The BOD-O2 model

A template markdown file for a simple dynamic model in 0D

your name here

Date of creation here

Contents

Introduction

This template file contains a simple box model describing the dynamics of molecular oxygen (O_2, in $mol\ m^{-3}$) and biochemical oxygen demand (BOD, in $mol\ m^{-3}$) in a lake. It is assumed that

- O_2 and BOD are removed due to BOD decay (O_2:BOD stoichiometry of 1:1),
- BOD decay is a first-order process with respect to BOD ($r = 0.05\ d^{-1}$) and depends on O_2 according to the Michaelis-Menten kinetics ($kO_2 = 0.001\ mol\ m^{-3}$),
- O_2 is added by air-water exchange ($k = 0.1\ d^{-1}$, $O_{2,sat} = 0.3\ mol\ m^{-3}$),
- DOB is added at a constant rate ($0.001\ mol\ m^{-3}\ d^{-1}$),
- initial concentrations are $O_{2,ini} = 0.25\ mol\ m^{-3}$ and $BOD_{ini} = 0.5\ mol\ m^{-3}$.

Model definition

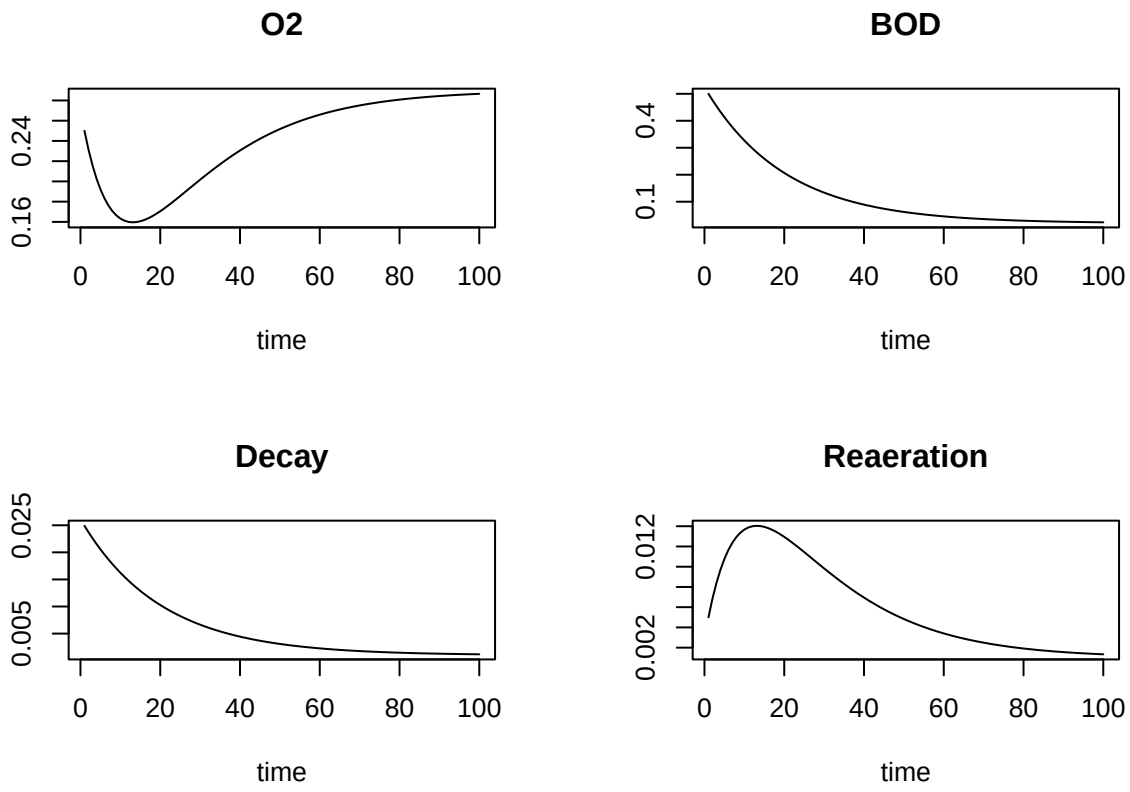
```
# Initial conditions of the state variables
yini <- c(O2 = 0.25, BOD = 0.5) # both in [mol/m3]

# Model parameters
pars <- c(
  rDecay   = 0.05 , # [/d]       first-order rate constant
  kO2      = 0.001, # [mol/m3]   half-saturation O2 concentration
  satO2    = 0.3  , # [mol/m3]   O2 solubility
  k        = 0.1  , # [/d]       reaeration rate constant
  inputBOD = 0.001  # [mol/m3/d] constant BOD input rate
)

# Model function: calculates time-derivatives and other output
BODmodel <-function(t, state, pars) {
  # t: time, state: state variables, pars: model parameters
  with (as.list(c(state, pars)),{

  # rate expressions [mol/m3/d]
    Decay      <- rDecay * BOD * O2/(O2+kO2) # BOD decay
    Reaeration <- k * (satO2-O2)             # air-water exchange

  # Time-derivatives: dC/dt = production - consumption [mol/m3/d]
    dO2dt      <- Reaeration - Decay
    dBODdt     <- inputBOD   - Decay

  # return time-derivatives and ordinary variables as a list
    list(c(dO2dt, dBODdt), # vector with derivatives
                           # (the same order as state variables!)
      Decay      = Decay,  # other output
      Reaeration = Reaeration)
  })
}
```

Model solution

Dynamic solution

We run the model dynamically over 100 days, using two different values of the reaeration rate constant:

```
require(deSolve)  # package with integration methods
# vector of output times
outtimes <- seq(from = 1, to = 100, length.out = 100)
# ode integrates the model
out <- ode(y=yini, parms=pars, func=BODmodel, times=outtimes)
# plot the model output
plot(out)
```

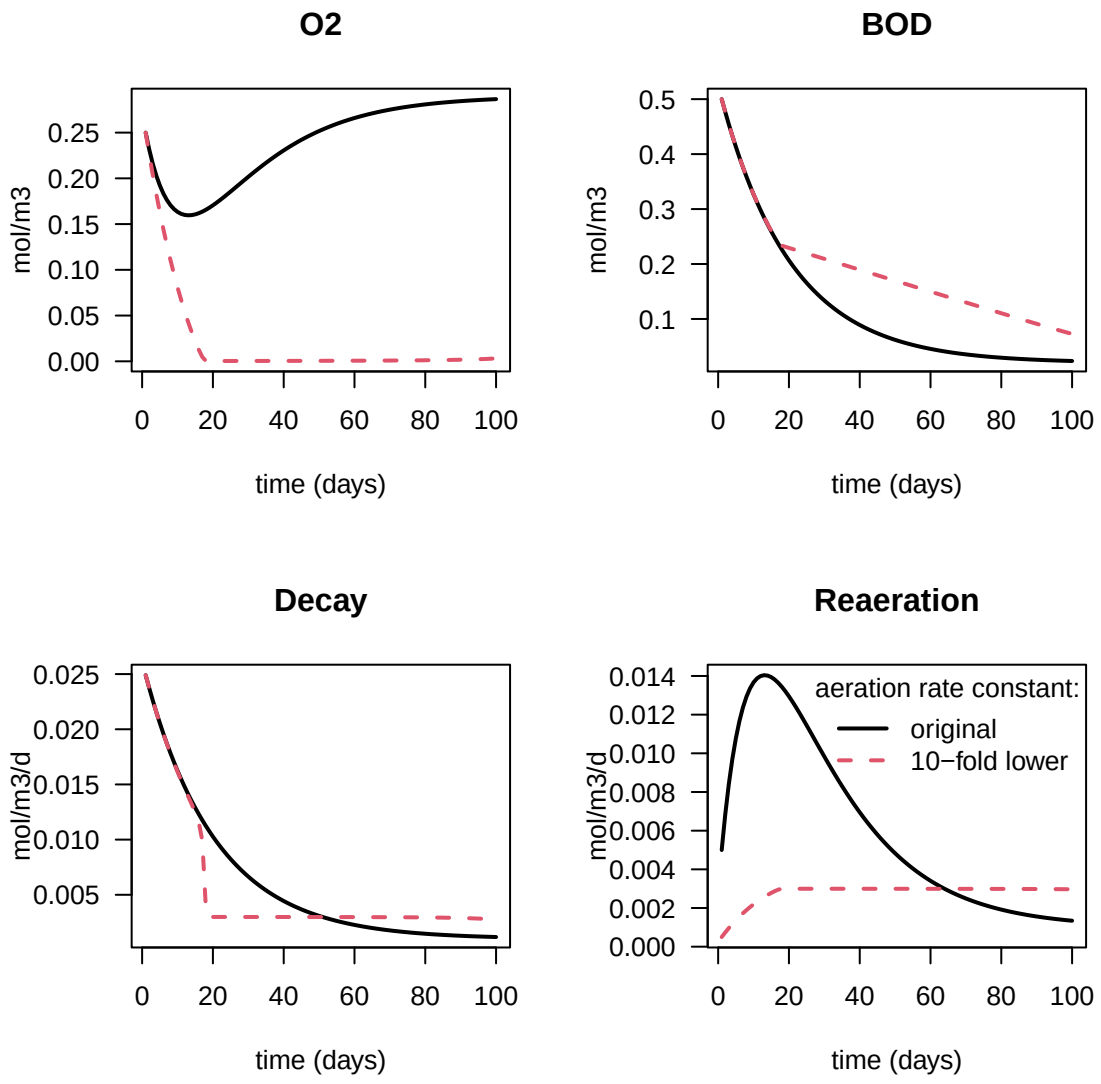



```
# change the value of the reaeration rate constant
pars2      <- pars          # copy the original parameter vector
pars2["k"] <- pars["k"]/10 # 10-fold lower k

# integrate the model with the new parameters
out2 <- ode(y = yini, parms = pars2, func = BODmodel, times = outtimes)
```

```
# print summary of the solution
summary(out)
```

We plot both solutions in one graph:

```
plot(out, out2, xlab="time (days)", las=1, lwd=2,
     ylab=list("mol/m3","mol/m3","mol/m3/d","mol/m3/d"))
legend("topright", legend = c("original", "10-fold lower"),
       title="aeration rate constant:",
       col=1:2, lwd=2, lty=1:2, bty="n")
```



Steady-state solution

We find the steady-state solution:

```
require(rootSolve) # package with solution methods
std <- steady(y = yini, parms = pars, func = BODmodel,
              positive = TRUE) # to ensure that the solution is positive
std$y
```

```
##         O2        BOD
## 0.29000000 0.02006897
```

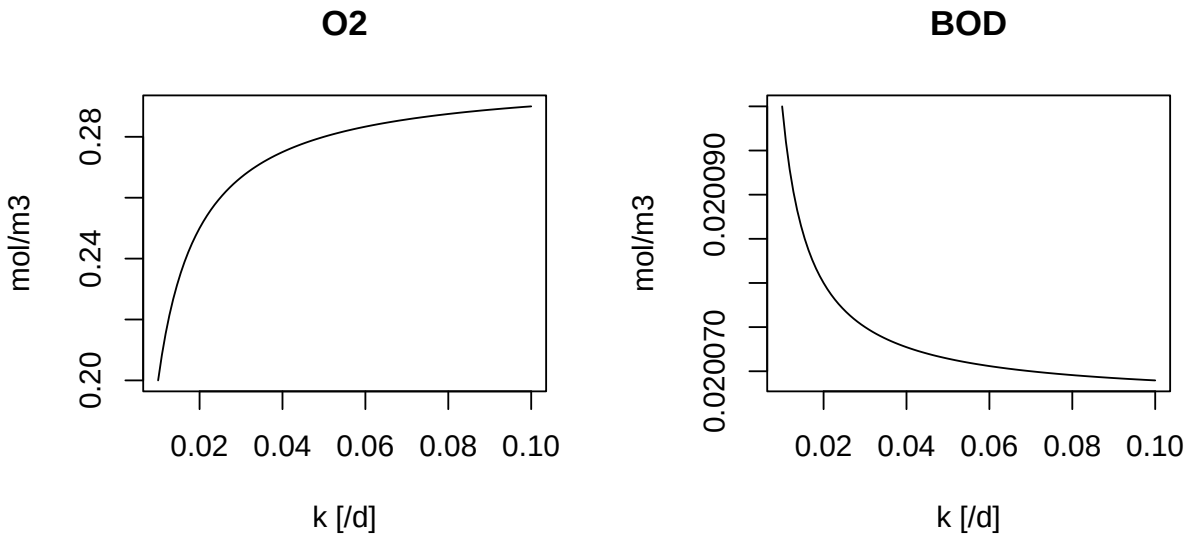
Sensitivity analysis

We perform a sensitivity analysis to find how the steady-state depends on the reaeration rate constant:

```
k.seq   <- seq(from = 0.01, to = 0.1, length.out = 100)
BOD.seq <- vector()  # will contain the results
O2.seq  <- vector()

for (i in 1:length(k.seq)){
  # parameter values for this run
  p <- pars
  p["k"] <- k.seq[i]  # reaeration rate constant based on the sequence
  # steady-state with new parameter values
  std <- steady(y = yini, parms = p, func = BODmodel, positive = TRUE)
  BOD.seq[i] <- std$y["BOD"]
  O2.seq[i]  <- std$y["O2"]
}
```

Finally, we plot the results of the sensitivity analysis:

```
par(mfrow=c(1,2)) # figures in 1 row, 2 columns
plot(k.seq, O2.seq,  type="l", xlab="k [/d]", main="O2",  ylab="mol/m3")
plot(k.seq, BOD.seq, type="l", xlab="k [/d]", main="BOD", ylab="mol/m3")
```



References

R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

Soetaert Karline (2009). rootSolve: Nonlinear root finding, equilibrium and steady-state analysis of ordinary differential equations. R-package version 1.6

Soetaert Karline, Thomas Petzoldt, R. Woodrow Setzer (2010). Solving Differential Equations in R: Package deSolve. Journal of Statistical Software, 33(9), 1–25. http://www.jstatsoft.org/v33/i09/ DOI: 10.18637/jss.v033.i09