

Exercise 4: Biogeochemical-Ecological Lake Model

ETH Zurich Course 701-0426-00L: Modelling Aquatic Ecosystems (Schuwirth)

April 17, 2024

```
# to display chunk code along with its results  
knitr::opts_chunk$set(echo = TRUE)
```

Goals:

- Understand the concepts and process formulations of the coupled biogeochemical-ecological lake model described in section 11.4.
- Understand the implementation of this model based on the R packages `stoichcalc` and `ecosim`.
- Understand the behaviour (time courses of the state variables and overall mass fluxes) of this model.
- Perform some simple sensitivity analyses to deepen the understanding of the behaviour of the model.

Task 1: Model formulation

Study and try to understand the model formulation given in section 11.4. Note that it may be useful to study first the “intermediately complex” model described in section 11.3.

Task 2: Model implementation

Study, complete and run the implementation of the model given below.

Load the package `ecosim` (and install if it is not yet installed).

```
# load required packages:  
  
if ( !require("ecosim") ) {install.packages("ecosim"); library(ecosim) }
```

Note that the packages `deSolve`, to numerically solve systems of ordinary differential equations, and `stoichcalc`, to calculate process stoichiometry, were loaded automatically.

2.1 Define parameters and calculate stoichiometric coefficients Define parameter values for stoichiometry, kinetics, reactor configuration and input:

```
# Define parameter values:  
  
param <- list(alpha.O.ALG      = 0.50,      # gO/gALG  
              alpha.H.ALG     = 0.07,      # gH/gALG  
              alpha.N.ALG     = 0.06,      # gN/gALG  
              alpha.P.ALG     = 0.005,     # gP/gALG
```

alpha.O.ZOO	= 0.50,	# gO/gZOO
alpha.H.ZOO	= 0.07,	# gH/gZOO
alpha.N.ZOO	= 0.06,	# gN/gZOO
alpha.P.ZOO	= 0.01,	# gP/gZOO
alpha.O.POM	= 0.39,	# gO/gPOM
alpha.H.POM	= 0.07,	# gH/gPOM
alpha.N.POM	= 0.06,	# gN/gPOM
alpha.P.POM	= 0.007,	# gP/gPOM
Y.ZOO	= 0.2,	# gZOO/gALG
f.e	= 0.2,	# gPOM/gALG
f.I	= 0.2,	# gPOMI/(gPOMD+gPOMI)
k.gro.ALG	= 0.8,	# 1/d
k.gro.ZOO	= 0.4,	# m3/gDM/d
k.resp.ALG	= 0.10,	# 1/d
k.resp.ZOO	= 0.10,	# 1/d
k.death.ALG	= 0.10,	# 1/d
k.death.ZOO	= 0.05,	# 1/d
k.nitri	= 0.1,	# gN/m3/d
k.miner.ox.POM	= 0.02,	# 1/d
k.miner.ox.POM.sed	= 5.0,	# gDM/m2/d
k.miner.anox.POM.sed	= 5.0,	# gDM/m2/d
K.POM.miner.sed	= 10,	# gDM/m2
K.HPO4	= 0.002,	# gP/m3
K.N	= 0.04,	# gN/m3
p.NH4	= 5,	# -
K.O2.ZOO	= 0.2,	# gO/m3
K.O2.resp	= 0.5,	# gO/m3
K.O2.nitri	= 0.4,	# gO/m3
K.O2.miner	= 0.5,	# gO/m3
K.NO3.miner	= 0.1,	# gN/m3
K.NH4.nitri	= 0.5,	# gN/m3
A	= 5e+006,	# m2
h.epi	= 5,	# m
h.hypo	= 10,	# m
Q.in	= 5,	# m3/s
C.HPO4.in	= 0.04,	# gP/m3
C.NO3.in	= 0.5,	# gN/m3
C.O2.in	= 10,	# gO/m3
C.ALG.ini	= 0.1,	# gDM/m3
C.ZOO.ini	= 0.1,	# gDM/m3
C.POMD.ini	= 0.0,	# gDM/m3
C.POMI.ini	= 0.0,	# gDM/m3
C.HPO4.ini	= 0.04,	# gP/m3
C.NH4.ini	= 0.1,	# gN/m3
C.NO3.ini	= 0.5,	# gN/m3
C.O2.ini	= 10,	# gO/m3
D.POMD.ini	= 0,	# gDM/m2
D.POMI.ini	= 0,	# gDM/m2
beta.ALG	= 0.046,	# 1/degC
beta.ZOO	= 0.08,	# 1/degC
beta.BAC	= 0.046,	# 1/degC
T0	= 20,	# degC
K.I	= 30,	# W/m2

```

lambda.1      = 0.10,      # 1/m
lambda.2      = 0.10,      # m2/gDM
v.ex.O2       = 1,        # m/d
v.sed.POM     = 1,        # m/d
Kz.summer     = 0.02,     # m2/d
Kz.winter     = 20,       # m2/d
h.meta        = 5,        # m
t.max         = 230,      # d
IO.min        = 25,       # W/m2
IO.max        = 225,      # W/m2
T.min         = 5,        # degC
T.max         = 25,       # degC
p             = 101325)   # Pa

```

We now set the parameters for the mass fraction of carbon in ALG, ZOO and POM.

```
# choose carbon fractions to guarantee that the fractions sum to unity:
```

```

param$alpha.C.ALG <- 1 - (param$alpha.O.ALG + param$alpha.H.ALG +
  param$alpha.N.ALG + param$alpha.P.ALG)
param$alpha.C.ZOO <- 1 - (param$alpha.O.ZOO + param$alpha.H.ZOO +
  param$alpha.N.ZOO + param$alpha.P.ZOO)
param$alpha.C.POM <- 1 - (param$alpha.O.POM + param$alpha.H.POM +
  param$alpha.N.POM + param$alpha.P.POM)

```

Next we define the composition of all substances to be considered for the calculation of the stoichiometries of all processes considered in the model. Note that we have to be careful with the units.

```
# Construct composition matrix:
```

```

NH4    <- c(H      = 4*1/14,      # gH/gNH4-N
            N      = 1,          # gN/gNH4-N
            charge = 1/14)       # chargeunits/gNH4-N
NO3    <- c(O      = 3*16/14,     # gO/gNO3-N
            N      = 1,          # gN/gNO3-N
            charge = -1/14)      # chargeunits/gNO3-N
N2     <- c(N      = 1)          # gN/gN2-N
HPO4   <- c(O      = 4*16/31,     # gO/gHPO4-P
            H      = 1*1/31,     # gH/gHPO4-P
            P      = 1,          # gP/gHPO4-P
            charge = -2/31)      # chargeunits/gHPO4-P
HCO3   <- c(C      = 1,          # gC/gHCO3-C
            O      = 3*16/12,    # gO/gHCO3-C
            H      = 1*1/12,    # gH/gHCO3-C
            charge = -1/12)      # chargeunits/gHCO3-C
O2     <- c(O      = 1)          # gO/gO2-O
H      <- c(H      = 1,          # gH/molH
            charge = 1)          # chargeunits/molH
H2O    <- c(O      = 1*16,       # gO/molH2O
            H      = 2*1)       # gH/molH2O
ALG    <- c(C      = param$alpha.C.ALG, # gC/gALG
            O      = param$alpha.O.ALG, # gO/gALG
            H      = param$alpha.H.ALG, # gH/gALG

```

```

      N      = param$alpha.N.ALG,      # gN/gALG
      P      = param$alpha.P.ALG)      # gP/gALG
ZOO  <- c(C  = param$alpha.C.ZOO,      # gC/gZOO
         O  = param$alpha.O.ZOO,      # gO/gZOO
         H  = param$alpha.H.ZOO,      # gH/gZOO
         N  = param$alpha.N.ZOO,      # gN/gZOO
         P  = param$alpha.P.ZOO)      # gP/gZOO
POM  <- c(C  = param$alpha.C.POM,      # gC/gPOM
         O  = param$alpha.O.POM,      # gO/gPOM
         H  = param$alpha.H.POM,      # gH/gPOM
         N  = param$alpha.N.POM,      # gN/gPOM
         P  = param$alpha.P.POM)      # gP/gPOM

```

Once all substances are defined, we can construct a composition matrix by passing the list of substances to the function `calc.comp.matrix` of the package `stoichcalc`:

```

subst.comp <- list(C.NH4   = NH4,
                  C.NO3   = NO3,
                  C.N2    = N2,
                  C.HPO4  = HPO4,
                  C.HCO3  = HCO3,
                  C.O2    = O2,
                  C.H      = H,
                  C.H2O   = H2O,
                  C.ALG   = ALG,
                  C.ZOO   = ZOO,
                  C.POMD  = POM,
                  D.POMD  = POM,
                  C.POMI  = POM,
                  D.POMI  = POM)

alpha <- calc.comp.matrix(subst.comp)
print(round(alpha,3))

```

We now need to define yields for the death of ALG and ZOO, can you explain why? *Hint*: Look at the section 8.3 of the manuscript.

```

# choose yield of death to guarantee that no nutrients are required
# (oxygen content of POM was reduced to avoid need of oxygen):

param$Y.ALG.death <- min(1,
                        param$alpha.N.ALG/param$alpha.N.POM,
                        param$alpha.P.ALG/param$alpha.P.POM,
                        param$alpha.C.ALG/param$alpha.C.POM)

param$Y.ZOO.death <- min(1,
                        param$alpha.N.ZOO/param$alpha.N.POM,
                        param$alpha.P.ZOO/param$alpha.P.POM,
                        param$alpha.C.ZOO/param$alpha.C.POM)

print(param$Y.ALG.death)
print(param$Y.ZOO.death)

```

Having defined the composition matrix and the stoichiometric parameters, we can define the stoichiometries of the processes to be considered by the model.

```

# Calculate stoichiometric coefficients of selected processes:

# Growth of algae on ammonium:
nu.gro.ALG.NH4 <-
  calc.stoich.coef(alpha      = alpha,
                  name       = "gro.ALG.NH4",
                  subst      = c("C.NH4", "C.HPO4", "C.HCO3", "C.O2",
                                "C.H", "C.H2O", "C.ALG"),
                  subst.norm  = "C.ALG",
                  nu.norm    = 1)

# Growth of algae on nitrate:
nu.gro.ALG.NO3 <-
  calc.stoich.coef(alpha      = alpha,
                  name       = "gro.ALG.NO3",
                  subst      = c("C.NO3", "C.HPO4", "C.HCO3", "C.O2",
                                "C.H", "C.H2O", "C.ALG"),
                  subst.norm  = "C.ALG",
                  nu.norm    = 1)

# Respiration of algae:
nu.resp.ALG <-
  calc.stoich.coef(alpha      = alpha,
                  name       = "resp.ALG",
                  subst      = c("C.NH4", "C.HPO4", "C.HCO3", "C.O2",
                                "C.H", "C.H2O", "C.ALG"),
                  subst.norm  = "C.ALG",
                  nu.norm    = -1)

# Death of algae:
nu.death.ALG <-
  calc.stoich.coef(alpha      = alpha,
                  name       = "death.ALG",
                  subst      = c("C.NH4", "C.HPO4", "C.HCO3", "C.O2",
                                "C.H", "C.H2O", "C.ALG", "C.POMD", "C.POMI"),
                  subst.norm  = "C.ALG",
                  nu.norm    = -1,
                  constraints = list(c("C.ALG" = param$Y.ALG.death,
                                      "C.POMD" = 1,
                                      "C.POMI" = 1),
                                   c("C.POMD" = -param$f.I,
                                      "C.POMI" = 1-param$f.I)))

# Growth of zooplankton:
nu.gro.ZOO <-
  calc.stoich.coef(alpha      = alpha,
                  name       = "gro.ZOO",
                  subst      = c("C.NH4", "C.HPO4", "C.HCO3", "C.O2", "C.H",

```

```

                                "C.H2O", "C.ALG", "C.ZOO", "C.POMD", "C.POMI"),
subst.norm = "C.ZOO",
nu.norm    = 1,
constraints = list(c("C.ZOO" = 1,
                    "C.ALG" = param$Y.ZOO),
                  c("C.POMD" = 1,
                    "C.POMI" = 1,
                    "C.ALG" = param$f.e),
                  c("C.POMD" = -param$f.I,
                    "C.POMI" = 1-param$f.I)))

# Respiration of zooplankton:
nu.resp.ZOO <-
  calc.stoich.coef(alpha      = alpha,
                   name       = "resp.ZOO",
                   subst      = c("C.NH4", "C.HPO4", "C.HCO3", "C.O2",
                                   "C.H", "C.H2O", "C.ZOO"),
                   subst.norm = "C.ZOO",
                   nu.norm    = -1)

# Death of zooplankton:
nu.death.ZOO <-
  calc.stoich.coef(alpha      = alpha,
                   name       = "death.ZOO",
                   subst      = c("C.NH4", "C.HPO4", "C.HCO3", "C.O2",
                                   "C.H", "C.H2O", "C.ZOO", "C.POMD", "C.POMI"),
                   subst.norm = "C.ZOO",
                   nu.norm    = -1,
                   constraints = list(c("C.ZOO" = param$Y.ZOO.death,
                                       "C.POMD" = 1,
                                       "C.POMI" = 1),
                                     c("C.POMD" = -param$f.I,
                                       "C.POMI" = 1-param$f.I)))

# Nitrification:
nu.nitri <-
  calc.stoich.coef(alpha      = alpha,
                   name       = "nitri",
                   subst      = c("C.NH4", "C.NO3", "C.O2", "C.H", "C.H2O"),
                   subst.norm = "C.NH4",
                   nu.norm    = -1)

# Oxidic mineralization of suspended organic particles:
nu.miner.ox.POM <-
  calc.stoich.coef(alpha      = alpha,
                   name       = "miner.ox.POM",
                   subst      = c("C.NH4", "C.HPO4", "C.HCO3", "C.O2",
                                   "C.H", "C.H2O", "C.POMD"),
                   subst.norm = "C.POMD",

```

```

        nu.norm      = -1)

# Oxidic mineralization of sedimented organic particles:

nu.miner.ox.POM.sed <-
  calc.stoich.coef(alpha      = alpha,
                   name       = "miner.ox.POM.sed",
                   subst      = c("C.NH4", "C.HPO4", "C.HCO3", "C.O2",
                                   "C.H", "C.H2O", "D.POMD"),
                   subst.norm = "D.POMD",
                   nu.norm    = -1)

# Anoxic mineralization of sedimented organic particles:

nu.miner.anox.POM.sed <-
  calc.stoich.coef(alpha      = alpha,
                   name       = "miner.anox.POM.sed",
                   subst      = c("C.NH4", "C.NO3", "C.HPO4", "C.HCO3", "C.N2",
                                   "C.H", "C.H2O", "D.POMD"),
                   subst.norm = "D.POMD",
                   nu.norm    = -1,
                   constraints = list(c("C.NO3" = 1,
                                        "C.N2"  = 1)))

# Sedimentation of degradable organic particles:

nu.sed.POMD <-
  calc.stoich.coef(alpha      = alpha,
                   name       = "sed.POMD",
                   subst      = c("C.POMD", "D.POMD"),
                   subst.norm = "C.POMD",
                   nu.norm    = -1)

# Sedimentation of inert organic particles:

nu.sed.POMI <-
  calc.stoich.coef(alpha      = alpha,
                   name       = "sed.POMI",
                   subst      = c("C.POMI", "D.POMI"),
                   subst.norm = "C.POMI",
                   nu.norm    = -1)

```

Finally, we bind the stoichiometries together for the full stoichiometric matrix. Check the signs of the coefficients. They show which process produces or consumes which substance, does it make biological sense?

```

# Construct stoichiometric matrix:

nu <- rbind(nu.gro.ALG.NH4,
            nu.gro.ALG.NO3,
            nu.resp.ALG,
            nu.death.ALG,
            nu.gro.ZOO,
            nu.resp.ZOO,

```

```

    nu.death.Z00,
    nu.nitri,
    nu.miner.ox.POM,
    nu.miner.ox.POM.sed,
    nu.miner.anox.POM.sed,
    nu.sed.POMD,
    nu.sed.POMI)
print(round(nu,3))

```

2.2 Define the transformation processes and the final system We can now proceed with the definition of transformation processes. This is analogous as we did it in exercises 1 and 2, with the exception that we can use the stoichiometries as calculated above with `stoichcalc`.

First, we define the growth, respiration and death of ALG and Z00.

```

# Definition of transformation processes:

# Growth of algae on ammonium:

gro.ALG.NH4 <-
  new(Class = "process",
       name = "gro.ALG.NH4",
       rate = expression(k.gro.ALG
                         *exp(beta.ALG*(T-T0))
                         *log((K.I+I0)/
                              (K.I+I0*exp(-(lambda.1+lambda.2*C.ALG)*h.epi))))
                              /((lambda.1+lambda.2*C.ALG)*h.epi)
                         *min(C.HPO4/(K.HPO4+C.HPO4),
                              (C.NH4+C.NO3)/(K.N+C.NH4+C.NO3))
                         *(p.NH4*C.NH4/(p.NH4*C.NH4+C.NO3))
                         *C.ALG),
       stoich = as.list(nu["gro.ALG.NH4",]))

# Growth of algae on nitrate:

gro.ALG.NO3 <-
  new(Class = "process",
       name = "gro.ALG.NO3",
       rate = expression(k.gro.ALG
                         *exp(beta.ALG*(T-T0))
                         *log((K.I+I0)/
                              (K.I+I0*exp(-(lambda.1+lambda.2*C.ALG)*h.epi))))
                              /((lambda.1+lambda.2*C.ALG)*h.epi)
                         *min(C.HPO4/(K.HPO4+C.HPO4),
                              (C.NH4+C.NO3)/(K.N+C.NH4+C.NO3))
                         *(C.NO3/(p.NH4*C.NH4+C.NO3))
                         *C.ALG),
       stoich = as.list(nu["gro.ALG.NO3",]))

# Respiration of algae:

resp.ALG <-
  new(Class = "process",

```



```

name = "resp.ALG",
rate = expression(k.resp.ALG*exp(beta.ALG*(T-T0))*
                  (C.O2/(K.O2.resp+C.O2))*C.ALG),
stoich = as.list(nu["resp.ALG",])

# Death of algae:

death.ALG <-
  new(Class = "process",
       name = "death.ALG",
       rate = expression(k.death.ALG*C.ALG),
       stoich = as.list(nu["death.ALG",]))

# Growth of zooplankton:

gro.ZOO <-
  new(Class = "process",
       name = "gro.ZOO",
       rate = expression(k.gro.ZOO*exp(beta.ZOO*(T-T0))
                         *(C.O2/(K.O2.ZOO+C.O2))
                         *C.ALG
                         *C.ZOO),
       stoich = as.list(nu["gro.ZOO",]))

# Respiration of zooplankton:

resp.ZOO <-
  new(Class = "process",
       name = "resp.ZOO",
       rate = expression(k.resp.ZOO*exp(beta.ZOO*(T-T0))*
                         (C.O2/(K.O2.resp+C.O2))*C.ZOO),
       stoich = as.list(nu["resp.ZOO",]))

# Death of zooplankton:

death.ZOO <-
  new(Class = "process",
       name = "death.ZOO",
       rate = expression(k.death.ZOO*C.ZOO),
       stoich = as.list(nu["death.ZOO",]))

```

In a second step, we define additional processes:

- nitrification,
- oxic mineralization of POMD and SPOMD,
- anoxic mineralization of SPOMD.

Complete the missing equations for the expression of the rates based on Table 11.10.

```
# Nitrification: TO BE COMPLETED
```

```

nitri <-
  new(Class = "process",
      name  = "nitri",
      rate  = expression(...),
      stoich = as.list(nu["nitri",]))

# Oxic mineralization of suspended organic particles: TO BE COMPLETED

miner.ox.POM <-
  new(Class = "process",
      name  = "miner.ox.POM",
      rate  = expression(...),
      stoich = as.list(nu["miner.ox.POM",]))

# Oxic mineralization of sedimented organic particles: TO BE COMPLETED

miner.ox.POM.sed <-
  new(Class = "process",
      name  = "miner.ox.POM.sed",
      rate  = expression(...),
      stoich = as.list(nu["miner.ox.POM.sed",]),
      pervol = F)

# Anoxic mineralization of sedimented organic particles: TO BE COMPLETED

miner.anox.POM.sed <-
  new(Class = "process",
      name  = "miner.anox.POM.sed",
      rate  = expression(...),
      stoich = as.list(nu["miner.anox.POM.sed",]),
      pervol = F)

```

You can observe that the only difference between the oxic and anoxic mineralization of sedimented particulated matter is a quadratic term. Can you explain why? *Hint*: Read the explanation given in section 11.4 of the manuscript.

As a last step, we define the processes of sedimentation of POMD and POMI.

```

# Sedimentation of degradable organic particles:

sed.POMD <-
  new(Class = "process",
      name  = "sed.POMD",
      rate  = expression(v.sed.POM/h.hypo*C.POMD),
      stoich = as.list(nu["sed.POMD",]))

# Sedimentation of inert organic particles:

sed.POMI <-
  new(Class = "process",
      name  = "sed.POMI",
      rate  = expression(v.sed.POM/h.hypo*C.POMI),
      stoich = as.list(nu["sed.POMI",]))

```

We now define the seasonally varying light intensity and temperature and, as a consequence of temperature variation, the seasonally varying saturation concentration of dissolved oxygen. In addition, we define the turbulent diffusion coefficient, K_z , that quantifies mixing between the hypolimnion (deep water) and the epilimnion (upper part of the water column). To describe, mixing during the winter and stratification in the summer, we use a large value in winter and a small value in summer.

```
# Definition of environmental conditions:

cond.epi <- list(I0      = expression( 0.5*(I0.min+I0.max)
                                     +0.5*(I0.max-I0.min)
                                     *cos(2*pi/365.25*(t-t.max))),
               T        = expression( 0.5*(T.min+T.max)
                                     +0.5*(T.max-T.min)
                                     *cos(2*pi/365.25*(t-t.max))),
               C.O2.sat = expression(exp(7.7117-1.31403*log(T+45.93))
                                     *p/101325))

cond.hypo <- list(I0 = 0,
                 T  = 5)

cond.gen <- list(Kz=expression( 0.5*(Kz.summer+Kz.winter)
                               -0.5*(Kz.winter-Kz.summer)
                               *sign(cos(2*pi/365.25*(t-t.max))+0.4)))

# Plot the environmental conditions

t <- 1:365

Kz <- numeric(0)
I0 <- numeric(0)
T  <- numeric(0)
C.O2.sat <- numeric(0)

for(i in 1:length(t))
{
  Kz[i]      <- eval(cond.gen$Kz,  envir=c(param, t=t[i]))
  I0[i]      <- eval(cond.epi$I0,  envir=c(param, t=t[i]))
  T[i]       <- eval(cond.epi$T,   envir=c(param, t=t[i]))
  C.O2.sat[i] <- eval(cond.epi$C.O2.sat, envir=c(param, T=T[i]))
}

#Plots
par.def <- par(no.readonly=TRUE) # store current plotting parameter
par(mfrow=c(2,2),
    xaxs="i",
    yaxs="i",mar=c(4.5,4.5,2,1.5)+0.1) # modify plot parameters
plot(t, Kz , ylim=c(0,25))
plot(t, I0, type="l")
plot(t, T, type="l")
plot(t, C.O2.sat, type="l")
par(par.def) # restore previous plotting parameters
```

We now define the two boxes that allow us to describe mixed and stratified conditions of the lake. Inflow and outflow are into and out of the upper box, called epilimnion. In the lower box, called hypolimnion,

there is no primary production due to the absence of light. Also the sediment is assumed to be only relevant in the hypolimnion, which is typical for deep lakes with steep banks.

```

# Definition of reactors:

# Epilimnion:

epilimnion <-
  new(Class      = "reactor",
       name      = "Epi",
       volume.ini = expression(A*h.epi),
       conc.pervol.ini = list(C.HPO4 = expression(C.HPO4.ini), # gP/m3
                              C.NH4  = expression(C.NH4.ini), # gN/m3
                              C.NO3  = expression(C.NO3.ini), # gN/m3
                              C.O2   = expression(C.O2.ini), # gO/m3
                              C.ALG   = expression(C.ALG.ini), # gDM/m3
                              C.ZOO   = expression(C.ZOO.ini), # gDM/m3
                              C.POMD  = expression(C.POMD.ini), # gDM/m3
                              C.POMI  = expression(C.POMI.ini)), # gDM/m3
       input      = list(C.O2 = expression(v.ex.O2*A
                                           *(C.O2.sat-C.O2))), # gas ex.
       inflow     = expression(Q.in*86400), # m3/d
       inflow.conc = list(C.HPO4 = expression(C.HPO4.in),
                          C.NH4  = 0,
                          C.NO3  = expression(C.NO3.in),
                          C.O2   = expression(C.O2.in),
                          C.ALG   = 0,
                          C.ZOO   = 0,
                          C.POMD  = 0,
                          C.POMI  = 0),
       outflow    = expression(Q.in*86400),
       cond       = cond.epi,
       processes  = list(gro.ALG.NH4,gro.ALG.NO3,resp.ALG,death.ALG,
                        gro.ZOO,resp.ZOO,death.ZOO,
                        nitri,miner.ox.POM))

# Hypolimnion:

hypolimnion <-
  new(Class      = "reactor",
       name      = "Hypo",
       volume.ini = expression(A*h.hypo),
       area       = expression(A),
       conc.pervol.ini = list(C.HPO4 = expression(C.HPO4.ini), # gP/m3
                              C.NH4  = expression(C.NH4.ini), # gN/m3
                              C.NO3  = expression(C.NO3.ini), # gN/m3
                              C.O2   = expression(C.O2.ini), # gO/m3
                              C.ALG   = expression(C.ALG.ini), # gDM/m3
                              C.ZOO   = expression(C.ZOO.ini), # gDM/m3
                              C.POMD  = expression(C.POMD.ini), # gDM/m3
                              C.POMI  = expression(C.POMI.ini)), # gDM/m3
       conc.perarea.ini = list(D.POMD = expression(D.POMD.ini), # gDM/m2
                              D.POMI  = expression(D.POMI.ini)), # gDM/m2
       cond         = cond.hypo,

```

```

processes      = list( resp.ALG, death.ALG,
                      gro.ZOO, resp.ZOO, death.ZOO,
                      nitri,
                      miner.ox.POM, miner.ox.POM.sed, miner.anox.POM.sed,
                      sed.POMD, sed.POMI))

```

The final element needed is the metalimnion that separates the epilimnion from the hypolimnion. We neglect its volume (this means we expand the epilimnion and the hypolimnion) and specify directed flow of dead organic particles through sedimentation from the epilimnion to the hypolimnion and diffusive exchange of all substances with the turbulent diffusion coefficient K_z that was defined above to be large in winter and small in summer.

```

# Definition of links:

# Exchange between epilimnion and hypolimnion:

metalimnion <-
  new(Class      = "link",
       name      = "Metalimnion",
       from      = "Epi",
       to        = "Hypo",
       qadv.spec = list(C.POMD = expression(v.sed.POM*A),
                       C.POMI = expression(v.sed.POM*A)),
       qdiff.gen = expression(A/h.meta*Kz))

```

The definition of the model is concluded with the system definition that contains the reactors, links, environmental conditions, parameters and output time points for simulations:

```

#Definition of the lake system:

system.11.4 <- new(Class   = "system",
                  name     = "Lake",
                  reactors = list(epilimnion,hypolimnion),
                  links    = list(metalimnion),
                  cond     = cond.gen,
                  param    = param,
                  t.out    = seq(0,730,by=1)) #number of days 365, 730, 1095 ,1460

```

2.3 Perform simulations and plot results Now, we start simulations. The default option is to use an implicit integration scheme that is stable also for large time steps; as an alternative, we could apply an explicit Runge-Kutta scheme that is faster for each time step but needs more of them:

```

# Perform simulation:

ptm <- proc.time()
res.11.4 <- calcres(system.11.4)
print(proc.time()-ptm)

ptm <- proc.time()
res.11.4.rk4 <- calcres(system.11.4,method="rk4")
print(proc.time()-ptm)

```

We then plot the results to the screen and write them also with different options to files. Please note that you have to adjust the size of the pdf files with the `width` and `height` arguments to have reasonable sizes of the legend and the descriptions of the axes. You may also want to plot multiple lines into a single plot by specifying the argument `colnames` of the function `plotres` (you may type `?plotres` to get a description of the arguments of this function).

```
# Plot results:

plotres(res      = res.11.4,
        colnames =list(c("C.NH4.Epi", "C.NH4.Hypo"),
                        c("C.NO3.Epi", "C.NO3.Hypo"),
                        c("C.HPO4.Epi", "C.HPO4.Hypo"),
                        c("C.O2.Epi", "C.O2.Hypo"),
                        c("C.ALG.Epi", "C.ALG.Hypo"),
                        c("C.ZOO.Epi", "C.ZOO.Hypo"),
                        c("C.POMD.Epi", "C.POMD.Hypo"),
                        c("C.POMI.Epi", "C.POMI.Hypo"),
                        c("D.POMD.Hypo", "D.POMI.Hypo")))

plotres(res      = res.11.4,
        colnames = list(c("C.NH4.Epi", "C.NH4.Hypo"),
                        c("C.NO3.Epi", "C.NO3.Hypo"),
                        c("C.HPO4.Epi", "C.HPO4.Hypo"),
                        c("C.O2.Epi", "C.O2.Hypo"),
                        c("C.ALG.Epi", "C.ALG.Hypo"),
                        c("C.ZOO.Epi", "C.ZOO.Hypo"),
                        c("C.POMD.Epi", "C.POMD.Hypo"),
                        c("C.POMI.Epi", "C.POMI.Hypo"),
                        c("D.POMD.Hypo", "D.POMI.Hypo")),
        file      = "exercise_4_results.pdf",
        width     = 8,
        height    = 8)

# compare numerical solutions:
plotres(res      = list(lsoda=res.11.4,rk4=res.11.4.rk4),
        colnames = list(c("C.NH4.Epi", "C.NH4.Hypo"),
                        c("C.NO3.Epi", "C.NO3.Hypo"),
                        c("C.HPO4.Epi", "C.HPO4.Hypo"),
                        c("C.O2.Epi", "C.O2.Hypo"),
                        c("C.ALG.Epi", "C.ALG.Hypo"),
                        c("C.ZOO.Epi", "C.ZOO.Hypo"),
                        c("C.POMD.Epi", "C.POMD.Hypo"),
                        c("C.POMI.Epi", "C.POMI.Hypo"),
                        c("D.POMD.Hypo", "D.POMI.Hypo")),
        file      = "exercise_4_results_rk4.pdf",
        width     = 8,
        height    = 8)
```

Overall mass balance of phosphorus over the whole simulation period:

```
# Phosphorus mass balance:

nr.days  <- (as.numeric(rownames(res.11.4)[nrow(res.11.4)])-
            as.numeric(rownames(res.11.4)[1]))
```

```

nr.steps <- (nrow(res.11.4)-1)

F.in.P <- c(HP04 = param$Q.in*param$C.HP04.in*nr.days*86400/1e6)
F.out.P <- c(HP04 = sum(param$Q.in*res.11.4[, "C.HP04.Epi"])*
  nr.days/nr.steps*86400/1e6,
  ALG = sum(param$Q.in*res.11.4[, "C.ALG.Epi"]*param$alpha.P.ALG)*
  nr.days/nr.steps*86400/1e6,
  ZOO = sum(param$Q.in*res.11.4[, "C.ZOO.Epi"]*param$alpha.P.ZOO)*
  nr.days/nr.steps*86400/1e6,
  POMD = sum(param$Q.in*res.11.4[, "C.POMD.Epi"]*param$alpha.P.POM)*
  nr.days/nr.steps*86400/1e6,
  POMI = sum(param$Q.in*res.11.4[, "C.POMI.Epi"]*param$alpha.P.POM)*
  nr.days/nr.steps*86400/1e6)
Acc.P <- c(HP04 = param$A/1e6*
  ((param$h.epi*res.11.4[nrow(res.11.4), "C.HP04.Epi"]+
  param$h.hypo*res.11.4[nrow(res.11.4), "C.HP04.Hypo"])-
  (param$h.epi*res.11.4[1, "C.HP04.Epi"]+
  param$h.hypo*res.11.4[1, "C.HP04.Hypo"])),
  ALG = param$A/1e6*param$alpha.P.ALG*
  ((param$h.epi*res.11.4[nrow(res.11.4), "C.ALG.Epi"]+
  param$h.hypo*res.11.4[nrow(res.11.4), "C.ALG.Hypo"])-
  (param$h.epi*res.11.4[1, "C.ALG.Epi"]+
  param$h.hypo*res.11.4[1, "C.ALG.Hypo"])),
  ZOO = param$A/1e6*param$alpha.P.ZOO*
  ((param$h.epi*res.11.4[nrow(res.11.4), "C.ZOO.Epi"]+
  param$h.hypo*res.11.4[nrow(res.11.4), "C.ZOO.Hypo"])-
  (param$h.epi*res.11.4[1, "C.ZOO.Epi"]+
  param$h.hypo*res.11.4[1, "C.ZOO.Hypo"])),
  POMD = param$A/1e6*param$alpha.P.POM*
  ((param$h.epi*res.11.4[nrow(res.11.4), "C.POMD.Epi"]+
  param$h.hypo*res.11.4[nrow(res.11.4), "C.POMD.Hypo"])-
  (param$h.epi*res.11.4[1, "C.POMD.Epi"]+
  param$h.hypo*res.11.4[1, "C.POMD.Hypo"])),
  POMI = param$A/1e6*param$alpha.P.POM*
  ((param$h.epi*res.11.4[nrow(res.11.4), "C.POMI.Epi"]+
  param$h.hypo*res.11.4[nrow(res.11.4), "C.POMI.Hypo"])-
  (param$h.epi*res.11.4[1, "C.POMI.Epi"]+
  param$h.hypo*res.11.4[1, "C.POMI.Hypo"])),
  POMDsed = param$A/1e6*param$alpha.P.POM*
  (res.11.4[nrow(res.11.4), "D.POMD.Hypo"]-
  res.11.4[1, "D.POMD.Hypo"]),
  POMIsed = param$A/1e6*param$alpha.P.POM*
  (res.11.4[nrow(res.11.4), "D.POMI.Hypo"]-
  res.11.4[1, "D.POMI.Hypo"]))

print("Input (tP)")
print(signif(F.in.P,3))
print("Output (tP)")
print(signif(F.out.P,3))
print(signif(sum(F.out.P),3))
print("Accumulation (tP)")
print(signif(Acc.P,3))
print(signif(sum(Acc.P),3))

```

```
print("Input - Output - Accumulation:")
print(sum(F.in.P)-sum(F.out.P)-sum(Acc.P))
```

Overall mass balance of nitrogen over the whole simulation period:

```
# Nitrogen mass balance:

nr.days <- (as.numeric(rownames(res.11.4)[nrow(res.11.4)])-
           as.numeric(rownames(res.11.4)[1]))
nr.steps <- (nrow(res.11.4)-1)

F.in.N <- c(NO3      = param$Q.in*param$C.NO3.in*nr.days*86400/1e6)
F.out.N <- c(NO3      = sum(param$Q.in*res.11.4[, "C.NO3.Epi"])*
           nr.days/nr.steps*86400/1e6,
           NH4       = sum(param$Q.in*res.11.4[, "C.NH4.Epi"])*
           nr.days/nr.steps*86400/1e6,
           ALG       = sum(param$Q.in*res.11.4[, "C.ALG.Epi"]*param$alpha.N.ALG)*
           nr.days/nr.steps*86400/1e6,
           ZOO       = sum(param$Q.in*res.11.4[, "C.ZOO.Epi"]*param$alpha.N.ZOO)*
           nr.days/nr.steps*86400/1e6,
           POMD      = sum(param$Q.in*res.11.4[, "C.POMD.Epi"]*param$alpha.N.POM)*
           nr.days/nr.steps*86400/1e6,
           POMI      = sum(param$Q.in*res.11.4[, "C.POMI.Epi"]*param$alpha.N.POM)*
           nr.days/nr.steps*86400/1e6)

Acc.N <- c(NO3      = param$A/1e6*
           ((param$h.epi*res.11.4[nrow(res.11.4), "C.NO3.Epi"]+
            param$h.hypo*res.11.4[nrow(res.11.4), "C.NO3.Hypo"])-
            (param$h.epi*res.11.4[1, "C.NO3.Epi"]+
             param$h.hypo*res.11.4[1, "C.NO3.Hypo"])),
           NH4       = param$A/1e6*
           ((param$h.epi*res.11.4[nrow(res.11.4), "C.NH4.Epi"]+
            param$h.hypo*res.11.4[nrow(res.11.4), "C.NH4.Hypo"])-
            (param$h.epi*res.11.4[1, "C.NH4.Epi"]+
             param$h.hypo*res.11.4[1, "C.NH4.Hypo"])),
           ALG       = param$A/1e6*param$alpha.N.ALG*
           ((param$h.epi*res.11.4[nrow(res.11.4), "C.ALG.Epi"]+
            param$h.hypo*res.11.4[nrow(res.11.4), "C.ALG.Hypo"])-
            (param$h.epi*res.11.4[1, "C.ALG.Epi"]+
             param$h.hypo*res.11.4[1, "C.ALG.Hypo"])),
           ZOO       = param$A/1e6*param$alpha.N.ZOO*
           ((param$h.epi*res.11.4[nrow(res.11.4), "C.ZOO.Epi"]+
            param$h.hypo*res.11.4[nrow(res.11.4), "C.ZOO.Hypo"])-
            (param$h.epi*res.11.4[1, "C.ZOO.Epi"]+
             param$h.hypo*res.11.4[1, "C.ZOO.Hypo"])),
           POMD      = param$A/1e6*param$alpha.N.POM*
           ((param$h.epi*res.11.4[nrow(res.11.4), "C.POMD.Epi"]+
            param$h.hypo*res.11.4[nrow(res.11.4), "C.POMD.Hypo"])-
            (param$h.epi*res.11.4[1, "C.POMD.Epi"]+
             param$h.hypo*res.11.4[1, "C.POMD.Hypo"])),
           POMI      = param$A/1e6*param$alpha.N.POM*
           ((param$h.epi*res.11.4[nrow(res.11.4), "C.POMI.Epi"]+
            param$h.hypo*res.11.4[nrow(res.11.4), "C.POMI.Hypo"])-
            (param$h.epi*res.11.4[1, "C.POMI.Epi"]+
             param$h.hypo*res.11.4[1, "C.POMI.Hypo"])))
```



```

        param$h.hypo*res.11.4[1,"C.POMI.Hypo"])),
POMDsed = param$A/1e6*param$alpha.N.POM*
        (res.11.4[nrow(res.11.4),"D.POMD.Hypo"]-
        res.11.4[1,"D.POMD.Hypo"]),
POMIsed = param$A/1e6*param$alpha.N.POM*
        (res.11.4[nrow(res.11.4),"D.POMI.Hypo"]-
        res.11.4[1,"D.POMI.Hypo"]))

print("Input (tN)")
print(signif(F.in.N,3))
print("Output (tN)")
print(signif(F.out.N,3))
print(signif(sum(F.out.N),3))
print("Accumulation (tN)")
print(signif(Acc.N,3))
print(signif(sum(Acc.N),3))

print("Input - Output - Accumulation:")
print(sum(F.in.N)-sum(F.out.N)-sum(Acc.N))

```

Task 3: Model results

Study the time courses of the state variables and the overall fluxes of phosphorus and nitrogen resulting from executing the chunks above and interpret the results.

Task 4: Homework: Model sensitivity

Perform a sensitivity analysis with respect to the most important kinetic parameters and interpret the results.

Theory questions

1. Why is it important that some stoichiometric coefficients are indicated as 0/+?
2. How is the metalimnion represented in the model?
3. Look at the process rates of mineralization in the sediment. Why are they different from those in chapter 8.5?
4. Look at the mass balance for P and N. If there is a difference between input and output + accumulation, where does it come from? *Hint*: Have a look at the stoichiometric coefficients for anoxic mineralization.