Package ‘sundialr’

May 31, 2020

Type Package
Title An Interface to 'SUNDIALS' Ordinary Differential Equation (ODE) Solvers
Version 0.1.4
Maintainer Satyaprakash Nayak <sn248@cornell.edu>
URL https://github.com/sn248/sundialr
BugReports https://github.com/sn248/sundialr/issues
Description Provides a way to call the functions in 'SUNDIALS' C ODE solving library (<https://computation.llnl.gov/projects/sundials>). Currently the serial version of ODE solver, 'CVODE', sensitivity calculator 'CVODES' and differential algebraic solver 'IDA' from the 'SUNDIALS' library are implemented. The package requires ODE to be written as an 'R' or 'Rcpp' function and does not require the 'SUNDIALS' library to be installed on the local machine.
License BSD_3_clause + file LICENSE
Copyright file COPYRIGHTS
LazyData TRUE
Imports Rcpp (>= 0.12.5)
LinkingTo Rcpp, RcppArmadillo
RoxygenNote 7.1.0
Suggests knitr, rmarkdown, testthat
VignetteBuilder knitr
NeedsCompilation yes
Author Satyaprakash Nayak [aut, cre, cph]
  (<https://orcid.org/0000-0001-7225-1317>), Lawrence Livermore National Security [cph], Southern Methodist University [cph]
Repository CRAN
Date/Publication 2020-05-31 21:30:07 UTC
cvode

Description

CVODE solver to solve stiff ODEs

Usage

cvode(
  time_vector,
  IC,
  input_function,
  Parameters,
  reltolerance = 1e-04,
  abstolerance = 1e-04
)

Arguments

time_vector  time vector
IC           Initial Conditions
input_function Right Hand Side function of ODEs
Parameters   Parameters input to ODEs
reltolerance Relative Tolerance (a scalar, default value = 1e-04)
abstolerance Absolute Tolerance (a scalar or vector with length equal to ydot, default = 1e-04)

Examples

# Example of solving a set of ODEs with cvode function
# ODEs described by an R function
ODE_R <- function(t, y, p){
  # vector containing the right hand side gradients
  ydot = vector(mode = "numeric", length = length(y))

  # R indices start from 1
}

# Example of solving a set of ODEs with cvode function
# ODEs described by an R function
ODE_R <- function(t, y, p){
  # vector containing the right hand side gradients
  ydot = vector(mode = "numeric", length = length(y))

  # R indices start from 1
}


ydot

}  

# ODEs can also be described using Rcpp
Rcpp::sourceCpp(code = '  
#include "Rcpp.h"
using namespace Rcpp;

// ODE functions defined using Rcpp
// [[Rcpp::export]]
NumericVector ODE_Rcpp (double t, NumericVector y, NumericVector p){
  
  // Initialize ydot filled with zeros
  NumericVector ydot(y.length());
  ydot[0] = -p[0]*y[0] + p[1]*y[1]*y[2];
  ydot[2] = p[2]*y[1]*y[1];

  return ydot;
}

\n
# R code to generate time vector, IC and solve the equations

time_vec <- c(0.0, 0.4, 4.0, 40.0, 4E2, 4E3, 4E4, 4E5, 4E6, 4E7, 4E8, 4E9, 4E10)
IC <- c(1,0,0)
params <- c(0.04, 10000, 30000000)
reltol <- 1e-04
abstol <- c(1e-8,1e-14,1e-6)

## Solving the ODEs using cvode function
df1 <- cvode(time_vec, IC, ODE_R , params, reltol, abstol) ## using R
df2 <- cvode(time_vec, IC, ODE_Rcpp , params, reltol, abstol) ## using Rcpp

## Check that both solutions are identical
# identical(df1, df2)
Description

CVODES solver to solve ODEs and calculate sensitivities

Usage

cvodes(
  time_vector,
  IC,
  input_function,
  Parameters,
  reltolerance = 1e-04,
  abstolerance = 1e-04,
  SensType = "STG",
  ErrCon = "F"
)

Arguments

time_vector   time vector
IC             Initial Conditions
input_function Right Hand Side function of ODEs
Parameters     Parameters input to ODEs
reltolerance   Relative Tolerance (a scalar, default value = 1e-04)
abstolerance   Absolute Tolerance (a scalar or vector with length equal to ydot, default = 1e-04)
SensType       Sensitivity Type - allowed values are Staggered (default), "STG" (for Staggered) or "SIM" (for Simultaneous)
ErrCon         Error Control - allowed values are TRUE or FALSE (default)

Examples

# Example of solving a set sensitivity equations for ODEs with cvodes function
# ODEs described by an R function
ODE_R <- function(t, y, p){

  # vector containing the right hand side gradients
  ydot = vector(mode = "numeric", length = length(y))

  # R indices start from 1


  ydot
}
# ODEs can also be described using Rcpp
Rcpp::sourceCpp(code = '  
#include <Rcpp.h>
using namespace Rcpp;

// ODE functions defined using Rcpp
// [[Rcpp::export]]
NumericVector ODE_Rcpp (double t, NumericVector y, NumericVector p){

  // Initialize ydot filled with zeros
  NumericVector ydot(y.length());
  ydot[0] = -p[0]*y[0] + p[1]*y[1]*y[2];
  ydot[2] = p[2]*y[1]*y[1];
  return ydot;
}

# R code to generate time vector, IC and solve the equations
time_vec <- c(0.0, 0.4, 4.0, 40.0, 4E2, 4E3, 4E4, 4E5, 4E6, 4E7, 4E8, 4E9, 4E10)
IC <- c(1,0,0)
params <- c(0.04, 10000, 30000000)
reltol <- 1e-04
abstol <- c(1e-8,1e-14,1e-6)

## Solving the ODEs and Sensitivities using cvodes function
df1 <- cvodes(time_vec, IC, ODE_R, params, reltol, abstol,"STG",FALSE) ## using R
df2 <- cvodes(time_vec, IC, ODE_Rcpp, params, reltol, abstol,"STG",FALSE) ## using Rcpp

## Check that both solutions are identical
identical(df1, df2)

---

cvsolve
cvsolve

**Description**

CVSOLVE solver to solve stiff ODEs with discontinuities

**Usage**

cvsolve(
  time_vector,
IC, input_function, Parameters, Events = NULL, reltolerance = 1e-04, abstolerance = 1e-04)

Arguments

time_vector  time vector
IC  Initial Conditions
input_function  Right Hand Side function of ODEs
Parameters  Parameters input to ODEs
Events  Discontinuities in the solution (a DataFrame, default value is NULL)
reltolerance  Relative Tolerance (a scalar, default value = 1e-04)
abstolerance  Absolute Tolerance (a scalar or vector with length equal to ydot, default = 1e-04)

Examples

# Example of solving a set of ODEs with multiple discontinuities using cvsolve
# A simple One dimensional equation, y = -0.1 * y
# ODEs described by an R function
ODE_R <- function(t, y, p){

  # vector containing the right hand side gradients
  ydot = vector(mode = "numeric", length = length(y))

  # R indices start from 1

  ydot
}

# R code to generate time vector, IC and solve the equations
TSAMP <- seq(from = 0, to = 100, by = 0.1)  # sampling time points
IC <- c(1)
params <- c(0.1)

# A dataset describing the dosing at times at which additions to y[1] are to be done
# Names of the columns don't matter, but they MUST be in the order of state index,
# times and Values at discontinuity.
TDOSE <- data.frame(ID = 1, TIMES = c(0, 10, 20, 30, 40, 50), VAL = 100)
df1 <- cvsolve(TSAMP, c(1), ODE_R, params)  # solving without any discontinuity
df2 <- cvsolve(TSAMP, c(1), ODE_R, params, TDOSE)  # solving with discontinuity
ida

Description
IDA solver to solve stiff DAEs

Usage
ida(
    time_vector,
    IC,
    IRes,
    input_function,
    Parameters,
    reltolerance = 1e-04,
    abstolerance = 1e-04
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_vector</td>
<td>time vector</td>
</tr>
<tr>
<td>IC</td>
<td>Initial Value of y</td>
</tr>
<tr>
<td>IRes</td>
<td>Initial Value of ydot</td>
</tr>
<tr>
<td>input_function</td>
<td>Right Hand Side function of DAEs</td>
</tr>
<tr>
<td>Parameters</td>
<td>Parameters input to ODEs</td>
</tr>
<tr>
<td>reltolerance</td>
<td>Relative Tolerance (a scalar, default value = 1e-04)</td>
</tr>
<tr>
<td>abstolerance</td>
<td>Absolute Tolerance (a scalar or vector with length equal to ydot, default = 1e-04)</td>
</tr>
</tbody>
</table>
Index

cvode, 2
cvodes, 3
cvsolve, 5
ida, 7