Package ‘diffEq’

February 19, 2015

Version 1.0-1

Title Functions from the book Solving Differential Equations in R

Author Karline Soetaert <karline.soetaert@nioz.nl>

Maintainer Karline Soetaert <karline.soetaert@nioz.nl>

Depends R (>= 2.01), deSolve, rootSolve, bvpSolve, ReacTran, deTestSet

Imports shape

Suggests scatterplot3d


License GPL

LazyData yes

Repository CRAN

Repository/R-Forge/Project bvp.solve

Repository/R-Forge/Revision 223

Repository/R-Forge/DateTimeStamp 2014-12-26 10:33:59

Date/Publication 2014-12-29 13:40:11

NeedsCompilation no

R topics documented:

diffEq-package .................................................. 2
Coefficients ...................................................... 3
rkMethodPlot ..................................................... 4
stability.multipstep .............................................. 6

Index 10
diffEq-package

Functions to create the figures of the book "solving differential equations in R" by Karline Soetaert, Jeff R. Cash and Francesca Mazzia. Published by Springer

Description

R package diffEq contains the functions for generating figures relating to differential equations

Details

Package: diffEq
Type: Package
Version: 1.0
License: GNU Public License 2 or above

Author(s)

Karline Soetaert (Maintainer),
Jeff Cash,
Francesca Mazzia

See Also

rkMethodPlot for plotting the steps of runge-kutta methods
stability.multistep, stability.bruteforce for plotting stability regions
Coefficients for the BDF, AdamsMoulton, AdamsBashford coefficients.
rkMethod from package deSolve for the Runge-Kutta coefficients

Examples

## Not run:
## show examples (see respective help pages for details)
example(rkMethodPlot)
example(stability.multistep)
example(BDF)

## open the directory with R sourcecode examples
browseURL(paste(system.file(package = "diffEq"), "/doc/examples", sep = ""))

## show package vignette with how to use the package
## + source code of the vignette
vignette("diffEq")
Coefficients

The coefficients of multistep methods

Description

Coefficients alpha and beta of the Adams-Bashforth, Adams-Moulton and Backward differentiation formulae.

\[ \sum_{j=0}^{k} \alpha_j y_{n-j} = h \sum_{j=0}^{k} \beta_j f(x_{n-j}, y_{n-j}) \]

For the BDF methods, the angle of the stability-region (the alpha of the A(alpha) stability, in radians is also given.

Usage

BDF
AdamsMoulton
AdamsBashforth

Author(s)

Karline Soetaert

See Also

stability.multistep for plotting stability regions

Examples

```R
## Stability properties

BDF

stability.multistep(alpha = BDF$alpha[3,], beta = BDF$beta[3,],
xlim = c(-7,7), ylim = c(-7,7))

stability.multistep(alpha = AdamsMoulton$alpha[3,], beta = AdamsMoulton$beta[3,],
xlim = c(-7,7), ylim = c(-7,7))

stability.multistep(alpha = AdamsBashforth$alpha[3,], beta = AdamsBashforth$beta[3,])
```

## Running a BDF

```R
```
# test model
ode1 <- function(t, y) return(cos(t)*y)

h <- 0.01
times <- seq(from = 0, to = 20, by = h)
yout <- vector(length = length(times))
yout[1] <- 1

# 3rd order BDF
Alpha <- BDF$alpha[3:2:4]
Beta <- BDF$beta[3]

bdf <- function(y, t, h, f, ys) {
  rootfun <- function(ynext) ynext - sum(Alpha * ys) + Beta * h * f(t + h, ynext)
  y <- multiroot(f=rootfun, start=y)$root
  ys[2:3] <- y[1:2]
  ys[1] <- y
  list(y = y, ys=ys)
}

# Spinup uses Euler...
Euler <- function(y, t, h, f) {
  fn <- f(t, y)
  ynext <- y + h * fn
  list(y = ynext, fn = fn)
}

for (i in 2:3) yout[i] <- Euler(yout[i-1], times[i-1], h, ode1)$y

ys <- rev(yout[1:3])

# BDF steps
for (i in 4:length(times)){
  step <- bdf (y=yout[i-1], t=times[i-1], h=h, f=ode1, ys=ys)
  yout[i] <- step$y
  ys <- step$ys
}

rkMethodPlot
Plots the steps in runge-kutta methods

Description

...
**Usage**

rkMethodPlot (rk, ...)

**Arguments**

rk  
A list containing the runge-kutta coefficients, implicit or explicit, e.g. matrix A, vectors b1, b2, codec. The list can be of type rkMethod, as defined in package deSolve.

...  
arguments passed to the plotting function.

**Value**

Returns nothing

**Author(s)**

Karline Soetaert

**See Also**

stability.bruteforce for plotting stability regions of Runge-Kutta methods.

**Examples**

```r
# This to plot all runge kutta methods
RKS <- rkMethod()
for (i in 4:21)  rkMethodPlot( rkMethod(RKS[i])

## Figures A and B: Cash-Karp and Radau 5 steps
####----------------------------------------------------------------------------
par(mfrow=c(2,2))

rkMethodPlot( rkMethod("rk45ck"), main="Cash-Karp")
writelabel("A")

rkMethodPlot( rkMethod("irk5"), main="Radau5")
writelabel("B")

rkMethodPlot( rkMethod("rk45dp6"), main="Dopri")
writelabel("C")

rkMethodPlot( rkMethod("irk61"), main="Lobatto")
writelabel("D")

legend("bottomright", pch = c(16, 16, 1, NA), pt.cex = c(1.5, 1.5, 1),
       legend = c(expression(y[0]), expression(y[1]), "intermediary", "k"),
       col = c("grey", "black", "black", "black"), lty = c(NA, NA, NA, 1),
       lwd = c(1, 1, 1, 2))
```
stability.multistep

Plots the stability function of multistep methods

Description

...

Usage

stability.multistep (alpha, beta, add = FALSE, fill = NULL, ...)

stability.bruteforce (Rez = seq(-2, 2, by = 0.02),
                    Imz = seq(-2, 2, by = 0.02),
                    func = function (z) return(abs(1 + h*z) <=1),
                    fill = "grey", cex = 1.5, add = FALSE, ...)

Arguments

alpha        alpha coefficients of the multistep method.
beta         beta coefficients of the multistep method.
add           if TRUE, the new region will be added to the existing plot
fill          color of region to be filled
Rez           The range in the real plane for testing stability
Imz           The range in the imaginary plane for testing stability
func          The function to be tested; default is test for the euler method.
cex           The relative size of the plotting symbol. If too small, the region will not be completely covered. If too large, it will extend beyond its boundaries.

Value

A matrix with the boundary value.

Author(s)

Karline Soetaert

See Also

rkMethodPlot for plotting runge-kutta method steps.
Examples

```r
par(mfrow=c(2,2))

## Stability regions for multistep methods

# Adams-Bashforth
stability.multistep(alpha = AdamsBashforth$alpha[2,], beta = AdamsBashforth$beta[2,],
xlim = c(-3,1), ylim = c(-1.5, 1.5),
fill = "black", main = "Adams-Bashforth")

stability.multistep(alpha = AdamsBashforth$alpha[3,], beta = AdamsBashforth$beta[3,],
add = TRUE, lty = 1, fill = "darkgrey")

stability.multistep(alpha = AdamsBashforth$alpha[4,], beta = AdamsBashforth$beta[4,],
add = TRUE, fill = "lightgrey", lty = 1)

stability.multistep(alpha = AdamsBashforth$alpha[5,], beta = AdamsBashforth$beta[5,],
add = TRUE, fill = "white", lty = 1)

legend("topleft", fill = c("black", "darkgrey", "lightgrey", "white"),
title = "order", legend = 2:5)
writelabel("A")

# Adams-Moulton
stability.multistep(alpha = AdamsMoulton$alpha[3,], beta = AdamsMoulton$beta[3,],
xlim = c(-8,1), ylim = c(-4, 4),
fill = "black", main = "Adams-Moulton")

stability.multistep(alpha = AdamsMoulton$alpha[4,], beta = AdamsMoulton$beta[4,],
add = TRUE, fill = "darkgrey")

stability.multistep(alpha = AdamsMoulton$alpha[5,], beta = AdamsMoulton$beta[5,],
add = TRUE, fill = "lightgrey")

legend("topleft", fill = c("black", "darkgrey", "lightgrey"),
title = "order", legend = 3:5)
writelabel("B")

# 2nd-order BDF
plot(0, type="n", xlim = c(-3, 12), ylim = c(-8, 8),
main = "BDF order 2",
  xlab = "Re(z)", ylab = "Im(z)")
rect(-100, -100, 100, 100, col = "lightgrey")
box()

stability.multistep (alpha = BDF$alpha[2,], beta = BDF$beta[2,],
fill = "white", add = TRUE)

writelabel("C")

# 4th-order BDF
```

plot(0, type="n", xlim=c(-3, 12), ylim = c(-8, 8),
  main = "BDF order 4",
  xlab = "Re(z)", ylab = "Im(z)"
rect(-i00, -i00, i00, i00, col = "lightgrey")
box()
stability.multistep(alpha = BDF$alpha[4,], beta = BDF$beta[4,],
  fill = "white", add = TRUE)

writelabel("D")

## Stability regions for runge-kutta methods
## # Drawing the stability regions - brute force
# stability function for explicit runge-kutta's
rkstabfunc <- function (z, order = 1) {
  h <- 1
  ss <- 1
  for (p in 1: order) ss <- ss + (h*z)^p / factorial(p)
  return (abs(ss) <= 1)
}

# regions for stability orders 5 to 1 - rather crude
Rez <- seq(-5, 1, by = 0.1)
Imz <- seq(-3, 3, by = 0.1)
stability.bruteforce(main = "Explicit RK",
  func = function(z) rkstabfunc(z, order = 5),
  Rez = Rez, Imz = Imz, fill = grey(0.95) )
stability.bruteforce(add = TRUE,
  func = function(z) rkstabfunc(z, order = 4),
  Rez = Rez, Imz = Imz, fill = grey(0.75) )
stability.bruteforce(add = TRUE,
  func = function(z) rkstabfunc(z, order = 3),
  Rez = Rez, Imz = Imz, fill = grey(0.55) )
stability.bruteforce(add = TRUE,
  func = function(z) rkstabfunc(z, order = 2),
  Rez = Rez, Imz = Imz, fill = grey(0.35) )
stability.bruteforce(add = TRUE,
  func = function(z) rkstabfunc(z, order = 1),
  Rez = Rez, Imz = Imz, fill = grey(0.15) )

legend("topleft", legend = 5:1, title = "order",
  fill = grey(c(0.95, 0.75, 0.55, 0.35, 0.15)))

# stability function for radau method
stability.bruteforce(main = "Radau 5",
  Rez = seq(-5,15,by=0.1), Imz = seq(-10,10,by=0.12),
func = function(z) return(abs((1 + 2*z/5 + z^2/20) / 
(1 - 3*z/5 + 3*z^2/20 - z^3/60)) <= 1),
col = grey(0.8) )
index

*Topic ** datasets
  Coefficients, 3
*Topic ** math
  stability.multistep, 6
*Topic ** package
  diffEq-package, 2
*Topic ** plot
  rkMethodPlot, 4

AdamsBashforth (Coefficients), 3
AdamsMoulton (Coefficients), 3

BDF (Coefficients), 3

Coefficients, 2, 3
diffEq (diffEq-package), 2
diffEq-package, 2

rkMethodPlot, 2, 4, 6

stability.bruteforce, 2, 5
stability.bruteforce
  (stability.multistep), 6
stability.multistep, 2, 3, 6