Haigh <rowan.haigh@dfo-mpo.gc.ca>
Copyright 2007-2023, Fisheries and Oceans Canada
Depends R (>= 3.5.0)
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 'ndesolve'. 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 2023-02-01 17:00:05 UTC

R topics documented:

   dde ................................................................. 2
   pastvalue ...................................................... 4
   PBSddesolve .................................................. 5
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 \( (t,y,\text{parms}) \), where \( t \) is the current time in the integration, \( y \) is a vector of the current estimated variables of the DDE system, and \( \text{parms} \) is any R object representing additional parameters (optional).

The argument \( \text{func} \) 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 \( \text{func} \), \( \text{switchfunc} \), and \( \text{mapfunc} \).
switchfunc function – an optional function that is used to manipulate state values at given times. The switch function takes the arguments \( (t,y) \) or \( (t,y,\text{parms}) \) and must return a numeric vector. The size of the vector determines the number of switches used by the model. As values of \( \text{switchfunc} \) pass through zero (from positive to negative), a corresponding call to \( \text{mapfunc} \) is made, which can then modify any state value.

mapfunc function – if \( \text{switchfunc} \) is defined, then a map function must also be supplied with arguments \( (t,y,\text{switch}_\text{id}) \) or \( t, y, \text{switch}_\text{id}, \text{parms} \), where \( t \) is the time, \( y \) are the current state values, \( \text{switch}_\text{id} \) is the index of the triggered switch, and \( \text{parms} \) 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: Regional Headquarters (RHQ), Vancouver BC

Last modified Rd: 2023-01-30

See Also

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)
    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"))
})

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 $t$.

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: Regional Headquarters (RHQ), Vancouver BC
Last modified Rd: 2023-01-30

See Also

dde
Index

* delay differential
  dde, 2
* math
  dde, 2
  pastvalue, 4
* package
  PBSddesolve, 5

dde, 2, 5
pastgradient, 3
pastgradient (pastvalue), 4
pastvalue, 3, 4
PBSddesolve, 5
PBSddesolve-package (PBSddesolve), 5