typetopcompile.r

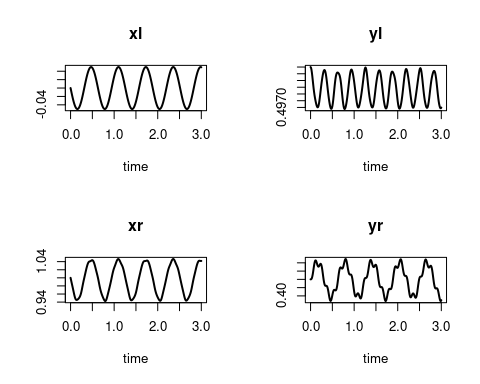
denis

2021-07-12

#!/usr/bin/r  
  
### R code from vignette source 'compiledCode.Rnw'  
  
###################################################  
### code chunk number 1: preliminaries  
###################################################  
library("deSolve")  
options(prompt = "R> ")  
options(width=70)  
  
  
###################################################  
### code chunk number 2: the\_Rmodel  
###################################################  
model <- function(t, Y, parameters) {  
 with (as.list(parameters),{  
   
 dy1 = -k1\*Y[1] + k2\*Y[2]\*Y[3]  
 dy3 = k3\*Y[2]\*Y[2]  
 dy2 = -dy1 - dy3  
   
 list(c(dy1, dy2, dy3))  
 })  
}  
  
  
###################################################  
### code chunk number 3: Jacobian\_in\_R  
###################################################  
jac <- function (t, Y, parameters) {  
 with (as.list(parameters),{  
   
 PD[1,1] <- -k1  
 PD[1,2] <- k2\*Y[3]  
 PD[1,3] <- k2\*Y[2]  
 PD[2,1] <- k1  
 PD[2,3] <- -PD[1,3]  
 PD[3,2] <- k3\*Y[2]  
 PD[2,2] <- -PD[1,2] - PD[3,2]  
   
 return(PD)  
 })  
}  
  
  
###################################################  
### code chunk number 4: Run\_Rmodel  
###################################################  
parms <- c(k1 = 0.04, k2 = 1e4, k3=3e7)  
Y <- c(1.0, 0.0, 0.0)  
times <- c(0, 0.4\*10^(0:11))  
PD <- matrix(nrow = 3, ncol = 3, data = 0)  
out <- ode(Y, times, model, parms = parms, jacfunc = jac)  
  
  
###################################################  
### code chunk number 5: compile\_DLLmodel\_F (eval = FALSE)  
###################################################  
## system("R CMD SHLIB mymod.f")  
  
  
###################################################  
### code chunk number 6: compile\_DLLmodel\_C (eval = FALSE)  
###################################################  
## system("R CMD SHLIB mymod.c")  
  
  
###################################################  
### code chunk number 7: compiledCode.Rnw:725-767  
###################################################  
caraxisfun <- function(t, y, parms) {  
 with(as.list(c(y, parms)), {  
 yb <- r \* sin(w \* t)  
 xb <- sqrt(L \* L - yb \* yb)  
 Ll <- sqrt(xl^2 + yl^2)  
 Lr <- sqrt((xr - xb)^2 + (yr - yb)^2)  
   
 dxl <- ul; dyl <- vl; dxr <- ur; dyr <- vr  
   
 dul <- (L0-Ll) \* xl/Ll + 2 \* lam2 \* (xl-xr) + lam1\*xb  
 dvl <- (L0-Ll) \* yl/Ll + 2 \* lam2 \* (yl-yr) + lam1\*yb - k \* g  
   
 dur <- (L0-Lr) \* (xr-xb)/Lr - 2 \* lam2 \* (xl-xr)  
 dvr <- (L0-Lr) \* (yr-yb)/Lr - 2 \* lam2 \* (yl-yr) - k \* g  
   
 c1 <- xb \* xl + yb \* yl  
 c2 <- (xl - xr)^2 + (yl - yr)^2 - L \* L  
   
 list(c(dxl, dyl, dxr, dyr, dul, dvl, dur, dvr, c1, c2))  
 })  
}  
eps <- 0.01; M <- 10; k <- M \* eps^2/2;  
L <- 1; L0 <- 0.5; r <- 0.1; w <- 10; g <- 1  
  
parameter <- c(eps = eps, M = M, k = k, L = L, L0 = L0,  
 r = r, w = w, g = g)  
  
yini <- c(xl = 0, yl = L0, xr = L, yr = L0, ul = -L0/L, vl = 0,  
 ur = -L0/L, vr = 0, lam1 = 0, lam2 = 0)  
  
# the mass matrix  
Mass <- diag(nrow = 10, 1)  
Mass[5,5] <- Mass[6,6] <- Mass[7,7] <- Mass[8,8] <- M \* eps \* eps/2  
Mass[9,9] <- Mass[10,10] <- 0  
Mass

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]  
## [1,] 1 0 0 0 0e+00 0e+00 0e+00 0e+00 0 0  
## [2,] 0 1 0 0 0e+00 0e+00 0e+00 0e+00 0 0  
## [3,] 0 0 1 0 0e+00 0e+00 0e+00 0e+00 0 0  
## [4,] 0 0 0 1 0e+00 0e+00 0e+00 0e+00 0 0  
## [5,] 0 0 0 0 5e-04 0e+00 0e+00 0e+00 0 0  
## [6,] 0 0 0 0 0e+00 5e-04 0e+00 0e+00 0 0  
## [7,] 0 0 0 0 0e+00 0e+00 5e-04 0e+00 0 0  
## [8,] 0 0 0 0 0e+00 0e+00 0e+00 5e-04 0 0  
## [9,] 0 0 0 0 0e+00 0e+00 0e+00 0e+00 0 0  
## [10,] 0 0 0 0 0e+00 0e+00 0e+00 0e+00 0 0

# index of the variables: 4 of index 1, 4 of index 2, 2 of index 3  
index <- c(4, 4, 2)  
  
times <- seq(0, 3, by = 0.01)  
out <- radau(y = yini, mass = Mass, times = times, func = caraxisfun,  
 parms = parameter, nind = index)  
  
  
###################################################  
### code chunk number 8: caraxis  
###################################################  
plot(out, which = 1:4, type = "l", lwd = 2)  
  
  
###################################################  
### code chunk number 9: figcaraxis  
###################################################  
plot(out, which = 1:4, type = "l", lwd = 2)



###################################################  
### code chunk number 10: compiledCode.Rnw:950-979  
###################################################  
## the model, 5 state variables  
f1 <- function (t, y, parms) {  
 ydot <- vector(len = 5)  
   
 ydot[1] <- 0.1\*y[1] -0.2\*y[2]  
 ydot[2] <- -0.3\*y[1] +0.1\*y[2] -0.2\*y[3]  
 ydot[3] <- -0.3\*y[2] +0.1\*y[3] -0.2\*y[4]  
 ydot[4] <- -0.3\*y[3] +0.1\*y[4] -0.2\*y[5]  
 ydot[5] <- -0.3\*y[4] +0.1\*y[5]  
   
 return(list(ydot))  
}  
  
## the Jacobian, written in banded form  
bandjac <- function (t, y, parms) {  
 jac <- matrix(nrow = 3, ncol = 5, byrow = TRUE,  
 data = c( 0 , -0.2, -0.2, -0.2, -0.2,  
 0.1, 0.1, 0.1, 0.1, 0.1,  
 -0.3, -0.3, -0.3, -0.3, 0))  
 return(jac)  
}  
  
## initial conditions and output times  
yini <- 1:5  
times <- 1:20  
  
## stiff method, user-generated banded Jacobian  
out <- lsode(yini, times, f1, parms = 0, jactype = "bandusr",  
 jacfunc = bandjac, bandup = 1, banddown = 1)  
  
  
###################################################  
### code chunk number 11: compiledCode.Rnw:1062-1073  
###################################################  
## Parameter values and initial conditions  
Parms <- c(0.182, 4.0, 4.0, 0.08, 0.04, 0.74, 0.05, 0.15, 0.32,  
 16.17, 281.48, 13.3, 16.17, 5.487, 153.8, 0.04321671,  
 0.4027255, 1000, 0.02, 1.0, 3.8)  
  
yini <- c( AI=21, AAM=0, AT=0, AF=0, AL=0, CLT=0, AM=0 )  
  
## the rate of change  
DLLfunc(y = yini, dllname = "deSolve", func = "derivsccl4",  
 initfunc = "initccl4", parms = Parms, times = 1,  
 nout = 3, outnames = c("DOSE", "MASS", "CP") )

## $dy  
## AI AAM AT AF AL   
## -20.0582048 6.2842256 9.4263383 0.9819102 2.9457307   
## CLT AM   
## 0.0000000 0.0000000   
##   
## $var  
## DOSE MASS CP   
## 1.758626 0.000000 922.727067

###################################################  
### code chunk number 12: compiledCode.Rnw:1084-1100  
###################################################  
pars <- c(K = 1, ka = 1e6, r = 1)  
  
## Initial conc; D is in equilibrium with A,B  
y <- c(A = 2, B = 3, D = 2\*3/pars["K"])  
  
## Initial rate of change  
dy <- c(dA = 0, dB = 0, dD = 0)  
  
## production increases with time  
prod <- matrix(nc=2,data=c(seq(0,100,by=10),seq(0.1,0.5,len=11)))  
  
DLLres(y=y,dy=dy,times=5,res="chemres",  
 dllname="deSolve", initfunc="initparms",  
 initforc="initforcs", parms=pars, forcings=prod,  
 nout=2, outnames=c("CONC","Prod"))

## $delta  
## A B D.K   
## 0.00 -3.00 0.12   
##   
## $var  
## CONC Prod   
## 11.00 0.12

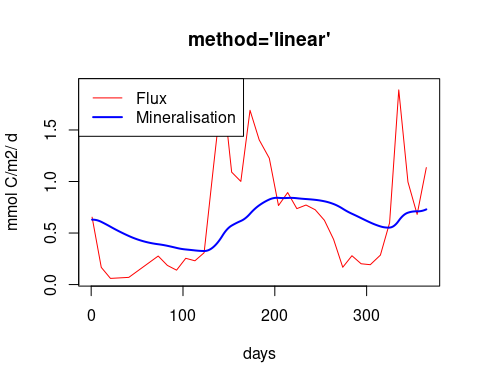
###################################################  
### code chunk number 13: compiledCode.Rnw:1268-1276  
###################################################  
Flux <- matrix(ncol=2,byrow=TRUE,data=c(  
 1, 0.654, 11, 0.167, 21, 0.060, 41, 0.070, 73,0.277, 83,0.186,  
 93,0.140,103, 0.255, 113, 0.231,123, 0.309,133,1.127,143,1.923,  
 153,1.091,163,1.001, 173, 1.691,183, 1.404,194,1.226,204,0.767,  
 214, 0.893,224,0.737, 234,0.772,244, 0.726,254,0.624,264,0.439,  
 274,0.168,284 ,0.280, 294,0.202,304, 0.193,315,0.286,325,0.599,  
 335, 1.889,345, 0.996,355,0.681,365,1.135))  
head(Flux)

## [,1] [,2]  
## [1,] 1 0.654  
## [2,] 11 0.167  
## [3,] 21 0.060  
## [4,] 41 0.070  
## [5,] 73 0.277  
## [6,] 83 0.186

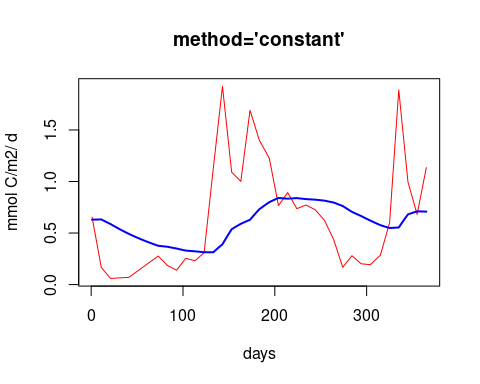
###################################################  
### code chunk number 14: compiledCode.Rnw:1281-1282  
###################################################  
parms <- 0.01  
  
  
###################################################  
### code chunk number 15: compiledCode.Rnw:1288-1290  
###################################################  
meanDepo <- mean(approx(Flux[,1],Flux[,2], xout=seq(1,365,by=1))$y)  
Yini <- c(y=meanDepo/parms)  
  
  
###################################################  
### code chunk number 16: compiledCode.Rnw:1306-1313  
###################################################  
times <- 1:365  
out <- ode(y=Yini, times, func = "scocder",  
 parms = parms, dllname = "deSolve",  
 initforc="scocforc", forcings=Flux,  
 initfunc = "scocpar", nout = 2,  
 outnames = c("Mineralisation","Depo"))  
head(out)

## time y Mineralisation Depo  
## [1,] 1 63.00301 0.6300301 0.6540  
## [2,] 2 63.00262 0.6300262 0.6053  
## [3,] 3 62.95377 0.6295377 0.5566  
## [4,] 4 62.85694 0.6285694 0.5079  
## [5,] 5 62.71259 0.6271259 0.4592  
## [6,] 6 62.52124 0.6252124 0.4105

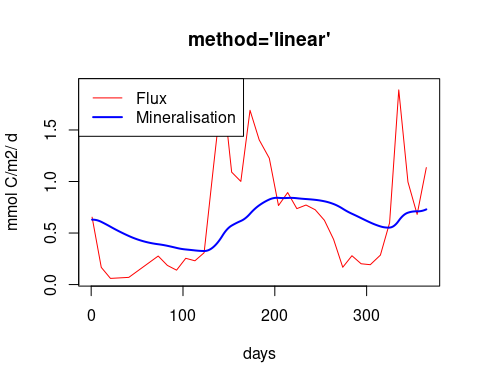
###################################################  
### code chunk number 17: compiledCode.Rnw:1319-1325  
###################################################  
fcontrol <- list(method="constant")  
out2 <- ode(y=Yini, times, func = "scocder",  
 parms = parms, dllname = "deSolve",  
 initforc="scocforc", forcings=Flux, fcontrol=fcontrol,  
 initfunc = "scocpar", nout = 2,  
 outnames = c("Mineralisation","Depo"))  
  
  
###################################################  
### code chunk number 18: scoc  
###################################################  
par (mfrow=c(1,2))  
plot(out, which = "Depo", col="red",  
 xlab="days", ylab="mmol C/m2/ d", main="method='linear'")  
lines(out[,"time"], out[,"Mineralisation"], lwd=2, col="blue")  
legend("topleft",lwd=1:2,col=c("red","blue"), c("Flux","Mineralisation"))



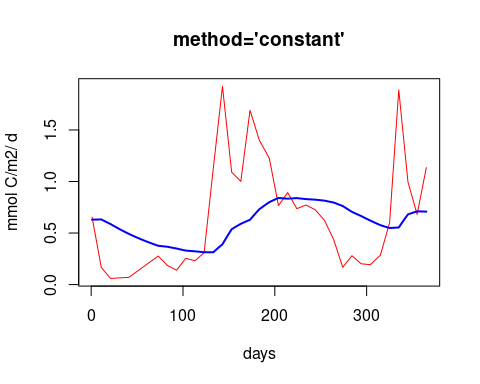
plot(out, which = "Depo", col="red",  
 xlab="days", ylab="mmol C/m2/ d", main="method='constant'")  
lines(out2[,"time"], out2[,"Mineralisation"], lwd=2, col="blue")



###################################################  
### code chunk number 19: figscoc  
###################################################  
par (mfrow=c(1,2))  
plot(out, which = "Depo", col="red",  
 xlab="days", ylab="mmol C/m2/ d", main="method='linear'")  
lines(out[,"time"], out[,"Mineralisation"], lwd=2, col="blue")  
legend("topleft",lwd=1:2,col=c("red","blue"), c("Flux","Mineralisation"))



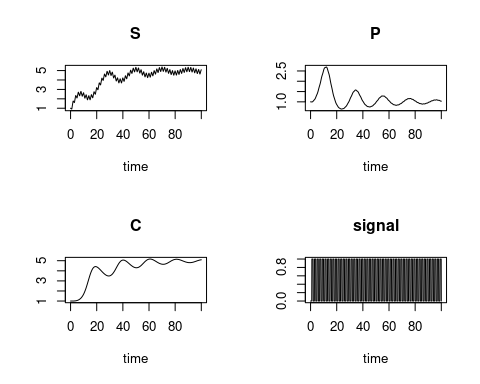
plot(out, which = "Depo", col="red",  
 xlab="days", ylab="mmol C/m2/ d", main="method='constant'")  
lines(out2[,"time"], out2[,"Mineralisation"], lwd=2, col="blue")



###################################################  
### code chunk number 20: compiledCode.Rnw:1360-1392  
###################################################  
SPCmod <- function(t, x, parms, input) {  
 with(as.list(c(parms, x)), {  
 import <- input(t)  
 dS <- import - b\*S\*P + g\*C # substrate  
 dP <- c\*S\*P - d\*C\*P # producer  
 dC <- e\*P\*C - f\*C # consumer  
 res <- c(dS, dP, dC)  
 list(res, signal = import)  
 })  
}  
  
## The parameters  
parms <- c(b = 0.1, c = 0.1, d = 0.1, e = 0.1, f = 0.1, g = 0.0)  
  
## vector of timesteps  
times <- seq(0, 100, by=0.1)  
  
## external signal with several rectangle impulses  
signal <- as.data.frame(list(times = times,  
 import = rep(0, length(times))))  
  
signal$import <- ifelse((trunc(signal$times) %% 2 == 0), 0, 1)  
sigimp <- approxfun(signal$times, signal$import, rule = 2)  
  
## Start values for steady state  
xstart <- c(S = 1, P = 1, C = 1)  
  
## Solve model  
print (system.time(  
 out <- ode(y = xstart,times = times,  
 func = SPCmod, parms, input = sigimp)  
))

## user system elapsed   
## 0.220 0.000 0.224

###################################################  
### code chunk number 21: lv  
###################################################  
plot(out)  
  
  
###################################################  
### code chunk number 22: figlv  
###################################################  
plot(out)



###################################################  
### code chunk number 23: compiledCode.Rnw:1511-1514  
###################################################  
eventdata <- data.frame(var=rep("C",10),time=seq(10,100,10),value=rep(0.5,10),  
 method=rep("multiply",10))  
eventdata

## var time value method  
## 1 C 10 0.5 multiply  
## 2 C 20 0.5 multiply  
## 3 C 30 0.5 multiply  
## 4 C 40 0.5 multiply  
## 5 C 50 0.5 multiply  
## 6 C 60 0.5 multiply  
## 7 C 70 0.5 multiply  
## 8 C 80 0.5 multiply  
## 9 C 90 0.5 multiply  
## 10 C 100 0.5 multiply

###################################################  
### code chunk number 24: compiledCode.Rnw:1601-1619  
###################################################  
derivs <- function(t, y, parms) {  
 with(as.list(c(y, parms)), {  
 if (t < tau)  
 ytau <- c(1, 1)  
 else  
 ytau <- lagvalue(t - tau, c(1, 2))  
   
 dN <- f \* N - g \* N \* P  
 dP <- e \* g \* ytau[1] \* ytau[2] - m \* P  
 list(c(dN, dP), tau=ytau[1], tau=ytau[2])  
 })  
}  
  
yinit <- c(N = 1, P = 1)  
times <- seq(0, 500)  
parms <- c(f = 0.1, g = 0.2, e = 0.1, m = 0.1, tau = .2)  
yout <- dede(y = yinit, times = times, func = derivs, parms = parms)  
head(yout)

## time N P tau tau  
## [1,] 0 1.0000000 1.0000000 1.0000000 1.0000000  
## [2,] 1 0.9119190 0.9228219 0.9277522 0.9378886  
## [3,] 2 0.8441425 0.8502511 0.8562938 0.8643922  
## [4,] 3 0.7924546 0.7823984 0.8016679 0.7955899  
## [5,] 4 0.7537398 0.7192603 0.7605654 0.7315146  
## [6,] 5 0.7256893 0.6607481 0.7305357 0.6720885