Package: RTMBode (via r-universe)

September 16, 2024

Type Package

Title Solving ODEs with 'deSolve' and 'RTMB'.

Version 1.0

Date 2024-03-21

Author Kasper Kristensen

Maintainer kaskr <kaskr@dtu.dk>

Description Solve ODEs as part of 'RTMB' objective functions with algorithmic derivatives to any order. ODE adjoint code is obtained via 'RTMB' using automatic variable augmentation of the system equation.

License GPL (>= 2)

Imports RTMB, deSolve

Roxygen list(markdown = TRUE)

Repository https://kaskr.r-universe.dev

RemoteUrl https://github.com/kaskr/RTMB

RemoteRef HEAD

RemoteSha 4341d96793684669bd389db500d5f9ff63bf1950

Contents

ode 2

Index **4**

ode *ODE solver via 'deSolve' with 'RTMB' autodiff capabilities.*

Description

This ode solver is essentially a wrapper around the corresponding function from the deSolve package. It adds the following extra features:

- Autodiff adjoint code so that ODE solving can be used as part of general gradient based optimization.
- Faster ODE solving using RTMB 'tapes' to eliminate R interpreter overhead.

Usage

```
ode(y, times, func, parms, method = NULL, ...)
```

Arguments

y	the initial (state) values for the ODE system, a vector (see deSolve package).
times	time sequence for which output is wanted (see deSolve package).
func	an R-function that computes the values of the derivatives in the ODE system (see deSolve package).
parms	parameters passed to func (see deSolve package).
method	the integrator to use (see deSolve package).
...	additional arguments passed to the integrator (see deSolve package).

Value

Solution matrix with time as the first column (see deSolve package).

Examples

```
require(RTMB)
## Lotka-Volterra example from 'deSolve' manual
LVmod <- function(Time, State, Pars) {
    with(as.list(c(State, Pars)), {
        Ingestion <- rIng * Prey * Predator
        GrowthPrey <- rGrow * Prey * (1 - Prey/K)
        MortPredator <- rMort * Predator
        dPrey <- GrowthPrey - Ingestion
        dPredator <- Ingestion * assEff - MortPredator
        return(list(c(dPrey, dPredator)))
    })
}
pars <- c(rIng = 0.2, # /day, rate of ingestion
          rGrow = 1.0, # /day, growth rate of prey
          rMort = 0.2 , # /day, mortality rate of predator
```

```
          assEff = 0.5, # -, assimilation efficiency
          K = 10) # mmol/m3, carrying capacity
yini <- c(Prey = 1, Predator = 2)
times <- seq(0, 200, by = 1)
## Simulate ODE with measurement noise
set.seed(1)
obs <- deSolve::ode(func = LVmod, y = yini, parms = pars, times = times)[,-1]
obs <- obs + rnorm(length(obs), sd=1)
## Likelihood function
likfun <- function(p) {
    getAll(p)
    obs <- OBS(obs)
    sol <- ode(func = LVmod, y = yini, parms = pars, times = times, atol=1e-8, rtol=1e-8)
    obs %~% dnorm(mean=sol[,-1], sd=sdobs)
}
## Initial guess
p <- list(pars=pars*1.5, yini=yini*1.5, sdobs=1.5)
## Parameter estimation
obj <- MakeADFun(likfun, p, silent=TRUE)
system.time(opt <- nlminb(obj$par,obj$fn,obj$gr))
(sdr <- sdreport(obj))
## as.list(sdr, "Est")
## as.list(sdr, "Std")
```

Index

ode, 2