R Package ecolMod: figures and examples from Soetaert and Herman (2009)

Karline Soetaert
Centre for Estuarine and Marine Ecology
Netherlands Institute of Ecology
The Netherlands

Abstract

This document contains some examples of the demos from package **ecolMod**.
This package has, in its demo's all the figures of the book:
Soetaert K. and P.M.J. Herman (2009). A Practical Guide to Ecological Modelling. Using R as a Simulation Platform. Springer, 372 pp.

Keywords: ecological Modelling, graphics, figures, examples, R.

This vignette is a Sweave (**?**) application of parts of the demos from package **ecolMod** (**?**), accompanying the book:

Soetaert K. and P.M.J. Herman (2009). A Practical Guide to Ecological Modelling. Using R as a Simulation Platform. Springer, 372 pp.

All figures from this book (>100) were created in **R**. While making these figures, I was not concerned with *how* they are programmed, but rather with the result. Therefore, it is likely that there exist better and more elegant ways to create them.

Some figures make use of packages

- shape (**?**), to draw graphical shapes.
- diagram (**?**), to draw simple graphs, networks, diagrams

These two packages were written in support of the book.

For each chapter of the book, the figures can be displayed by typing

```
demo(chap1)
demo(chap2)
...
demo(chap11)
```

Also, the examples in the book can be found in the `examples` subdirectory of the package.

In this vignette, one (or two) figures for each chapter are created.

1. Chapter 1. Introduction

This figure represents the modelling steps and ingredients.

The figure makes use of package **diagram**. After opening a plot, and positioning of the elements (`elpos`), first the arrows connecting them are drawn (`segmentarrow, straightarrow`) and the texts written in rounded or diamond-shaped boxes `textround, textdiamond`, or as plain text.

```
> par(mar = c(0, 0, 0, 0))
> openplotmat()
> elpos <- coordinates (c(1, 1, 1, 1, 1, 1, 1, 1), mx = -0.1)
> segmentarrow(elpos[7, ], elpos[2, ], arr.pos = 0.15, dd = 0.3, arr.side = 3)
> segmentarrow(elpos[7, ], elpos[3, ], arr.pos = 0.15, dd = 0.3, arr.side = 3)
> segmentarrow(elpos[7, ], elpos[4, ], arr.pos = 0.15, dd = 0.3, arr.side = 3)
> pin <- par ("pin")
> xx  <- 0.2
> yy  <- xx*pin[1]/pin[2]*0.15
> sx    <- rep(xx, 8)
> sx[7] <- 0.05
> sy    <- rep(yy, 8)
> sy[6] <- yy*1.5
> sy[7] <- sx[7]*pin[1]/pin[2]
> for (i in 1:7)
+   straightarrow (from = elpos[i, ], to = elpos[i+1, ], lwd = 2, arr.pos = 0.5)
> lab <- c("Problem", "Conceptual model", "Mathematical model", "Parameterisation",
+          "Mathematical solution", "", "OK?", "Prediction,  Analysis")
> for (i in c(1:6, 8))
+   textround(elpos[i, ], sx[i], sy[i], lab = lab[i])
> textround(elpos[6, ],  xx,  yy*2,
+   lab = c("Calibration, sensitivity", "Verification, validation"))
> textdiamond(elpos[7, ], sx[7],  sy[7],
+   lab = lab[7])
> textplain(c(0.7, elpos[2, 2]),  yy*2,
+   lab = c("main components", "relationships"), font = 3, adj = c(0, 0.5))
> textplain(c(0.7, elpos[3, 2]),  yy ,
+   "general theory", adj = c(0, 0.5), font = 3)
> textplain(c(0.7, elpos[4, 2]),  yy*2,
+   lab = c("literature", "measurements"), font = 3, adj = c(0, 0.5))
> textplain(c(0.7, elpos[6, 2]),  yy*2,
+   lab = c("field data", "lab measurements"), font = 3, adj = c(0, 0.5))
```

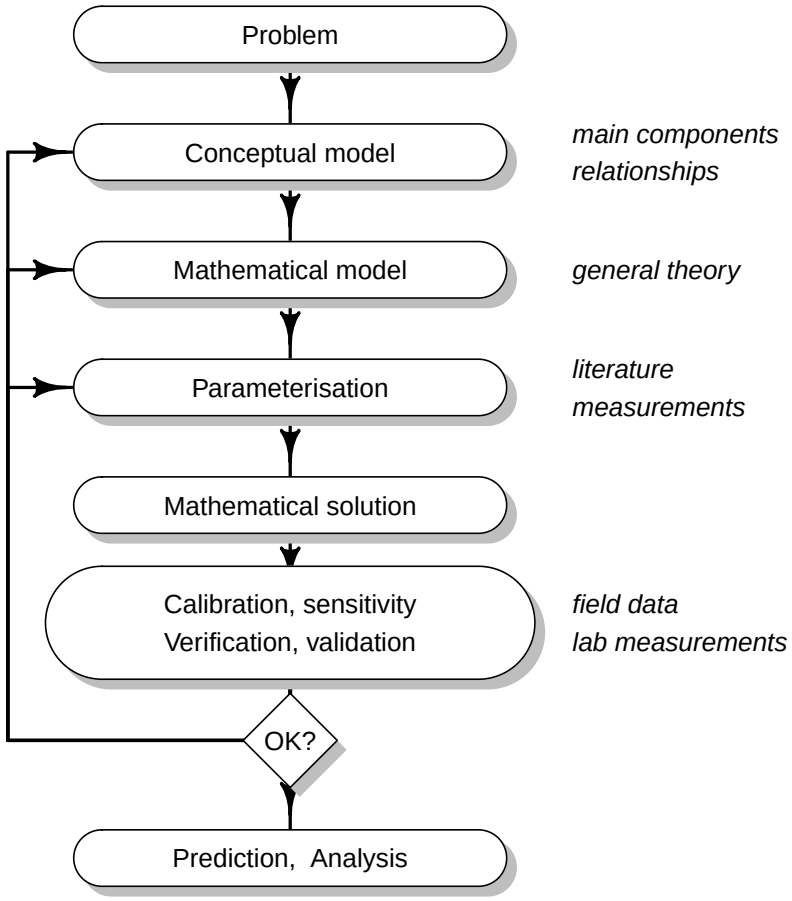

Figure 1: Figure 1.7 from chapter 1

2. Chapter 2. Model formulation

This figure makes use of package **diagram**. It depicts two types of chemical reactions.

```
> par(mfrow = c(1, 2))
> par(mar = c(3, 3, 3, 3))
> # reversible reaction
> openplotmat()
> elpos <- coordinates (c(3, 1))
> treearrow(from = elpos[1:3, ], to = elpos[4, ], arr.side = 2, path = "H")
> treearrow(from = elpos[4, ], to = elpos[1:3, ], arr.side = 2, path = "H")
> labs <- c("C", "D", "D", "E")
> text(0.55, 0.4, expression(k[1]), font = 3, adj = 0, cex = 0.8)
> text(0.55, 0.6, expression(k[2]), font = 3, adj = 0, cex = 0.8)
> for ( i in 1:4)
+   textrect (elpos[i, ], 0.1, 0.1, lab = labs[i], cex = 1.5)
> box(col = "grey")
> title("reversible reaction")
> writelabel("A", line = 0, at = -0.05)
> #
> # enzymatic reaction
> #
> openplotmat()
> elpos <- coordinates (c(3, 2, 3))
> elpos <- elpos[-c(5, 6), ]
> elpos[4, 1] <- 0.3333
> elpos[6, 1] <- 0.7
> treearrow(from = elpos[1:2, ], to = elpos[4, ], arr.side = 2, path = "H")
> treearrow(to = elpos[1:2, ], from = elpos[4, ], arr.side = 2, path = "H")
> treearrow(from = elpos[3:4, ], to = elpos[5:6, ], arr.side = 2, path = "H")
> labs <- c("E", "D", "F", "I", "E", "G")
> for ( i in 1:6)
+   textrect (elpos[i, ], 0.075, 0.07, lab = labs[i], cex = 1.5)
> text(0.35, 0.6, expression(k[1]), font = 3, adj = 0, cex = 0.8)
> text(0.52, 0.7, expression(k[2]), font = 3, adj = 0, cex = 0.8)
> text(0.72, 0.3, expression(k[3]), font = 3, adj = 0, cex = 0.8)
> box(col = "grey")
> title("enzymatic reaction")
> writelabel("B", line = 0, at = -0.05)
```

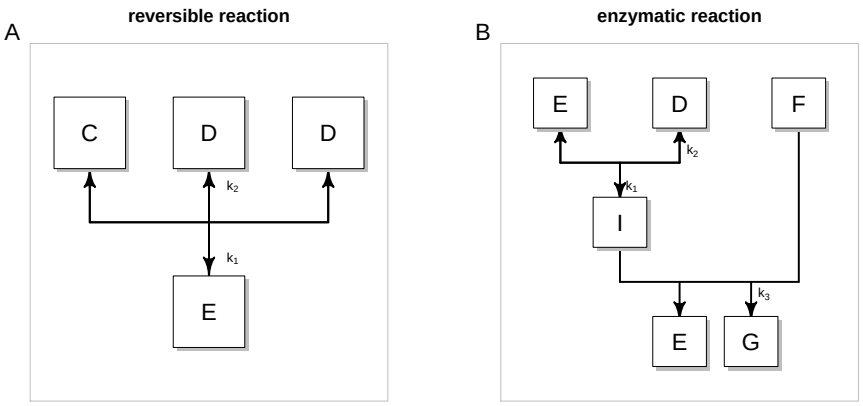

Figure 2: Figure 2.6 from chapter 2

3. chapter 3. Spatial components and transport

This figure makes use of package **shape** to draw a filled cylinder. We use it to derive the general transport equation.

```
> par(mfrow = c(1, 1))
> par(mar = c(1, 1, 1, 1))
> plot(0, type = "n", xlim = c(-1, 1), ylim = c(-0.6, 0.5), axes = FALSE,
+      xlab = "", ylab = "")
> col <- grey(seq(0.2, 1, length.out = 100))
> col <- c(col, rev(col))
> cex <- 1.75
> #
> filledcylinder (rx = 0.15,  ry = 0.4,  len = 1,  col = col,  lcol = "black",
+   lwd = 1,  lcolint = grey(0.25),  lwdint = 1,  ltyint = 3,
+   topcol = grey(0.5),  delt = 1.15)
> #
> segments(-1, 0, -0.5, 0)
> segments(0.5, 0, 1, 0)
> #
> Arrows(-0.8, 0, -0.5, 0, arr.type = "triangle",
+   arr.length = 0.5, lwd = 5, arr.adj = 1)
> Arrows(0.5, 0, 0.8, 0, arr.type = "triangle",
+   arr.length = 0.5, lwd = 3, arr.adj = 1)
> #
> text(0.0, 0.5, expression(Delta~V), cex = cex*0.9)
> text(-0.5, 0.225, expression(A[x]), cex = cex)
> text(0.5, 0.225, expression(A[x+Delta~x]), cex = cex)
> text(-0.75, 0.065, expression(J[x]), cex = cex)
> text(0.85, 0.065, expression(J[x+Delta~x]), cex = cex)
> #
> segments(-0.5, 0, -0.5, -0.5, lty = 3, col = grey(0.25))
> segments(0.5, 0, 0.5, -0.5, lty = 3, col = grey(0.25))
> #
> text(-0.5, -0.55, expression(x), cex = cex)
> text(0.5, -0.55, expression(x+Delta~x), cex = cex)
```

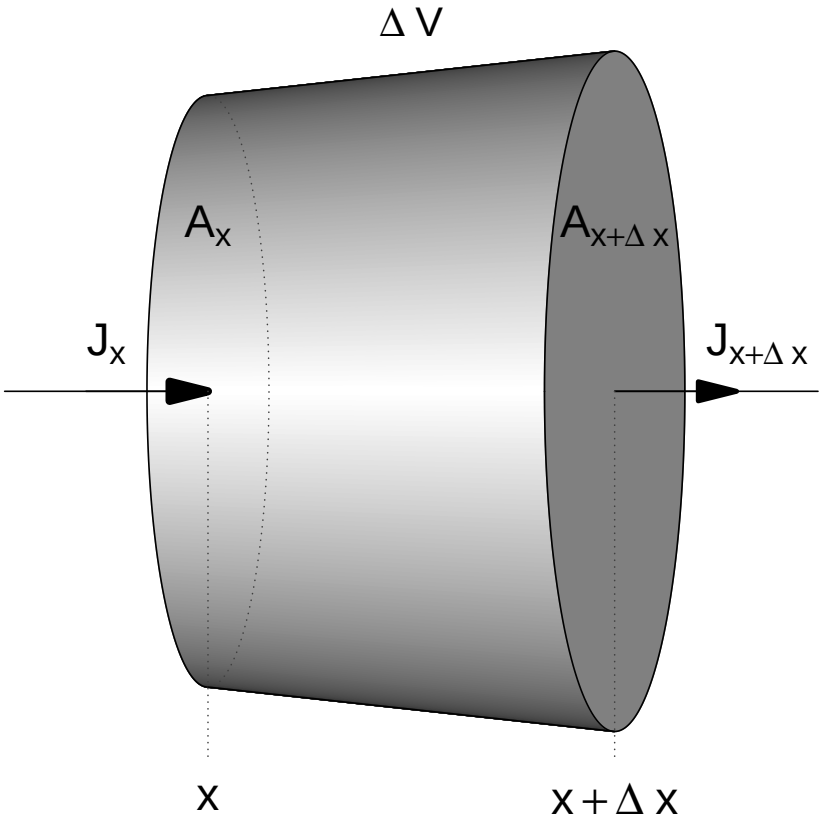

Figure 3: Figure 3.5 from chapter 3

4. chapter 4. Parameterisation

This figure represents a (fictuous) model cost landscape.

The figure has higher resolution in the book. It makes use of R's `persp` function, which has been draped with a grey colorscale (`drapecol`). The latter function is included in package `shape`.

```
> par(mfrow = c(1, 1))
> par(mar = c(1, 1, 1, 1))
> col <- grey(seq(0, 0.9, length.out = 100))
> #
> gg <- outer(seq(3 , 9.5, 0.05), seq(-4.8, -1, 0.05),
+   FUN = function(x, y) 4*cos(y)+sin(x)-(sin(x)/sqrt(x)*cos(y)*y^2)^2)
> #
> persp(gg, col = drapecol(gg, col), border = NA, theta = 30, phi = 30,
+   axes = TRUE, box = TRUE,  lty = 2, xlab = "parameter 1",
+   ylab = "parameter 2", zlab = "cost", ticktype = "simple", cex = 1.5)
> #
> text(0.15, 0.22, "2", cex = 2)
> text(-0.10, 0.27, "1", cex = 2)
> text(-0.15, -0.25, "global minimum")
> text(0.1, -0.18, "local minimum")
```

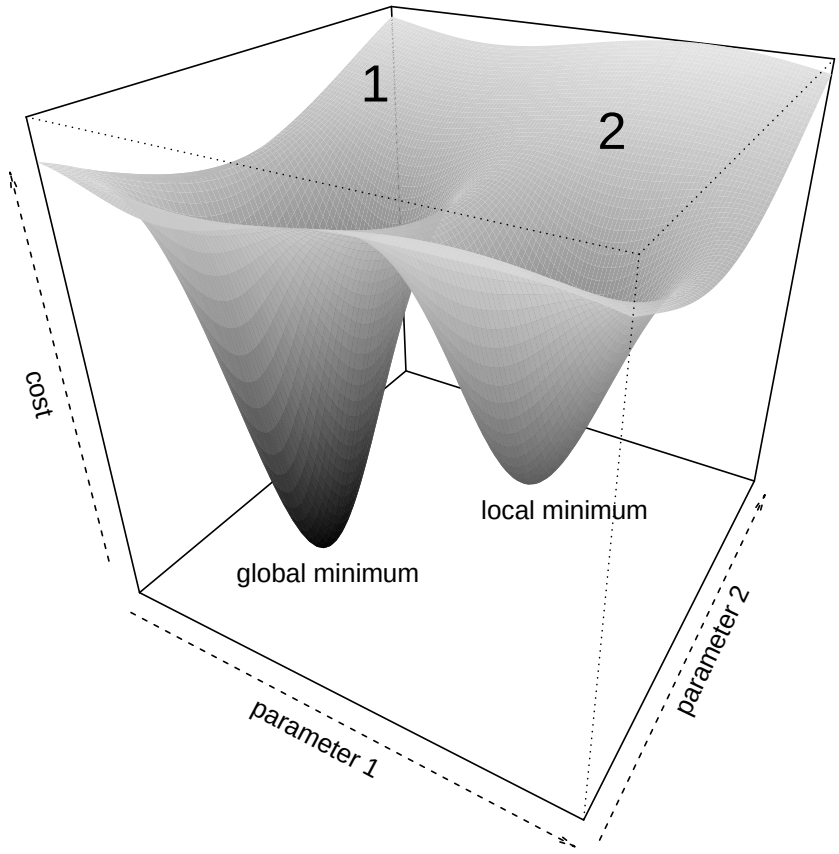

Figure 4: Figure 4.4 from chapter 4

5. chapter 5. Model solution - analytical methods

The analytical solution of a diffusion-reaction equation in 1-D, and using cylindrical coordinates is depicted. It makes use of R s function `persp`, draped with a red-yellow-blue color scheme. Functions `drapecol` and `femmecol` made available in package **shape**.

```
> Ds  <- 1     # diffusion coefficient
> ini <- 1     # initial condition
> k   <- 0.05  # growth rate
> #
>
> plotplane <- function(time,  rmax = 5,  ...) {
+   xx <- seq(-rmax, rmax, length = 50)
+   yy <- xx
+
+   val <- outer(xx,  yy,  FUN  =  function (x, y)
+      ini/(4*pi*Ds*time)*exp(k*time-(x*x+y*y)/(4*Ds*time))  )
+   persp(xx,  yy,  z = val,  theta = 150,  box = TRUE,  axes = TRUE,
+      col = drapecol(val, femmecol(100)), zlab = "Density", border = NA,  ...)
+  }
> #
> par(mfrow = c(2, 2), mar = c(3, 3, 3, 3))
> plotplane(0.1,  main =  "0.1 day")
> plotplane(1  ,  main =  "1 day")
> plotplane(2  ,  main =  "2 days")
> plotplane(5  ,  main =  "5 days")
```

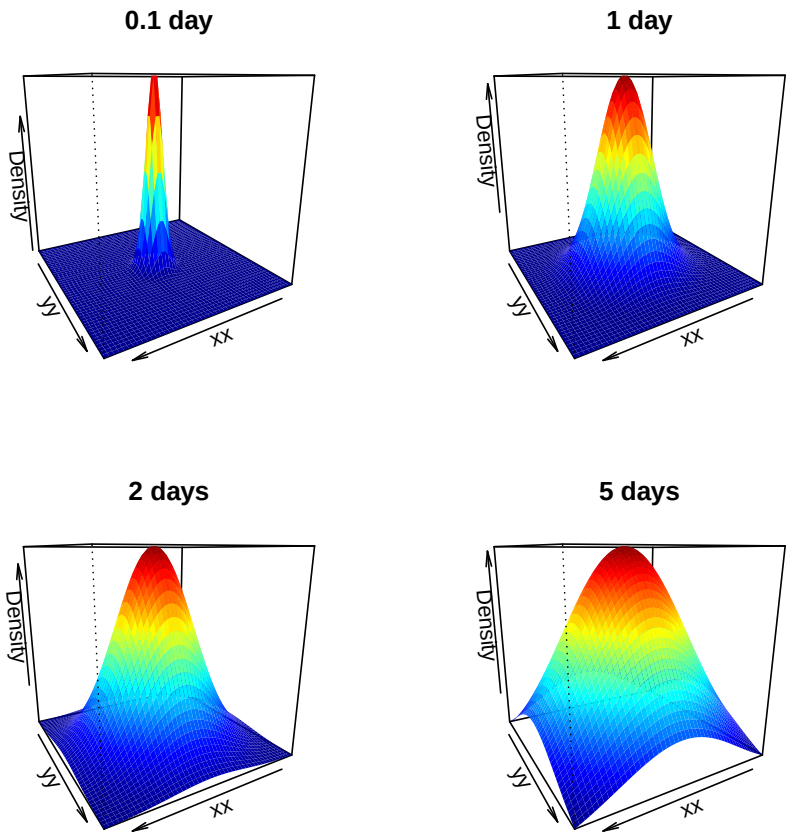

Figure 5: Figure 5.5 from chapter 5

6. chapter 6. Model solution numerical methods

Here, a 1-D diffusion-reaction model is implemented, solved using function `ode.1D` from package **deSolve** (**?**), and the results displayed using R s `filled.contour` function.

```
> model <- function(t, APHIDS, parameters)   {
+     Flux       <- -D*diff(c(0, APHIDS, 0))/deltax
+     dAPHIDS    <- -diff(Flux)/delx  + APHIDS*r
+
+     list(dAPHIDS )
+ }
> #-----------------------#
> # the model parameters: #
> #-----------------------#
>
> D         <- 0.3    # m2/day  diffusion rate
> r         <- 0.01   # /day    net growth rate
> delx      <- 1      # m       thickness of boxes
> numboxes  <- 60
> Distance  <- seq(from = 0.5, by = delx, length.out = numboxes)  # 1 m intervals
> deltax    <- c (0.5, rep(1, numboxes-1), 0.5)
> #--------------------------#
> # Initial conditions:      #
> #--------------------------#
>
> APHIDS          <- rep(0, times = numboxes)  # ind/m2  aphid density
> APHIDS[30:31]   <- 1
> state           <- c(APHIDS = APHIDS)       # initial conditions
> #----------------------#
> # RUNNING the model:   #
> #----------------------#
>
> times     <- seq(0, 200, by = 2)   # output wanted at these time intervals
> out       <- ode.1D(state, times, model, parms = 0,  nspec = 1)
> DENSITY   <- out[, 2:(numboxes  +1)]
> #------------------------#
> # PLOTTING model output: #
> #------------------------#
>
> par(mfrow = c(1, 1))
> par(oma = c(0, 0, 3, 0))   # set outer margin size (oma)
> filled.contour(x = times, y = Distance, DENSITY, color =  topo.colors,
+                xlab = "time,  days",  ylab =  "Distance on plant,  m",
+                main = "Density")
> mtext(outer = TRUE, side = 3, "Aphid model", cex = 1.5)  # margin text
```

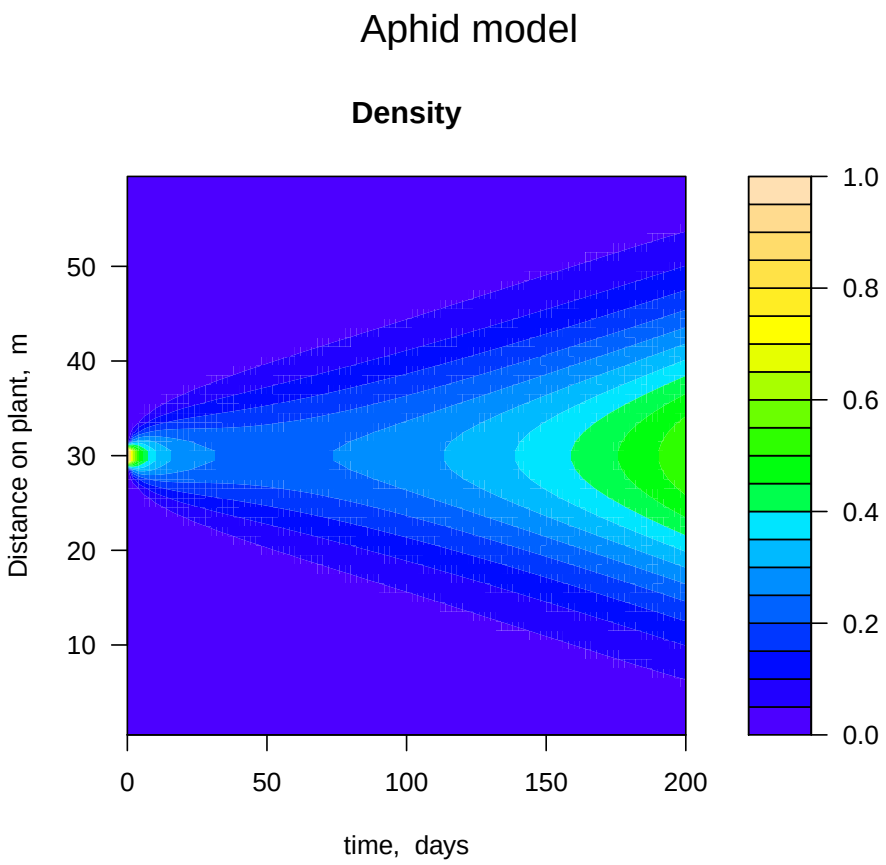

Figure 6: Figure 6.15 from chapter 6

7. chapter 7. Stability and steady state

The trajectories of a set of differential equations, displaying limit cycles are depicted. The model is solved using function `vode` from package **deSolve** (**?**). This is the only integrator that can solve this rather difficult set of equations (although even vode complains).

```
par(mfrow = c(1, 2))
eqn <- function (t, state, pars)  {
  with (as.list(c(state, pars)),
  {
  dx <-  a*y   +e*x*(x^2+y^2-1)
  dy <-  b*x   +f*y*(x^2+y^2-1)
  list(c(dx, dy))
  })
}
#
equation2 <- function(ini, i1 = 5, a = -1, b = 1, e = -1, f = -1,
                      endt = 100, dt = 0.1, ...) {
  times <- seq(0, endt, dt)
  state <- c(x = ini[1], y = ini[2])
  out   <- as.data.frame(vode(state, times, eqn,
                         parms = c(a = a, b = b, e = e, f = f)))
  lines(out$x, out$y)
  Arrows(out$x[i1], out$y[i1], out$x[i1+1], out$y[i1+1], ...)
}
#
xlim <- c(-1.5, 1.5)
ylim <- xlim
#
plot(0, type = "n", xlim = xlim, ylim = ylim, xlab = "", ylab = "",
  axes = FALSE, frame.plot = TRUE)
equation2(c(-0.01, -0.01), i1 = 45)
equation2(c(0.01, 0.01), i1 = 45)
equation2(c(1.1, 1.1), i1 = 5)
equation2(c(-1.1, 1.1), i1 = 5)
equation2(c(-1.1, -1.1), i1 = 5)
equation2(c(1.1, -1.1), i1 = 5)
points(0, 0, pch = 21, cex = 3, bg = "lightgrey", col = "lightgrey")
title("Stable limit cycle")
#
writelabel("A")
#
xlim <- c(-1.5, 1.5)
ylim <- xlim
#
plot(0, type = "n", xlim = xlim, ylim = ylim, xlab = "", ylab = "",
```

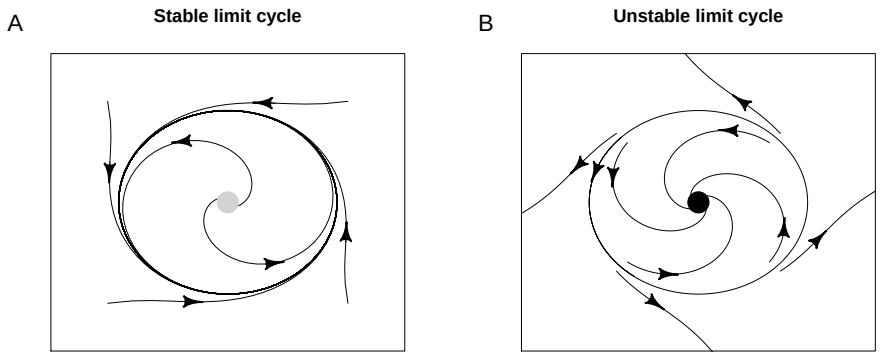

Figure 7: Figure 7.10 from chapter 7

```
  axes = FALSE, frame.plot = TRUE)
equation2(c(-0.65, -0.65), i1 = 45, endt = 10, dt = 0.01, e = 1, f = 1)
equation2(c(0.65, 0.65), i1 = 45, endt = 10, dt = 0.01, e = 1, f = 1)
equation2(c(0.75, 0.75), i1 = 45, endt = 10, dt = 0.01, e = 1, f = 1)
equation2(c(-0.75, -0.75), i1 = 45, endt = 10, dt = 0.01, e = 1, f = 1)
equation2(c(0.65, -0.65), i1 = 45, endt = 10, dt = 0.01, e = 1, f = 1)
equation2(c(0.75, -0.75), i1 = 45, endt = 10, dt = 0.01, e = 1, f = 1)
equation2(c(-0.65, 0.65), i1 = 45, endt = 10, dt = 0.01, e = 1, f = 1)
equation2(c(-.75, 0.75), i1 = 45, endt = 10, dt = 0.01, e = 1, f = 1)
points(0, 0, pch = 21, cex = 3, bg = "black", col = "black")
title("Unstable limit cycle")
equation2(c(-0.71, 0.70420315), i1 = 45, endt = 8,
           dt = 0.01, a = -1, b = 1, e = 1, f = 1)
#
writelabel("B")
#
```

8. chapter 8. Multiple time scales and equilibrium processes

This figures depicts the derivation of the Monod function. It makes use of package **diagram**.

```
> par(mfrow = c(1, 1))
> par(mar = c(1, 1, 1, 1))
> openplotmat()
> rect(0.075, 0.05, 0.575, 0.45,  angle = 45, density = 15,
+   col = "darkgrey", border = NA)
> elpos <- coordinates (c(4, 2, 4), hor = FALSE)
> elpos <- elpos[-c(3, 4, 7, 10), ]
> treearrow(from = elpos[1:2, ], to = elpos[3, ], lty = 1, path = "V")
> treearrow(from = elpos[3, ], to = elpos[1:2, ], lty = 1, path = "V")
> treearrow(from = elpos[3:4, ], to = elpos[5:6, ], lty = 1, path = "V")
> names <- c("A", "E", "EA", "B", "S", "E")
> for ( i in 1:6) textrect (elpos[i, ], 0.06, 0.06, lab = names[i], cex = 1.5)
> text(0.4, 0.28, expression(k^{"+"}))
> text(0.3, 0.15, expression(k^{"-"}))
> text(0.3, 0.4, expression(k^{"-"}))
> text(0.735, 0.4, "r")
> text(0.735, 0.65, "r")
> box(col = "grey")
> par(new = TRUE)
> par(fig = c(0, 0.4, 0.6, 1.0))
> par(mar = c(1, 1, 1, 1))
> openplotmat()
> elpos <- coordinates (c(2, 1), hor = FALSE)
> treearrow(from = elpos[1:2, ], to = elpos[3, ], lty = 1, path = "V")
> names <- c("A", "B", "S")
> for ( i in 1:3)
+   textrect (elpos[i, ], 0.09, 0.09, lab = names[i], cex = 1.5)
> text(0.55, 0.55, expression(r[f]))
> box(col = "grey")
> par(fig = c(0, 1, 0.0, 1))
```

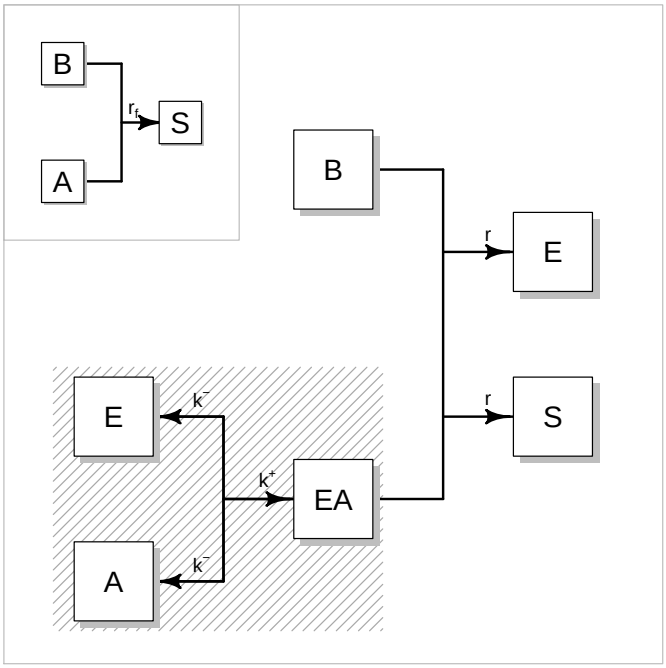

Figure 8: Figure 8.3 from chapter 8

9. chapter 9. Discrete time models

A rather spectacular play with a discrete-time parasitoid model. The figure in the book has more detail. Except for function `writelabel` (from package `shape`), this figure is made only with R's core functions.

```
> par (mfrow = c(2, 2), mar = c(5.1, 4.1, 4.1, 2.1))
> rH <- 2.82   # rate of increase
> tS <- 100    # searching time
> tH <- 1      # handling time
> A  <- tS/tH  # attack rate
> ks <- 30     # 1/tH*a
> Parasite <- function(P_H, ks) {
+  P <- P_H[1]
+  H <- P_H[2]
+  f <- A*P/(ks+H)
+  return(c(H*(1-exp(-f)),
+           H * exp(rH*(1-H)-f)))
+ }
> out <- matrix(nrow = 50, ncol = 2)
> plottraject <- function(ks) {
+   P_H <- c(0.5, 0.5)
+   for (i in 1:100)
+     P_H <- Parasite(P_H, ks)
+   for (i in 1:50) {
+     P_H <- Parasite(P_H, ks)
+     out[i, ] <- P_H
+   }
+
+ plot (out[, 1], type = "l", ylim = range(out), lwd = 2, xlab = "t",
+      ylab = "Population",  main = paste("ks = ", ks))
+ lines(out[, 2], lty = 2)
+ }
> #plottraject(35)
>
>
> plottraject(25)
> writelabel("A")
> plottraject(20)
> writelabel("B")
> legend("topright", c("Parasitoid", "Host"), lty = c(1, 2), lwd = c(2, 1))
> ksSeq <- seq(15, 35, 0.2) # sequence of a-values
> plot(0, 0, xlim = range(ksSeq), ylim = c(0., 2), xlab = "ks",
+   ylab = "Nt", main = "Bifurcation diagram")
> for ( ks in ksSeq) {
+   P_H <- c(0.5, 0.5)
+   for (i in 1:100)
```

```
+     P_H <- Parasite(P_H, ks)   # spinup steps
+   for (i in 1:200)  {
+     P_H <- Parasite(P_H, ks)
+     points(ks, P_H[2], pch = ".", cex = 1.5)
+   }
+ }
> writelabel("C")
> # domain of attraction
> ks   <- 23.09
> dz   <- 0.005 # 0.0025
> xlim <- c(0.001, 0.5)
> ylim <- c(0.001, 0.5)
> Initial <- expand.grid(P  =  seq(xlim[1], xlim[2], dz),
+                        H  =  seq(ylim[1], ylim[2], dz))
> plot(0,  0,  xlim = xlim,  ylim = ylim,  ylab = "Parasitoid initial",
+   xlab = "Host initial",  type = "n",  main = "Domain of attraction")
> PP   <- vector(length = 100)
> for ( ii in 1:nrow(Initial)) {
+   ini <- Initial[ii, ]
+   P_H <- unlist(ini)
+   for (i in 1:100)
+     P_H <- Parasite (P_H, ks)
+   for (i in 1:20) {
+     P_H <- Parasite(P_H, ks)
+     PP[i] <- P_H[1]
+   }
+
+   Freq <- length(unique(trunc(PP*10)))
+   ifelse (Freq  ==  4,  col <- "black",  col <- "white")
+   rect(ini$P-dz/2,  ini$H-dz/2,  ini$P+dz/2,  ini$H+dz/2,
+     col = col,  border = col)
+ }
> writelabel("D")
```

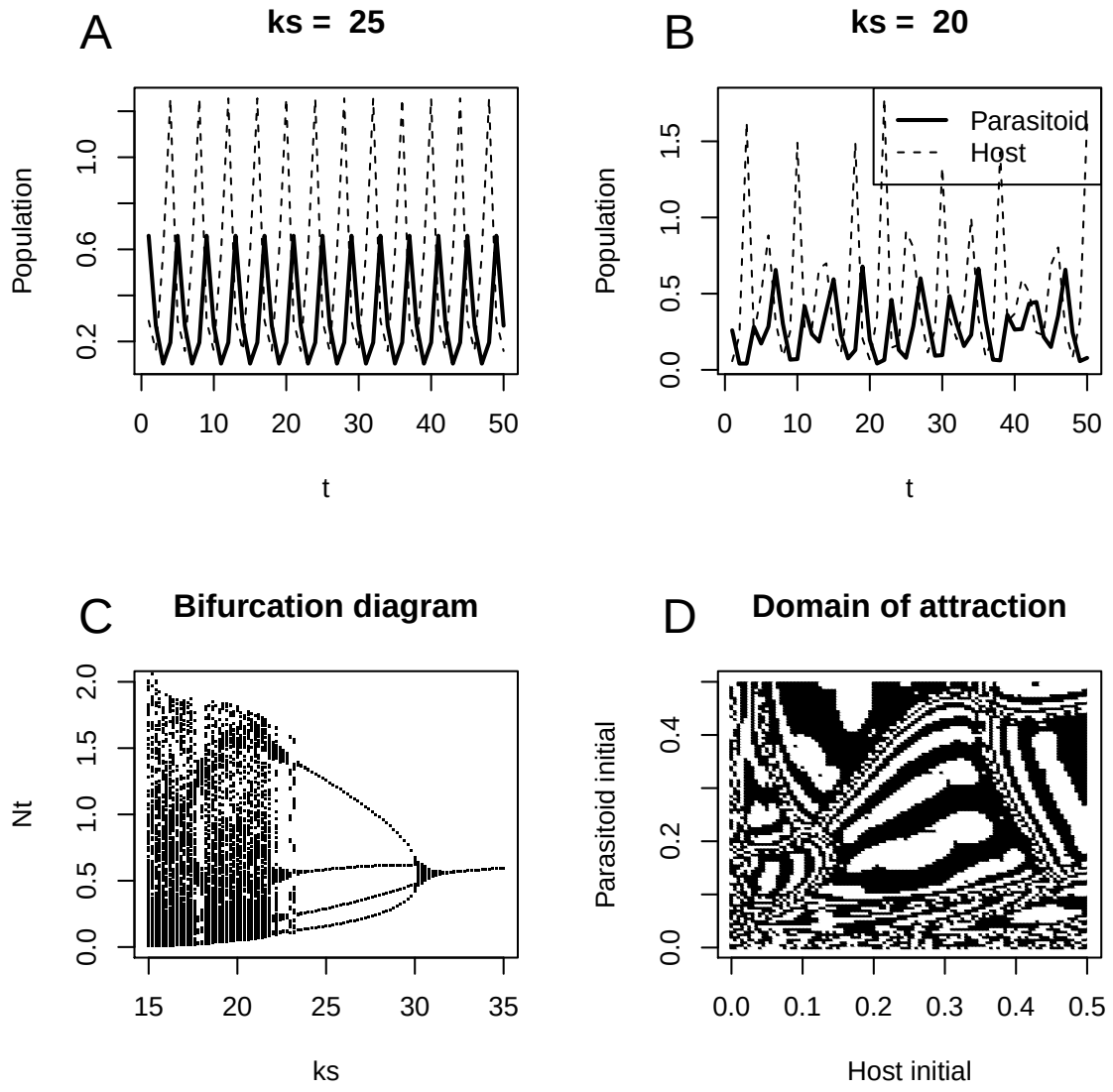

Figure 9: Figure 9.5 from chapter 9

10. chapter 9. Discrete time models, b

As there are no nice figures in chapter 10, I depict two figures from chapter 9.

This figure represents the US population transition matrix. First the transition matrix is created, and printed to the screen:

```
> NumClass  <- 10
> Fecundity <- c(0,       0.00102, 0.08515, 0.30574, 0.40002,
+                0.28061, 0.1526 , 0.0642 , 0.01483, 0.00089)
> Survival  <- c(0.9967 , 0.99837, 0.9978 , 0.99672, 0.99607,
+                0.99472, 0.99240, 0.98867, 0.98274, NA) # survival from i to i+1
> # Population matrix M
> DiffMatrix       <- matrix(data = 0, nrow = NumClass, ncol = NumClass)
> DiffMatrix[1, ]  <- Fecundity
> for (i in 1:(NumClass-1))  DiffMatrix[i+1, i] <- Survival[i]
> DiffMatrix                                  # print the matrix to screen
```

```
       [,1]    [,2]    [,3]    [,4]    [,5]    [,6]   [,7]    [,8]    [,9]   [,10]
 [1,] 0.0000 0.00102 0.08515 0.30574 0.40002 0.28061 0.1526 0.06420 0.01483 0.00089
 [2,] 0.9967 0.00000 0.00000 0.00000 0.00000 0.00000 0.0000 0.00000 0.00000 0.00000
 [3,] 0.0000 0.99837 0.00000 0.00000 0.00000 0.00000 0.0000 0.00000 0.00000 0.00000
 [4,] 0.0000 0.00000 0.99780 0.00000 0.00000 0.00000 0.0000 0.00000 0.00000 0.00000
 [5,] 0.0000 0.00000 0.00000 0.99672 0.00000 0.00000 0.0000 0.00000 0.00000 0.00000
 [6,] 0.0000 0.00000 0.00000 0.00000 0.99607 0.00000 0.0000 0.00000 0.00000 0.00000
 [7,] 0.0000 0.00000 0.00000 0.00000 0.00000 0.99472 0.0000 0.00000 0.00000 0.00000
 [8,] 0.0000 0.00000 0.00000 0.00000 0.00000 0.00000 0.9924 0.00000 0.00000 0.00000
 [9,] 0.0000 0.00000 0.00000 0.00000 0.00000 0.00000 0.0000 0.98867 0.00000 0.00000
[10,] 0.0000 0.00000 0.00000 0.00000 0.00000 0.00000 0.0000 0.00000 0.98274 0.00000
```

Plotting this matrix is relatively simple, using **diagram**s function `plotmat`:

```
> par(mfrow = c(1, 1))
> par(mar = c(2, 2, 2, 2))
> names <- c("0-5yr", "5-10yr", "10-15yr", "15-20yr", "20-25yr",
+        "25-30yr", "30-35yr", "35-40yr", "40-45yr", "45-50yr")
> # first generation in middle; other generations on a circle
> pos <- coordinates(NULL, N = NumClass-1)
> pos <- rbind(c(0.5, 0.5), pos)
> curves <- DiffMatrix
> curves[]   <- -0.4
> curves[1,  ] <- 0
> curves[2, 1] <- -0.125
> curves[1, 2] <- -0.125
> plotmat(DiffMatrix, pos = pos, name = names, curve = curves,
+         box.size = 0.07, arr.type = "triangle", cex.txt = 0.8,
+         box.col = grey(0.95), box.prop  = 1)
> mtext(side = 3, "US population life cycle,  1966", cex = 1.2)
```

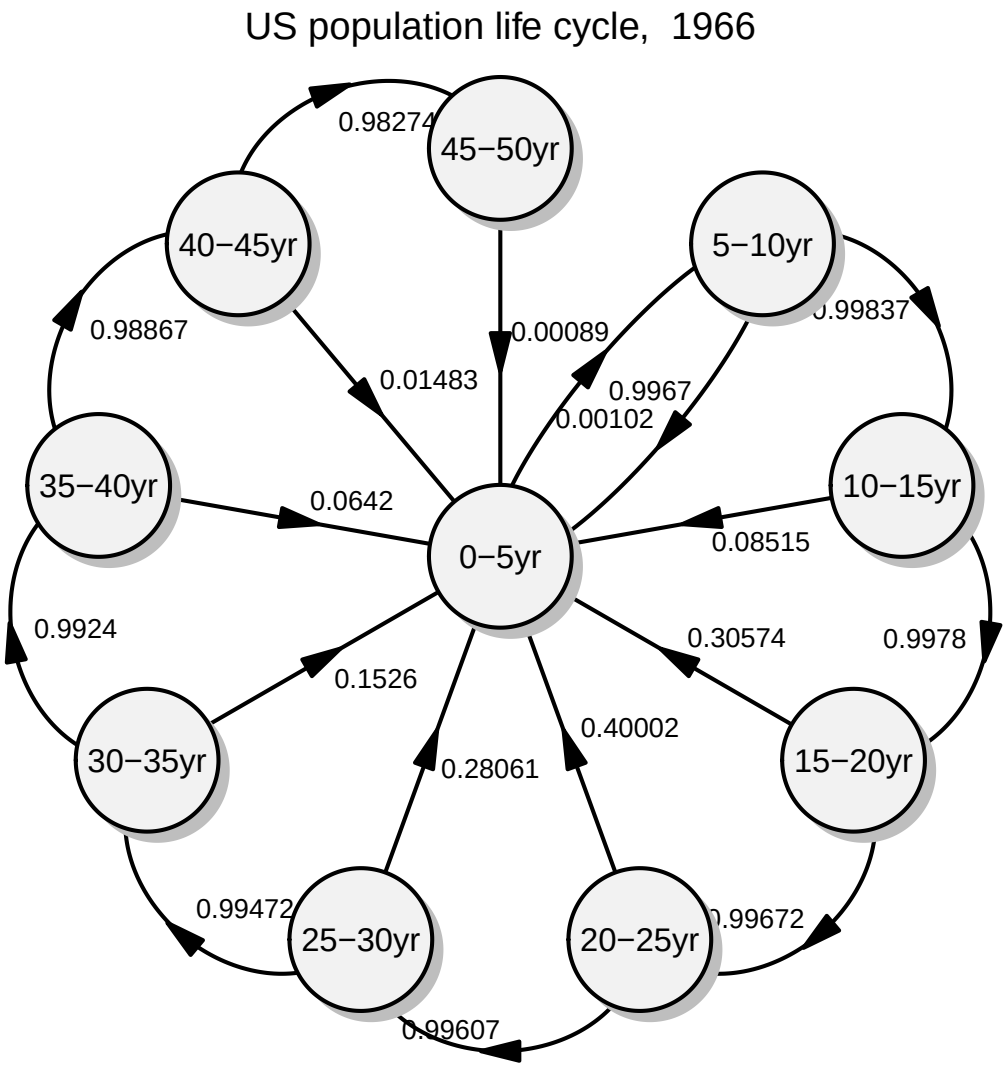

Figure 10: Figure 9.9 from chapter 9

11. chapter 11. Testing and validating the model

In this figure, a BOD (Biochemical Oxygen Demand) model is solved numerically, compared with the analytical solution, improved, and model sensitivity assessed.

```
> par(oma = c(0, 0, 2, 0))
> par(mar = c(5.1, 4.1, 4.1, 2.1))
> par(mfrow = c(2, 2))
> k        =  0.1             # /day     - reaeration
> O2sat    =  300             # mmol/m3  - saturated oxygen concentration
> r        =  0.05            # /day     - BOD decay rate
> O2_0     =  250             # mmol/m3  - Initial oxygen concentration
> BOD_0    =  500             # mmol/m3  - Initial BOD concentration
> ks       =  0               # mmol/m3  - half-saturation concentration
> # numerical model
> numBOD <- function (time, state, pars)  {
+  with (as.list(state),     {
+     dO2  <- -r*BOD*O2/(O2+ks)+k*(O2sat-O2)
+     dBOD <- -r*BOD*O2/(O2+ks)
+     return(list(c(dO2, dBOD)))
+   } )
+ }
> # analytical solution for O2
> analytical <- function(x, k = 0.1, r = 0.05, O2sat = 300)
+  BOD_0*r*(exp(-k*x)-exp(-r*x)) /(k-r)+O2_0*exp(-k*x)+O2sat*(1-exp(-k*x))
> # A comparison numerical / analytical model
> # numerical solution plotted as points
> times <- 0:100
> state <- c(O2 = O2_0, BOD = BOD_0)
> out   <- as.data.frame(ode(state, times, numBOD, 0))
> plot(out$time, out$O2, xlab = "time", ylab = "mmol O2/m3",
+   lwd = 2, main = "Correctness of solution")
> # analytical solution - added as a curve
> curve(analytical(x, k), lty = 1, lwd = 1, add = TRUE)
> legend("bottomright", c("analytic", "numerical"),
+   lwd = c(1, 2), lty = c(1, NA), pch = c(NA, 1))
> writelabel("A")
> # B: internal logic
> # wrong use of model : too low reaeration -> negative concentration
>
> k     <- 0.01
> times <- 0:200
> state <- c(O2 = O2_0, BOD = BOD_0)
> out   <- as.data.frame(ode(state, times, numBOD, 0))
> plot(out$time, out$O2, xlab = "time", ylab = "mmol O2/m3",
+   main = "Internal logic", type = "l", lty = 2)
> abline(h = 0, lty = 3)
```

```
> ks    <- 1
> state <- c(O2 = O2_0, BOD = BOD_0)
> out2  <- as.data.frame(ode(state, times, numBOD, 0))
> lines(out2$time, out2$O2, lwd = 2)
> legend("bottomright", c("no O2 limitation", "O2 limitation"),
+   lwd = c(1, 2), lty = c(2, 1))
> writelabel("B")
> # C: global sensitivity
> k       <- 0.1
> rseq   <- seq(0.0, 0.2, by = 0.002)
> rseq   <- rseq[rseq != k]    # cannot calculate analytical solution for this...
> minO2  <- rseq
> for (i in 1:length(rseq))
+   minO2[i]  <- min(analytical(times, r = rseq[i]))
> plot(rseq, minO2, type = "l", lwd = 2 , xlab = "r,  /day",
+   ylab = "minimum O2,  mmol/m3", main = "global sensitivity")
> writelabel("C")
> mtext(side = 3, outer = TRUE, line = 0, "BOD-O2 model", cex = 1.25, font = 2)
> # D: local sensitivity
>
> times   <- 0:100
> ss      <- 1.1
> kp      <- k * ss         # /day     - reaeration
> O2satp  <- O2sat*ss        # mmol/m3  - saturated oxygen concentration
> rp      <- r*ss           # /day     - BOD decay rate
> ref  <- analytical(times)
> outk <- analytical(times, k = kp)
> outs <- analytical(times, O2sat = O2satp)
> outr <- analytical(times, r = rp)
> outm <- mean(ref)
> ss   <- cbind(k = (outk-ref)/outm/0.1,
+   sat = (outs-ref)/outm/0.1, r = (outr-ref)/outm/0.1)
> plot(times, ref, ylim = range(c(ref, outs)), type = "l", lwd = 2, xlab = "time",
+   ylab = "mmol O2/m3", main = "local sensitivity")
> lines(times, outs, lwd = 2, lty = 2)
> arrseq <- seq(10, 100, 10)#c(10, 30, 50, 70, 90)
> Arrows(times[arrseq], ref[arrseq], times[arrseq], outs[arrseq],
+   arr.len = 0.25, arr.adj = 1)
> legend("topleft", c(expression(O[2]^"*" ==  300),
+   expression(O[2]^"*" ==  330)), lwd = 2, lty = c(1, 2))
> writelabel("D")
> par(new = TRUE)
> par(fig = c(0.7, 0.99, 0.01, 0.35))
> plot(times, ss[, 2], type = "l", lwd = 2,
+    xlab = "", ylab = "", axes = FALSE, frame.plot = TRUE)
> points(times[arrseq], ss[arrseq, 2])
> text(mean(times), diff(range(ss[, 2]))/2, expression(S["i, j"]))
```

```
> #
>
> msqr <- sqrt(colSums(ss*ss)/length(times))
> par(fig = c(0, 1, 0, 1))
```

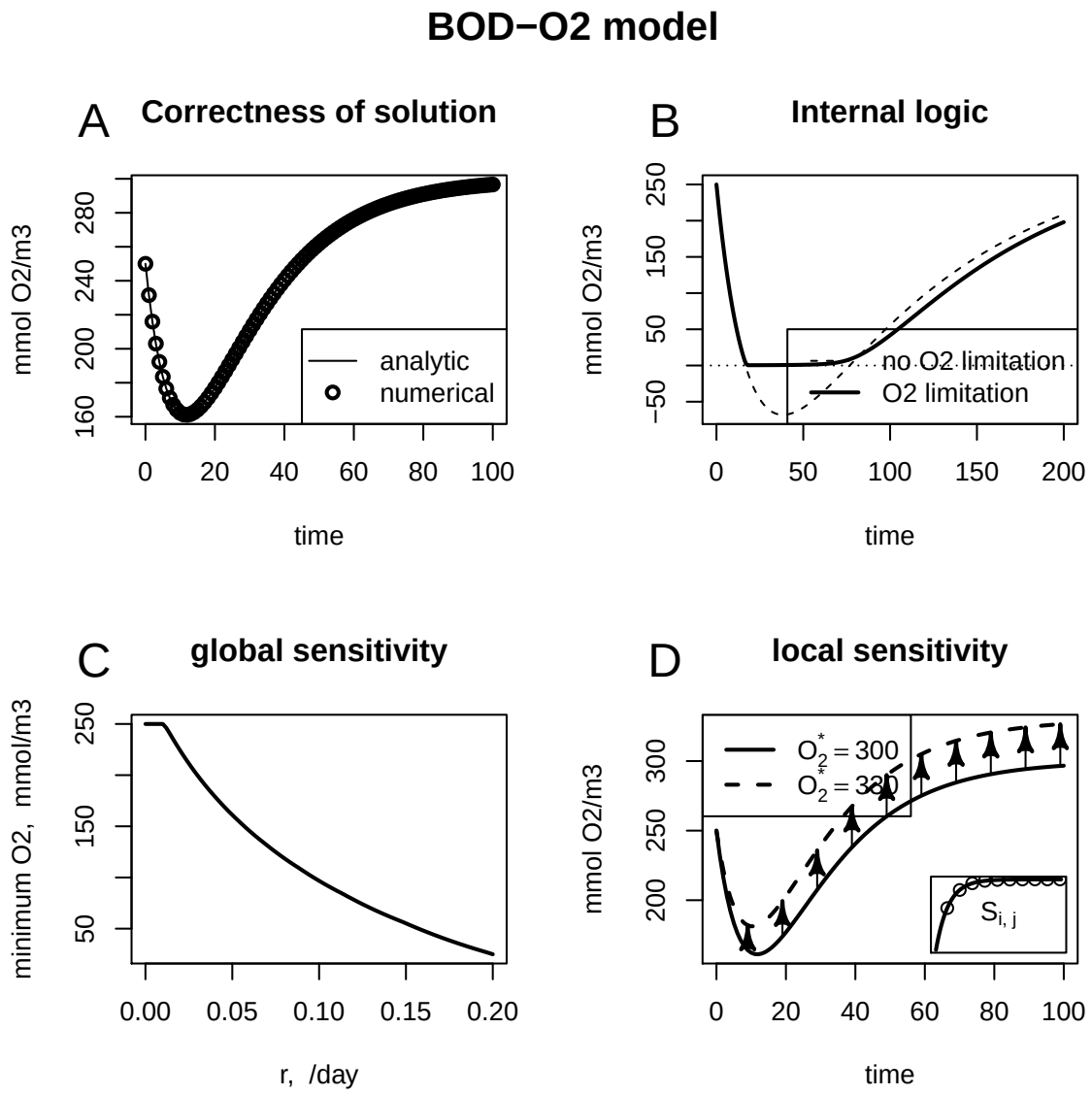

Figure 11: Figure 11.2 from chapter 11

12. And finally

More examples can be found in the demo's of package **ecolMod**.

Affiliation:

Karline Soetaert
Centre for Estuarine and Marine Ecology (CEME)
Netherlands Institute of Ecology (NIOO)
4401 NT Yerseke, Netherlands E-mail: k.soetaert@nioo.knaw.nl
URL: http://www.nioo.knaw.nl/users/ksoetaert