Package ‘insideRODE’

February 20, 2015

Type Package
Title insideRODE includes buildin functions with deSolve solver and C/FORTRAN interfaces to nlme, together with compiled codes.
Version 2.0
Date 2011-04-19
Author YUZHUO PAN, XIAOYU YAN
Maintainer YUZHUO PAN <yuzhuo.pan@gmail.com>
Depends R (>= 2.13.0), deSolve, nlme, lattice, compiler
Description insideRODE package includes buildin functions from deSolve, compiled functions from compiler, and C/FORTRAN code interfaces to nlme. It includes nlmLSODA, nlmODE, nlmVODE, nlmLSODE for general purpose; cfLSODA, cfLSODE, cfODE, cfVODE call C/FORTRAN compiled dll functions. ver2.0 add sink() function into example it helps to directly combine c/fortran source code in R files. Finally, with new compiler package, we generated compiled functions: nlmODEcp, nlmVODEcp, nlmLSODEcp, nlmLSODAcp and cpODE, cpLSODA, cpLSODE, cpVODE. They will help to increase speed.
License LGPL (> 2.0)
LazyLoad yes
Repository CRAN
Date/Publication 2012-10-29 08:59:01
NeedsCompilation no

R topics documented:

insideRODE-package ............................................... 2
cfLSODA .......................................................... 3
cfLSODE .......................................................... 5
cfODE ............................................................. 6
cfVODE ............................................................ 7
cpLSODA ............................................................ 8
insideRODE-package

Description

insideRODE build-in function with Ordinary Differential Equation solver and C/FORTRAN inter-
face to nlme, including nlmLSODA, nlmODE, nlmVODE, nlmLSODE for general ODE; cfLSODA, 
cfLSODE, cfODE, cfVODE solver for C/FORTRAN based ODE. WE USE SEPERATED FILE 
TO GENERATE FUNCTIONS. V1.0 can read dllname from dynload, sent them to cf FUNC-
TIONS. insideRODE package also includes buildin functions from deSolve, compiled functions 
from compiler, and C/FORTRAN code interfaces to nlme. It includes nlmLSODA, nlmODE, 
nlmVODE, nlmLSODE for general purpose; cfLSODA, cfLSODE, cfODE, cfVODE call C/FORTRAN 
compiled dll functions. Ver2.0 add sink()function into example it helps to directly combine c/fortran 
source code in R files. Finally, with new compiler package, we generated compiled functions: 
nlmODEcp, nlmVODEcp, nlmLSODEcp, nlmLSODAcp and cpODE, cpLSODA, cpLSODE, cpVODE. 
They will help to increase speed. This package depends on the package of nlmODE from Christof-
fer W. Tornoe. This package updated the package from ODESOLVE into deSolve package, and 
implement the interface to c/fortran code. It will greatly enhance the performance of R and nlme.

Details

<table>
<thead>
<tr>
<th>Package</th>
<th>insideRODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Package</td>
</tr>
<tr>
<td>Version</td>
<td>2.0</td>
</tr>
<tr>
<td>Date</td>
<td>2011-04-19</td>
</tr>
<tr>
<td>License</td>
<td>LGPL(&gt;2.0)</td>
</tr>
<tr>
<td>LazyLoad</td>
<td>yes</td>
</tr>
</tbody>
</table>
cflsoda

Author(s)

YUZHUO PAN, XIAOYU YAN

Maintainer: YUZHUO PAN <yuzhuo.pan@gmail.com>

See Also

nlme, nlmeODE, deSolve, lattice, compiler

Examples

## Not run:
## show examples
demo("testfile") # differential equations

## End(Not run)

---

cflsoda LSODA Solver for NLME using compiled code(c or fortran)

Description

Use Solver for Ordinary Differential Equations (ODE), Switching Automatically Between Stiff and Non-stiff Methods and Generate functions to be used in NLME

Usage

cflsoda(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL)
Arguments

model  either an R-function that computes the values of the derivatives in the ODE system (the model definition) at time t. The return value of model should be a list. See package "nlmeODE" for more details.

data  nlme GroupedData format.
LogParms  transform parameters into log scale
JAC  A JAC set FALSE. This time we can implement this parts.
SEQ  A SEQ set FALSE.
rtol  relative error tolerance, either a scalar or an array as long as y. See details.
atol  absolute error tolerance, either a scalar or an array as long as y. See details.
tcrit  if not NULL, then lsoda cannot integrate past tcrit. The FORTRAN routine lsoda overshoots its targets (times points in the vector times), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in tcrit.
dllname  a string giving the name of the shared library (without extension) that contains all the compiled function or subroutine definitions referred to in func and jacobfunc. See package "deSolve".
hmin  an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use hmin if you don’t know why!
hmax  an optional maximum value of the integration stepsize. If not specified, hmax is set to the largest difference in times, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.

Examples

# use c code
#cFLSODA SOLVER
rm(list=ls())
require(insideRODE)
data(Theoph)# examples from nlmeODE
TheophODE <- Theoph
TheophODE$Dose[TheophODE$Time!=0] <- 0
TheophODE$Cmt <- rep(1,dim(TheophODE)[1])

# model files
OneComp <- list(DiffEq=list(  dy1dt = -ka*y1,  dy2dt = -ka*y1-ke*y2),  ObsEq=list(  c1 = 0,  c2 = - y2/CL*ke), Parms=c("ka","ke","CL")),


**cfLSODE**

```r
States=c("y1","y2"),
Init=list(0,0))
```

TheophModel <- nlmLSODA(OneComp,TheophODE) #ode solver

---

**cfLSODE**

**LSODE Solver for NLME using compiled code(c or fortran)**

---

**Description**

Use Solver for Ordinary Differential Equations (ODE), Switching Automatically Between Stiff and Non-stiff Methods and Generate functions to be used in NLME

**Usage**

```r
cfLSODE(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL, dllname = NULL, hmin = NULL, hmax = NULL)
```

**Arguments**

- `model` either an R-function that computes the values of the derivatives in the ODE system (the *model definition*) at time t. The return value of `model` should be a list. See package "nlmeODE" for more details.
- `data` nlme GroupedData format.
- `LogParms` transform parameters into log scale.
- `JAC` A JAC set FALSE. This time we can implement this parts.
- `SEQ` A SEQ set FALSE.
- `rtol` relative error tolerance, either a scalar or an array as long as y. See details.
- `atol` absolute error tolerance, either a scalar or an array as long as y. See details.
- `tcrit` if not NULL, then lsoda cannot integrate past tcrit. The FORTRAN routine lsoda overshoots its targets (times points in the vector times), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in tcrit.
- `dllname` a string giving the name of the shared library (without extension) that contains all the compiled function or subroutine definitions refered to in `func` and `jacfunc`. See package "deSolve".
- `hmin` an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use `hmin` if you don’t know why!
- `hmax` an optional maximum value of the integration stepsize. If not specified, hmax is set to the largest difference in times, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.
Examples

```c
use c code
#see example cfLSODA
```

---

**cfODE**  
*General Solver for Ordinary Differential Equations and compiled functions to be used in NLME, this version just use default function. The future version will provide buildin methods such as “lsoda”, “lsode”, “lsodes”, “vode”, “daspk”, “euler”, “rk4”, “ode23”, “ode45”, “radau”, “bdf”, “bdf_d”, “adams”, “impAdams”, “impAdams_d”.*

---

**Description**

Generate functions for NLME Solves using a system of ordinary differential equations; a wrapper around the implemented ODE solvers

**Usage**

```r
cfODE(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL, dllname = NULL)
```

**Arguments**

- **model**
  - either an *R*-function that computes the values of the derivatives in the ODE system (the *model definition*) at time *t*. The return value of `model` should be a list. See package "n1meODE" for more details.

- **data**
  - nlme GroupedData format.

- **LogParms**
  - transform parameters into log scale

- **JAC**
  - A JAC set FALSE. This time we can implement this parts.

- **SEQ**
  - A SEQ set FALSE.

- **rtol**
  - relative error tolerance, either a scalar or an array as long as y. See details.

- **atol**
  - absolute error tolerance, either a scalar or an array as long as y. See details.

- **tcrit**
  - if not NULL, then lsoda cannot integrate past tcrit. The FORTRAN routine lsoda overshoots its targets (times points in the vector times), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in tcrit.

- **dllname**
  - a string giving the name of the shared library (without extension) that contains all the compiled function or subroutine definitions refered to in func and jactest. See package "deSolve".
an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use hmin if you don’t know why!

hmax
an optional maximum value of the integration stepsize. If not specified, hmax is set to the largest difference in times, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.

Examples

#use c code
#see example cfLSODA

---

cfvode

Solver for Ordinary Differential Equations (VODE) and compiled functions(c/fortran) to be used in NLME

Description

Generate functions for NLME Solves using a system of ordinary differential equations; a wrapper around the implemented ODE solvers. The R function vode provides an interface to the FORTRAN ODE solver of the same name, written by Peter N. Brown, Alan C. Hindmarsh and George D. Byrne.

Usage

cfvode(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL, dllname = nullL, hmin = NULL, hmax = inf)

Arguments

model
either an R-function that computes the values of the derivatives in the ODE system (the model definition) at time t. The return value of model should be a list. See package "nlmeODE" for more details.

data
nlme GroupedData format.

LogParms
transform parameters into log scale

JAC
A JAC set FALSE. This time we can implement this parts.

SEQ
A SEQ set FALSE.

rtol
relative error tolerance, either a scalar or an array as long as y. See details.

atol
absolute error tolerance, either a scalar or an array as long as y. See details.

tcrit
if not NULL, then lsoda cannot integrate past tcrit. The FORTRAN routine lsoda overshoots its targets (times points in the vector times), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in tcrit.
dllname 

a string giving the name of the shared library (without extension) that contains all the compiled function or subroutine definitions referred to in func and jacfunc. See package "deSolve".

hmin 

an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use hmin if you don’t know why!

hmax 

an optional maximum value of the integration stepsize. If not specified, hmax is set to the largest difference in times, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.

Examples

*****************************************************************************
use c code
see example cfLSODA
*****************************************************************************

### cpLSODA

**LSODA Solver for NLME using compiled code(c or fortran)**

**Description**

Use Solver for Ordinary Differential Equations (ODE), Switching Automatically Between Stiff and Non-stiff Methods and Generate functions to be used in NLME

**Usage**

```r
cpLSODA(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL)
```

**Arguments**

- `model` 
  
either an *R*-function that computes the values of the derivatives in the ODE system (the *model definition*) at time *t*. The return value of `model` should be a list. See package "nlmeODE" for more details.

- `data` 
  
nlme GroupedData format.

- `LogParms` 
  
transform parameters into log scale

- `JAC` 
  
A JAC set FALSE. This time we can implement this parts.

- `SEQ` 
  
A SEQ set FALSE.

- `rtol` 
  
relative error tolerance, either a scalar or an array as long as `y`. See details.

- `atol` 
  
absolute error tolerance, either a scalar or an array as long as `y`. See details.

- `tcrit` 
  
if not NULL, then lsoda cannot integrate past `tcrit`. The FORTRAN routine lsoda overshoots its targets (times points in the vector `times`), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in `tcrit`. 

dllname    a string giving the name of the shared library (without extension) that contains all the compiled function or subroutine definitions referred to in func and jacfunc. See package “deSolve”.

hmin      an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use hmin if you don’t know why!

hmax      an optional maximum value of the integration stepsize. If not specified, hmax is set to the largest difference in times, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.

Examples

```
#use c code
#cplSODA SOLVER

rm(list=ls())
require(insideRODE)

data(Theoph)# examples from nlmODE
TheophODE <- Theoph
TheophODE$tDose[TheophODE$tTime!=0] <- 0
TheophODE$tCmt <- rep(1,dim(TheophODE)[1])

# model files
OneComp <- list(DiffEq=list(  
  dy1dt = -ka*y1,  
  dy2dt = -ka*y1-ke*y2),
ObsEq=list(  
  c1 = ~ 0,  
  c2 = ~ y2/CL*ke),
Parms=c("ka","ke","CL"),
States=c("y1","y2"),
Init=list(0,0))

TheophModel <- nlmLSODA(OneComp,TheophODE) #ode solver
```

#example
#sink functions
#cplSODA

```
#sink("mymod.c")
cat(""
/* file mymod.c */
#include <R.h>
#include <math.h>
static double parms[3];
#define ka parms[0]
#define ke parms[1]
#define CL parms[2]
```
/* initializer */
void initmod(void (*odeparms)(int *, double *))
{
    int N=2;
    odeparms(&N, parms);
}

/* names for states and derivatives */
#define y1 y[0]
#define y2 y[1]
#define dy1 ydot[0]
#define dy2 ydot[1]
#define c1 yout[0]
#define c2 yout[1]

/* Derivatives and 1 output variable */
void derivs (int *neq, double *t, double *y, double *ydot, double *yout, int *ip)
{
    dy1 = -exp(ka)*y1;
    dy2 = exp(ka)*y1-exp(ke)*y2;
    c1 = 0.0;
    c2 = y2/exp(CL)*exp(ke);
}

/* END file mymod1.c */

",.fill=TRUE)
#sink()
#system("RCMD SHLIB mymod.c")
#dllname<~dyn.load("mymod.dll")[[1]]

#cplsodaLcflsodeL cplsodaL cfvode solver
#cplsodaLcplSODA, cflSODA, cplSODA, cfVODE SOLVER
#sink("mymodff.f")
cat"
c file mymodf.f
   subroutine initmod(odeparms)
external odeparms
double precision parms(3)
common /myparms/parms
call odeparms(2, parms)
return
end
   subroutine derivs (neq, t, y, ydot, yout, ip)
double precision t, y, ydot, ka, ke, CL
integer neq, ip(*)
dimension y(2), ydot(2), yout(2)
common /myparms/ka,ke,CL
ydot(1) = -exp(ka)*y(1)
ydot(2) = exp(ka)*y(1)-exp(ke)*y(2)
yout(1) = 0
yout(2) = y(2)/exp(CL)*exp(ke)
Use Solver for Ordinary Differential Equations (ODE), Switching Automatically Between Stiff and Non-stiff Methods and Generate functions to be used in NLME

Arguments

- **model**: either an \textit{R}-function that computes the values of the derivatives in the ODE system (the \textit{model definition}) at time \( t \). The return value of \texttt{model} should be a list. See package "nlmeODE" for more details.
- **data**: \textit{nlme GroupedData} format.
- **LogParms**: transform parameters into log scale
- **JAC**: A JAC set \texttt{FALSE}. This time we can implement this parts.
- **SEQ**: A SEQ set \texttt{FALSE}.
- **rtol**: relative error tolerance, either a scalar or an array as long as \( y \). See details.
- **atol**: absolute error tolerance, either a scalar or an array as long as \( y \). See details.
- **tcrit**: if not \texttt{NULL}, then \texttt{lsoda} cannot integrate past \texttt{tcrit}. The FORTRAN routine \texttt{lsoda} overshoots its targets (times points in the vector \texttt{times}), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in \texttt{tcrit}.
- **dllname**: a string giving the name of the shared library (without extension) that contains all the compiled function or subroutine definitions refered to in \texttt{func} and \texttt{jacfunc}. See package "deSolve".
- **hmin**: an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use \texttt{hmin} if you don’t know why!
- **hmax**: an optional maximum value of the integration stepsize. If not specified, \texttt{hmax} is set to the largest difference in \texttt{times}, to avoid that the simulation possibly ignores short-term events. If \texttt{0}, no maximal size is specified.
Examples

# use c code
# see cpLSODA

General Solver for Ordinary Differential Equations and compiled functions to be used in NLME, this version just use default function. The future version will provide buildin methods such as "lsoda", "lsode", "lsodes", "lsodar", "vode", "daspk", "euler", "rk4", "ode23", "ode45", "radua", "bdf", "bdf_d", "adams", "impAdams", "impAdams_d".

Description

Generate functions for NLME Solves using a system of ordinary differential equations; a wrapper around the implemented ODE solvers

Usage

cpODE(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL, dllname = NULL, hmin = NULL)

Arguments

model either an R-function that computes the values of the derivatives in the ODE system (the model definition) at time t. The return value of model should be a list. See package "nlmeODE" for more details.
data nlme GroupedData format.
LogParms transform parameters into log scale
JAC A JAC set FALSE. This time we can implement this parts.
SEQ A SEQ set FALSE.
rtol relative error tolerance, either a scalar or an array as long as y. See details.
atol absolute error tolerance, either a scalar or an array as long as y. See details.
tcrit if not NULL, then lsoda cannot integrate past tcrit. The FORTRAN routine lsoda overshoots its targets (times points in the vector times), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in tcrit.
dllname a string giving the name of the shared library (without extension) that contains all the compiled function or subroutine definitions refered to in func and jacfunc. See package "deSolve".

hmin an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use hmin if you don’t know why!
hmax

an optional maximum value of the integration stepsize. If not specified, hmax is set to the largest difference in times, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.

Examples

# use c code
# see cplSODA

Description

Generate functions for NLME Solves using a system of ordinary differential equations; a wrapper around the implemented ODE solvers. The \texttt{R} function \texttt{vode} provides an interface to the \texttt{FORTRAN} ODE solver of the same name, written by Peter N. Brown, Alan C. Hindmarsh and George D. Byrne.

Usage

\texttt{cpVODE(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL, dllname = NULL, hmin = NULL, hmax = NULL)}

Arguments

- **model**: either an \texttt{R}-function that computes the values of the derivatives in the ODE system (the model definition) at time \texttt{t}. The return value of \texttt{model} should be a list. See package "nlmeODE" for more details.
- **data**: nlme GroupedData format.
- **LogParms**: transform parameters into log scale.
- **JAC**: A JAC set FALSE. This time we can implement this parts.
- **SEQ**: A SEQ set FALSE.
- **rtol**: relative error tolerance, either a scalar or an array as long as \texttt{y}. See details.
- **atol**: absolute error tolerance, either a scalar or an array as long as \texttt{y}. See details.
- **tcrit**: if not \texttt{NULL}, then \texttt{lsoda} cannot integrate past \texttt{tcrit}. The \texttt{FORTRAN} routine \texttt{lsoda} overshoots its targets (times points in the vector \texttt{times}), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in \texttt{tcrit}.
- **dllname**: a string giving the name of the shared library (without extension) that contains all the compiled function or subroutine definitions refered to in \texttt{func} and \texttt{jacfunc}. See package "deSolve".
hmin

an optional minimum value of the integration stepsize. In special situations this
parameter may speed up computations with the cost of precision. Don’t use
hmin if you don’t know why!

hmax

an optional maximum value of the integration stepsize. If not specified, hmax
is set to the largest difference in times, to avoid that the simulation possibly
ignores short-term events. If 0, no maximal size is specified.

Examples

=========================================================================
# use c code
# see cplSODA
=========================================================================

dlmlSODA

**LSODA Solver for NLME**

Description

Use Solver for Ordinary Differential Equations (ODE), Switching Automatically Between Stiff and
Non-stiff Methods and Generate functions to be used in NLME

Usage

dlmlSODA(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL,
hmin = 0, hmax = Inf)

Arguments

model
either an R-function that computes the values of the derivatives in the ODE
system (the *model definition*) at time t. The return value of model should be a
list. See package "nlmeODE" for more details.
data
nlme GroupedData format.
LogParms
transform parameters into log scale
JAC
A JAC set FALSE. This time we can implement this parts.
SEQ
A SEQ set FALSE.
rtol
relative error tolerance, either a scalar or an array as long as y. See details.
atol
absolute error tolerance, either a scalar or an array as long as y. See details.
tcrit
if not NULL, then lsoda cannot integrate past tcrit. The FORTRAN routine
lsoda overshoots its targets (times points in the vector times), and interpolates
values for the desired time points. If there is a time beyond which integration
should not proceed (perhaps because of a singularity), that should be provided
in tcrit.
hmin
an optional minimum value of the integration stepsize. In special situations this
parameter may speed up computations with the cost of precision. Don’t use
hmin if you don’t know why!
**nlmLSODAcp**

hmax

an optional maximum value of the integration stepsize. If not specified, hmax is set to the largest difference in times, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.

**Examples**

```r
# general model from nlmeODE package
# nlmLSODA USE ACCORDING FUNCTIONS

rm(list=ls())
require(insideRODE)

data(Theoph) # examples from nlmeODE
TheophODE <- Theoph
TheophODE$Dose[TheophODE$Time! = 0] <- 0
TheophODE$Cmt <- rep(1, dim(TheophODE)[1])

# model files
OneComp <- list(DiffEq=list(
  dy1dt = ~ -ka*y1,
  dy2dt = ~ ka*y1-ke*y2),
  ObsEq=list(
    c1 = ~ 0,
    c2 = ~ y2/CL*ke),
 Parms=c("ka","ke","CL"),
  States=c("y1","y2"),
  Init=list(0,0))

TheophModel <- nlmLSODA(OneComp,TheophODE) # ode solver
Theoph.nlme <- nlme(conc ~ TheophModel(ka,ke,CL,Time,Subject),
data = TheophODE, fixed=ka+ke+CL~1, random = pdDiag(ka+CL~1),
start=c(ka=0.5,ke=-2.5,CL=-3.2),
control=list(returnObject=TRUE,msVerbose=TRUE),
verbose=TRUE)

plot(augPred(Theoph.nlme,level=0:1))
```

---

**nlmLSODAcp  
LSODA Solver for NLME**

**Description**

Use Solver for Ordinary Differential Equations (ODE), Switching Automatically Between Stiff and Non-stiff Methods and Generate functions to be used in NLME

**Usage**

```
nlmLSODAcp(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL)```

Arguments

model either an R-function that computes the values of the derivatives in the ODE system (the model definition) at time t. The return value of model should be a list. See package "nlmeODE" for more details.

data nlme GroupedData format.
LogParms transform parameters into log scale
JAC A JAC set FALSE. This time we can implement this parts.
SEQ A SEQ set FALSE.
rtol relative error tolerance, either a scalar or an array as long as y. See details.
atol absolute error tolerance, either a scalar or an array as long as y. See details.
tcrit if not NULL, then lsoda cannot integrate past tcrit. The FORTRAN routine lsoda overshoots its targets (times points in the vector times), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in tcrit.
hmin an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use hmin if you don’t know why!
hmax an optional maximum value of the integration stepsize. If not specified, hmax is set to the largest difference in times, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.

Examples

########################################################################
#general model from nlmeODE package
#nlmLSODAcp USE ACCORDING FUNCTIONS
########################################################################

nlmLSODE LSODE Solver for NLME

Description

Use Solver for Ordinary Differential Equations (ODE), Switching Automatically Between Stiff and Non-stiff Methods and Generate functions to be used in NLME

Usage

nlmLSODE(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL, hmin = NULL, hmax = NULL)
Arguments

model: either an R-function that computes the values of the derivatives in the ODE system (the model definition) at time t. The return value of model should be a list. See package "nlmeODE" for more details.

data: nlme GroupedData format.

LogParms: transform parameters into log scale

JAC: A JAC set FALSE. This time we can implement this parts.

SEQ: A SEQ set FALSE.

rtol: relative error tolerance, either a scalar or an array as long as y. See details.

atol: absolute error tolerance, either a scalar or an array as long as y. See details.

tcrit: if not NULL, then lsoda cannot integrate past tcrit. The FORTRAN routine lsoda overshoots its targets (times points in the vector times), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in tcrit.

hmin: an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use hmin if you don’t know why!

hmax: an optional maximum value of the integration stepsize. If not specified, hmax is set to the largest difference in times, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.

Examples

#general model from nlmeODE package
#nlmlsode solver, use according functions

nlmlsodecpcp LSODE Solver for NLME

Description

Use Solver for Ordinary Differential Equations (ODE), Switching Automatically Between Stiff and Non-stiff Methods and Generate functions to be used in NLME

Usage

nlmlsodecp(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL)
Arguments

model  either an \texttt{R}-function that computes the values of the derivatives in the ODE system (the \textit{model definition}) at time \( t \). The return value of \texttt{model} should be a list. See package "\texttt{nlmeODE}" for more details.

data  \texttt{nlme GroupedData} format.

LogParms  transform parameters into log scale

JAC  A JAC set \texttt{FALSE}. This time we can implement this parts.

SEQ  A SEQ set \texttt{FALSE}.

rtol  relative error tolerance, either a scalar or an array as long as \( y \). See details.

atol  absolute error tolerance, either a scalar or an array as long as \( y \). See details.

tcrit  if not \texttt{NULL}, then \texttt{lsoda} cannot integrate past \texttt{tcrit}. The \texttt{FORTRAN} routine \texttt{lsoda} overshoots its targets (times points in the vector \texttt{times}), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in \texttt{tcrit}.

hmin  an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use \texttt{hmin} if you don’t know why!

hmax  an optional maximum value of the integration stepsize. If not specified, \texttt{hmax} is set to the largest difference in \texttt{times}, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.

Examples

########################################################################
#general model from nlmeODE package
#nlmLSODEcp SOLVER, USE ACCORDING FUNCTIONS
########################################################################

\texttt{nlmODE}  \textit{General Solver for Ordinary Differential Equations and Generate functions to be used in NLME, this version just use default function. The future version will provide buildin methods such as "lsoda", "lsode", "lsodes", "lsodar", "vode", "daspk", "euler", "rk4", "ode23", "ode45", "radua", "bdf", "bdf_d", "adams", "impAdams", "impAdams_d".}

Description

Generate functions for NLME Solves using a system of ordinary differential equations; a wrapper around the implemented ODE solvers.

Usage

\texttt{nlmODE(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL, hmin = NULL, hmax = NULL)}
Arguments

model       either an R-function that computes the values of the derivatives in the ODE system (the model definition) at time t. The return value of model should be a list. See package "nlmeODE" for more details.
data        nlme GroupedData format.
LogParms    transform parameters into log scale
JAC         A JAC set FALSE. This time we can implement this parts.
SEQ         A SEQ set FALSE.
rtol        relative error tolerance, either a scalar or an array as long as y. See details.
atol        absolute error tolerance, either a scalar or an array as long as y. See details.
tcrit       if not NULL, then lsoda cannot integrate past tcrit. The FORTRAN routine lsoda overshoots its targets (times points in the vector times), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in tcrit.
hmin        an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use hmin if you don’t know why!
hmax        an optional maximum value of the integration stepsize. If not specified, hmax is set to the largest difference in times, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.

Examples

General Solver for Ordinary Differential Equations and Generate functions to be used in NLME, this version just use default function. The future version will provide buildin methods such as "lsoda", "lsode", "lsodes", "lsodar", "vode", "daspk", "euler", "rk4", "ode23", "ode45", "radau", "bdf", "bdf_d", "adams", "impAdams", "impAdams_d".

Description

Generate functions for NLME Solves using a system of ordinary differential equations; a wrapper around the implemented ODE solvers

Usage

nlmODEcp(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL,
Arguments

- **model**: either an R-function that computes the values of the derivatives in the ODE system (the *model definition*) at time t. The return value of model should be a list. See package "nlmeODE" for more details.

- **data**: nlme GroupedData format.

- **LogParms**: transform parameters into log scale

- **JAC**: A JAC set FALSE. This time we can implement this parts.

- **SEQ**: A SEQ set FALSE.

- **rtol**: relative error tolerance, either a scalar or an array as long as y. See details.

- **atol**: absolute error tolerance, either a scalar or an array as long as y. See details.

- **tcrit**: if not NULL, then lsoda cannot integrate past tcrit. The FORTRAN routine lsoda overshoots its targets (times points in the vector times), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in tcrit.

- **hmin**: an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use hmin if you don’t know why!

- **hmax**: an optional maximum value of the integration stepsize. If not specified, hmax is set to the largest difference in times, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.

Examples

```
#general model from nlmeODE package
#nlmODEcp SOLVER, USE ACCORDING FUNCTIONS

nlmvode
```

### Description

Generate functions for NLME Solves using a system of ordinary differential equations; a wrapper around the implemented ODE solvers. The R function vode provides an interface to the FORTRAN ODE solver of the same name, written by Peter N. Brown, Alan C. Hindmarsh and George D. Byrne.

#### Usage

```
nlmvode(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL,
```
Arguments

model: either an R-function that computes the values of the derivatives in the ODE system (the model definition) at time t. The return value of model should be a list. See package "nlmeODE" for more details.

data: nlme GroupedData format.

LogParms: transform parameters into log scale.

JAC: A JAC set FALSE. This time we can implement this parts.

SEQ: A SEQ set FALSE.

rtol: relative error tolerance, either a scalar or an array as long as y. See details.

atol: absolute error tolerance, either a scalar or an array as long as y. See details.

tcrit: if not NULL, then lsoda cannot integrate past tcrit. The FORTRAN routine lsoda overshoots its targets (times points in the vector times), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in tcrit.

hmin: an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use hmin if you don’t know why!

hmax: an optional maximum value of the integration stepsize. If not specified, hmax is set to the largest difference in times, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.

Examples

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
#general model from nlmeODE package
#nlmVODE SOLVER, USE ACCORDING FUNCTIONS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nlmVODEcp

Solver for Ordinary Differential Equations (ODE) and Generate functions to be used in NLME

Description

Generate functions for NLME Solves using a system of ordinary differential equations; a wrapper around the implemented ODE solvers. The R function vode provides an interface to the FORTRAN ODE solver of the same name, written by Peter N. Brown, Alan C. Hindmarsh and George D. Byrne.

Usage

nlmVODEcp(model, data, LogParms = TRUE, JAC = FALSE, SEQ = FALSE, rtol = 1e-4, atol = 1e-4, tcrit = NULL)
Arguments

model
either an \texttt{R}-function that computes the values of the derivatives in the ODE system (the \textit{model definition}) at time \(t\). The return value of \texttt{model} should be a list. See package "\texttt{nlmeODE}" for more details.

data
\texttt{nlme} GroupedData format.

LogParms
transform parameters into log scale

JAC
A JAC set FALSE. This time we can implement this parts.

SEQ
A SEQ set FALSE.

rtol
relative error tolerance, either a scalar or an array as long as \(y\). See details.

atol
absolute error tolerance, either a scalar or an array as long as \(y\). See details.

tcrit
if not \texttt{NULL}, then \texttt{lsoda} cannot integrate past \texttt{tcrit}. The FORTRAN routine \texttt{lsoda} overshoots its targets (times points in the vector \texttt{times}), and interpolates values for the desired time points. If there is a time beyond which integration should not proceed (perhaps because of a singularity), that should be provided in \texttt{tcrit}.

hmin
an optional minimum value of the integration stepsize. In special situations this parameter may speed up computations with the cost of precision. Don’t use \texttt{hmin} if you don’t know why!

hmax
an optional maximum value of the integration stepsize. If not specified, \texttt{hmax} is set to the largest difference in \texttt{times}, to avoid that the simulation possibly ignores short-term events. If 0, no maximal size is specified.

Examples

#general model from nlmeODE package
#nlmVODe solver, USE ACCORDING FUNCTIONS

******************************************************************************
Index

*Topic math
   cflSODA, 3
   cflSODE, 5
   cfODE, 6
   cfVODE, 7
   cplSODA, 8
   cplSODE, 11
   cpODE, 12
   cpVODE, 13
   nlmLSODA, 14
   nlmLSODAcp, 15
   nlmLSODE, 16
   nlmLSODEcpcp, 17
   nlmODE, 18
   nlmODEcpcp, 19
   nlmVODE, 20
   nlmVODEcp, 21

*Topic package
   insideRODE-package, 2

   cflSODA, 3
   cflSODE, 5
   cfODE, 6
   cfVODE, 7
   cplSODA, 8
   cplSODE, 11
   cpODE, 12
   cpVODE, 13

   insideRODE (insideRODE-package), 2
   insideRODE-package, 2

   nlmLSODA, 14
   nlmLSODAcp, 15
   nlmLSODE, 16
   nlmLSODEcpcp (nlmLSODEcpcp), 17
   nlmLSODEcpcp, 17
   nlmODE, 18
   nlmODEcpcp (nlmODEcpcp), 19
   nlmODEcpcp, 19