Exercise 1

Modelling Aquatic Ecosystems FS25

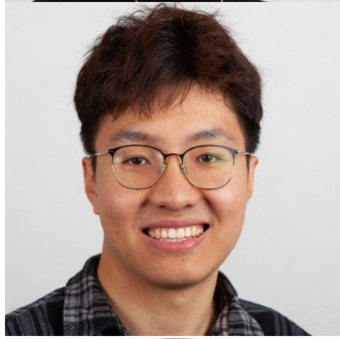
About exercises

- Check Program 2025 for course schedule
- Personal laptop is needed for all exercises
- Every exercise has some tasks, including homework. (Submission is not required)
- Programming skills is not assessed.
- Solution will be uploaded after every exercise class

Date	Topic	Type
19.02.25	Introduction and overview of the course (chapter 1 and 2); Mass balance in a mixed reactor (ch. 3.2); Process table (ch. 4.1); Process rates (ch. 4.2); Simple phytoplankton model for a mixed lake (ch. 11.1)	L
26.02.25	Introduction to R and the „ecosim“-package (chapter 17); Review of chapter 4.1-4.2 through the example of chapter 11.1.	E
05.03.25	Extension of the first lake model to a simple phytoplankton-zooplankton model for a mixed lake (chapter 11.2)	E
12.03.25	Process stoichiometry: introduction and analytical solution (chapter 4.3.1 and 4.3.2)	L
19.03.25	Process stoichiometry: general solution from chapter 4.3.3; Introduction to the „stoichcalc“-package (chapter 15), Application of the „stoichcalc“-package	L+E
26.03.25	Biological processes in lakes (ch. 8): mineralization, nitrification, secondary production; Extension of the lake model (sediment, phosphorous, oxygen, nitrogen) (chapter 11.3)	L
02.04.25	Physical processes in lakes; Mass balance in multi-box and continuous systems (chapter 3.3-3.4); Transport and mixing in lakes (ch. 6.1.1); Sedimentation (ch. 6.2); Gas exchange (ch. 6.3); model assignments - choice of topics	L
09.04.25	Spatially structured model for plankton and biogeochemical cycles in lakes (ch. 11.4).	E
16.04.25	Transport and mixing in rivers (ch. 6.1.2); Bacterial growth (ch. 8.8); Model for benthic populations, oxygen and nutrients in rivers (ch. 11.6)	L
30.04.25	Model for benthic populations, oxygen and nutrients in rivers (chapter 11.6)	E
08.05.25	Deadline code submission for assignments	
07.05.25	Uncertainty (chapter 9), Parameter estimation (chapter 10)	L
14.05.25	Parameter estimation (chapter 10)	E
21.05.25	Stochasticity and Uncertainty (chapter 9, chapter 11.7)	L+E
23.05.25	Deadline submission of assignments	
28.05.25	Overview of existing models and their application in research and practice (ch. 13); Preparation of the oral exam; Feedback	L
02. - 13.06.25	Oral exams (exact date to be decided)	

Get helped

Emma Chollet Ramampiandra
emma.chollet@eawag.ch



Chuxinyao Wang
Chuxinyao.wang@eawag.ch

Andreas Scheidegger
andreas.scheidegger@eawag.ch

Ask your questions in class or via emails!

Introduction to exercise 1

- Instructions is given in html and pdf.
- R programming
  - There are some useful links for R tutorials you can find in exercise instructions.
  - A little useful AI tool: Copilot encompasses in RStudio.
    - Set up guidance
    - Live demonstration
- You will manually copy the code to an R-file and run it in the console.

Exercise 1: Lake Phytoplankton Model

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

February 26, 2025

Goals

- Review basic elements of R needed to handle the technical aspects of the exercises.
- Be able to implement the simple lake phytoplankton model described in **section 11.1** of the manuscript.
- Understand the basic structure and functioning of the R package 'ecosim'.
- Understand the behaviour of the solutions of this model.

Notes

1. All the exercises files can be downloaded from the course homepage http://www.eawag.ch/forschung/siam/lehre/modaqecosys.
2. To conduct the exercises, install the newest version of R on your computer from http://rproject.org. We recommend to use RStudio as an editor for R: http://rstudio.com. All the R Markdown exercise files (ending with `.Rmd`) can then be opened and modified on RStudio. Finally, you can install the required packages `ecosim`, `stoichcalc` and `deSolve` by executing the following commands.

```
# load required packages:

# to conduct the exercises:
if ( !require("ecosim") ) {install.packages("ecosim"); library("ecosim") }
if ( !require("stoichcalc") ) {install.packages("stoichcalc"); library("stoichcalc") }
if ( !require("deSolve") ) {install.packages("deSolve"); library("deSolve") }

# to work with the R Markdown format:
if ( !require("markdown") ) {install.packages("markdown"); library("markdown") }
```

Note that these commands install the required packages only if they are not yet installed. However, only installing them explicitly with the function `install.packages("...")` guarantees that the newest version is installed (because required packages are not re-installed if the are already installed).

Task 1: Introduction to R

Become familiar with R. See presentation and separate documentation.

Other useful resources can be found at http://r-project.org, in particular:

- http://cran.r-project.org/manuals.html
- https://cran.r-project.org/other-docs.html
- https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf

If you are new to R and want to quickly get used to the basics, a short (~30 minutes) **optional** tutorial can be found at:

- https://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf

Introduction to exercise 1

- A comparison between

  a) Direct implementation of the differential equation and

  b) Using the "ecosim" package to implement and solve the differential equations.

- Write differential equations in a function with **time period, state variables** and **defined parameters**.

```
# definition of right-hand side of differential equations (11.8, 11.9):

rhs <- function(t,y,par)
{

  # equation (11.8):

  dC.HPO4_dt <-   par$Q.in*86400/(par$h.epi*par$A) * (par$C.HPO4.in - y["C.HPO4"]) -
                  par$alpha.P.ALG * par$k.gro.ALG * y["C.HPO4"] /
    (par$K.HPO4 + y["C.HPO4"]) * y["C.ALG"]

  # equation (11.9): TO BE COMPLETED

  dC.ALG_dt  <-  - par$Q.in*86400/(par$h.epi*...) * y["C.ALG"] +
                   par$k.gro.ALG * y["C.HPO4"] / (par$K.HPO4 + y["C.HPO4"]) * ... -
                   ... * y["C.ALG"]

  return(list(c(dC.HPO4_dt,dC.ALG_dt)))
}
```

Introduction to exercise 1

- A comparison between
- a) Direct implementation of the differential equation and
- b) Using the "ecosim" package to implement and solve the differential equations.

- Use ‘ODE’ solver to compute the equations

```
# solve differential equations:

res <- ode(y=c(C.HPO4=param$C.HPO4.ini,C.ALG=param$C.ALG.ini),
           times=seq(0,365,by=1),func=rhs,par=param)
```

Introduction to exercise 1

- A comparison between
- a) Direct implementation of the differential equation and
- **b)** Using the ‘ecosim’ package to implement and solve the differential equations.

- ‘ecosim’ is based on "object oriented programming". It uses **different classes** to define processes, reactors, links and the whole system.

Introduction to exercise 1

- Class: **process**
- Used to define a transformation process

Elements of class "process"

Name	Type	Meaning
name	string	Name of process.
rate	expression	Expression for the dependence of the process rate on substance concentrations, model parameters, and external influence factors.
stoich	list	List of numbers or expressions for stoichiometric coefficients. Substances are identified by their names.
pervol	logical	Type of process rate: mass per volume and time (TRUE) or per area and time (FALSE).

```
# definition of transformation processes

# growth of algae:

gro.ALG   <- new(Class  = "process",
                 name   = "Growth of algae",
                 rate   = expression(k.gro.ALG
                                     *C.HPO4/(K.HPO4+C.HPO4)
                                     *C.ALG),
                 stoich = list(C.ALG  = expression(1),              # gDM/gDM
                               C.HPO4 = expression(-alpha.P.ALG)))  # gP/gDM

# death of algae:

death.ALG <- new(Class = "process",
                 name   = "Death of algae",
                 rate   = expression(k.death.ALG*C.ALG),
                 stoich = list(C.ALG  = expression(-1)))            # gDM/gDM
```

Introduction to exercise 1

Process	Substances / Organisms HPO_4^{2-} gP	ALG gDM	Rate
Growth of algae	$-\alpha_{\mathrm{P,ALG}}$	1	$\rho_{\mathrm{gro,ALG}}$
Death of algae		-1	$\rho_{\mathrm{death,ALG}}$

Rate	Rate expression
$\rho_{\mathrm{gro,ALG}}$	$k_{\mathrm{gro,ALG}} \dfrac{C_{\mathrm{HPO_4^{2-}}}}{K_{\mathrm{HPO_4^{2-},ALG}} + C_{\mathrm{HPO_4^{2-}}}} C_{\mathrm{ALG}}$
$\rho_{\mathrm{death,ALG}}$	$k_{\mathrm{death,ALG}}\, C_{\mathrm{ALG}}$

```
# definition of transformation processes

# growth of algae:

gro.ALG   <- new(Class  = "process",
                 name   = "Growth of algae",
                 rate   = expression(k.gro.ALG
                                     *C.HPO4/(K.HPO4+C.HPO4)
                                     *C.ALG),
                 stoich = list(C.ALG  = expression(1),               # gDM/gDM
                               C.HPO4 = expression(-alpha.P.ALG)))   # gP/gDM

# death of algae:

death.ALG <- new(Class = "process",
                 name   = "Death of algae",
                 rate   = expression(k.death.ALG*C.ALG),
                 stoich = list(C.ALG  = expression(-1)))              # gDM/gDM
```

Introduction to exercise 1

- Class: **reactor**
- Used to define a mixed reactor

Name	Type	Meaning
name	string	Name of reactor.
volume.ini	expression	Initial volume of reactor.
area	expression	Surface area available for sessile organisms or attached (sedimented, adsorbed, etc.) substances.
conc.pervol.ini	list	Initial concentrations (mass per volume) of substances or organisms suspended or dissolved in the water column. Each substance to be calculated in the reactor must be initialized here.
conc.perarea.ini	list	Initial concentrations (mass per area) of substances or organisms attached to a surface. Each substance to be calculated in the reactor must be initialized here.
input	list	Input (mass per time) of substances to the reactor not associated with inflow.
inflow	expression	Inflow into the reactor (volume per time)
inflow.conc	list	Concentration of substances in the inflow.
outflow	expression	Outflow of the reactor (volume per time)
cond	list	Environmental conditions to which the reactor is exposed.
processes	list	Processes active in the reactor.

```
# definition of reactor to describe the epilimnion of the lake:

epilimnion <-
   new(Class               = "reactor",
    name                   = "Epilimnion",
    volume.ini             = expression(A*h.epi),
    conc.pervol.ini        = list(C.HPO4 = expression(C.HPO4.ini),     # gP/m3
                                  C.ALG  = expression(C.ALG.ini)),     # gDM/m3
    inflow                 = expression(Q.in*86400),                   # m3/d
    inflow.conc            = list(C.HPO4 = expression(C.HPO4.in),
                                  C.ALG  = 0),
    outflow                = expression(Q.in*86400),
    processes              = list(gro.ALG,death.ALG))# gDM/gDM
```

Introduction to exercise 1

$$\frac{d\mathbf{C}}{dt} = \frac{Q_{\text{in}}}{V}\left(\mathbf{C}_{\text{in}} - \mathbf{C}\right) + \frac{\mathbf{J}_{\text{int}}}{V} + \mathbf{r}$$

$$\mathbf{C}_{\text{in}} = \begin{pmatrix} C_{\text{in},\text{HPO}_4^{2-}} \\ 0 \end{pmatrix} \qquad \mathbf{C} = \begin{pmatrix} C_{\text{HPO}_4^{2-}} \\ C_{\text{ALG}} \end{pmatrix}$$

```
# definition of reactor to describe the epilimnion of the lake:

epilimnion <-
   new(Class              = "reactor",
       name               = "Epilimnion",
       volume.ini         = expression(A*h.epi),
       conc.pervol.ini    = list(C.HPO4 = expression(C.HPO4.ini),      # gP/m3
                                 C.ALG  = expression(C.ALG.ini)),      # gDM/m3
       inflow             = expression(Q.in*86400),                    # m3/d
       inflow.conc        = list(C.HPO4 = expression(C.HPO4.in),
                                 C.ALG  = 0),
       outflow            = expression(Q.in*86400),
       processes          = list(gro.ALG,death.ALG))# gDM/gDM
```

Introduction to exercise 1

- Class: **system**
- Used to define the model representing the system to be analyzed

Name	Type	Meaning
`name`	string	Name of system.
`reactors`	list	List of the reactors in the system.
`links`	list	List of advective links between reactors of the system.
`cond`	list	List of global environmental conditions to which all reactors are exposed.
`param`	list	List of model parameters in the form of numerical values or lists of vectors for x and y values describing a realization of a time-dependent parameter.
`t.out`	vector	Vector of points in time at which output should be calculated when dynamically solving the differential equations.

```
# definition of the system consisting of a single reactor:

system.11.1.a <- new(Class    = "system",
                     name     = "Lake",
                     reactors = list(epilimnion),
                     param    = param,
                     t.out    = seq(0,365,by=1))

# perform simulation:

res.11.1.a <- calcres(system.11.1.a)
```

Calculate dynamic solutions for the system

Introduction to exercise 1

$$\frac{dC_{HPO_4^{2-}}}{dt} = \boxed{\frac{Q_{in}}{V}\left(C_{in,HPO_4^{2-}} - C_{HPO_4^{2-}}\right)} \boxed{-\alpha_{P,ALG}} \cdot \boxed{k_{gro,ALG}\frac{C_{HPO_4^{2-}}}{K_{HPO_4^{2-},ALG} + C_{HPO_4^{2-}}}C_{ALG}}$$

$$\frac{dC_{ALG}}{dt} = \boxed{-\frac{Q_{in}}{V}C_{ALG}} \underbrace{\boxed{+}}_{1} \boxed{k_{gro,ALG}\frac{C_{HPO_4^{2-}}}{K_{HPO_4^{2-},ALG} + C_{HPO_4^{2-}}}C_{ALG} - k_{death,ALG}C_{ALG}}$$

Reactor properties

Stoichiometric coefficients

Process rates (growth & death of algae)

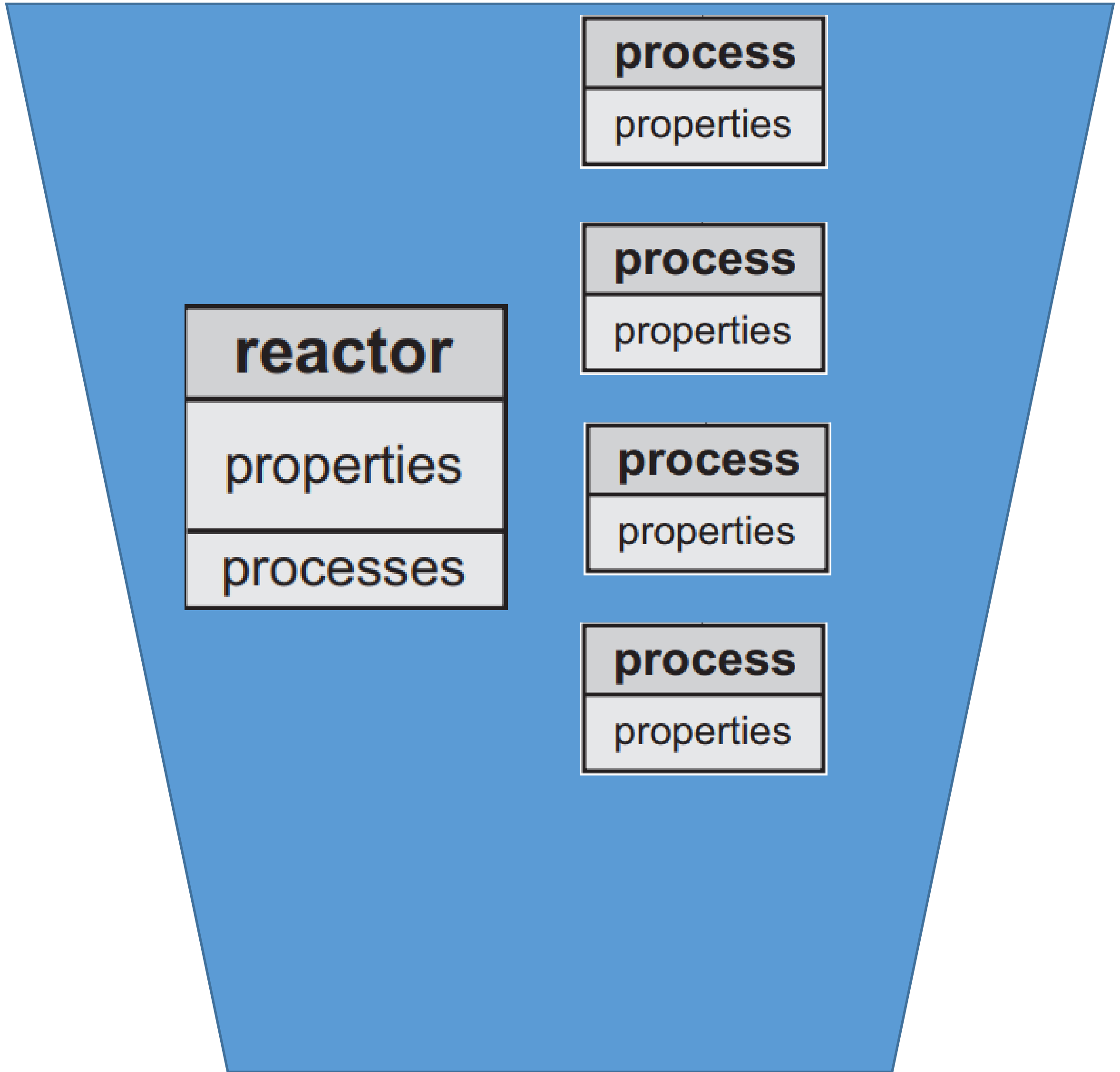
```
# definition of the system consisting of a single reactor:

system.11.1.a <- new(Class     = "system",
                     name      = "Lake",
                     reactors  = list(epilimnion),
                     param     = param,
                     t.out     = seq(0,365,by=1))

# perform simulation:

res.11.1.a <- calcres(system.11.1.a)
```

Introduction to exercise 1



All these classes comprise to the system we want to model.

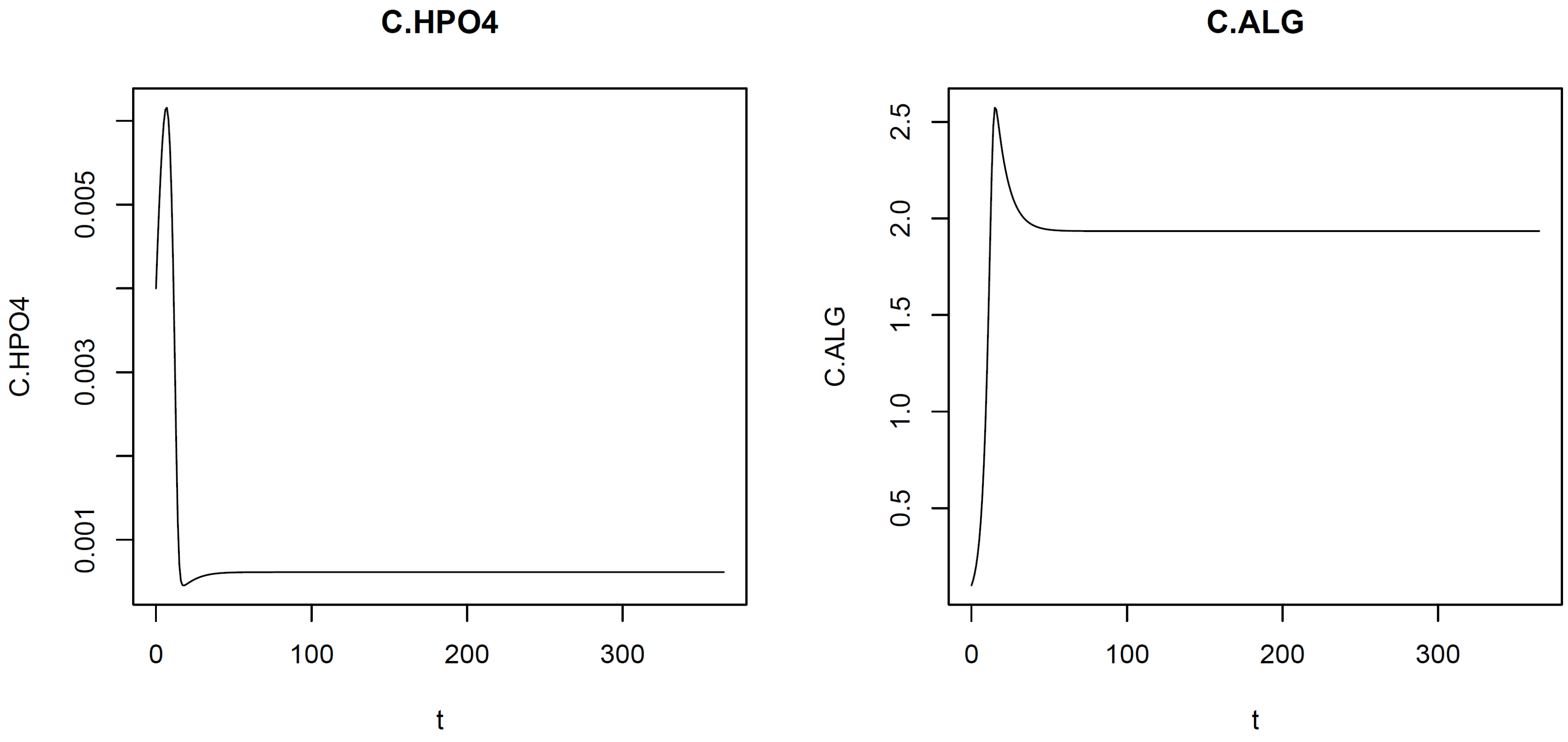
Now, it’s your turn to start the exercise!

We get back 15 minutes before 12pm for the answers.

Feel free to ask questions.

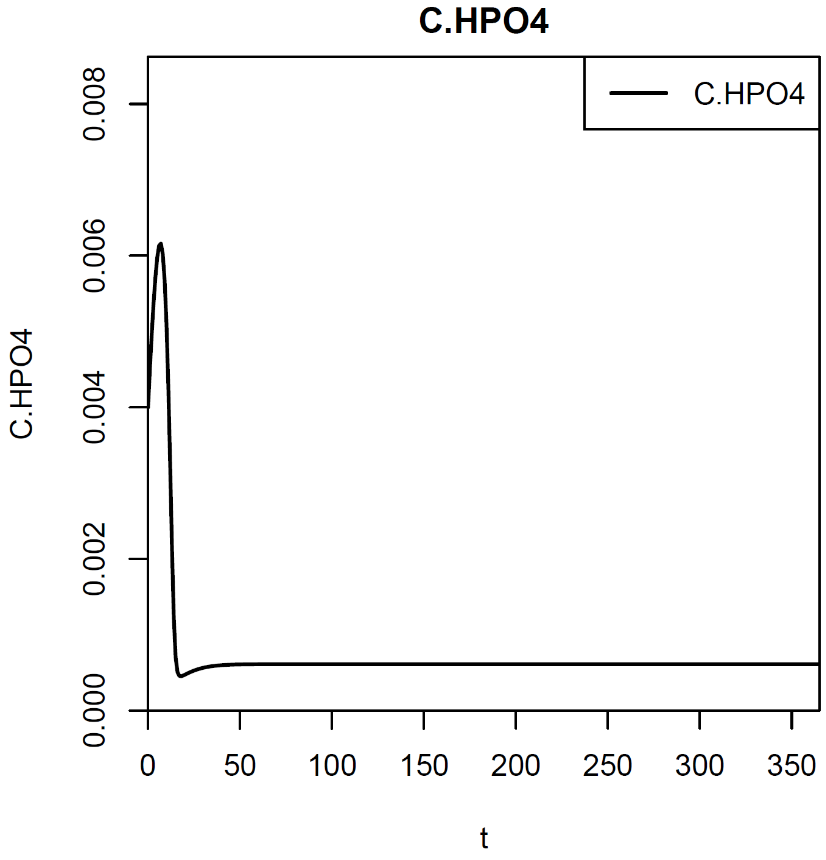
Task 2

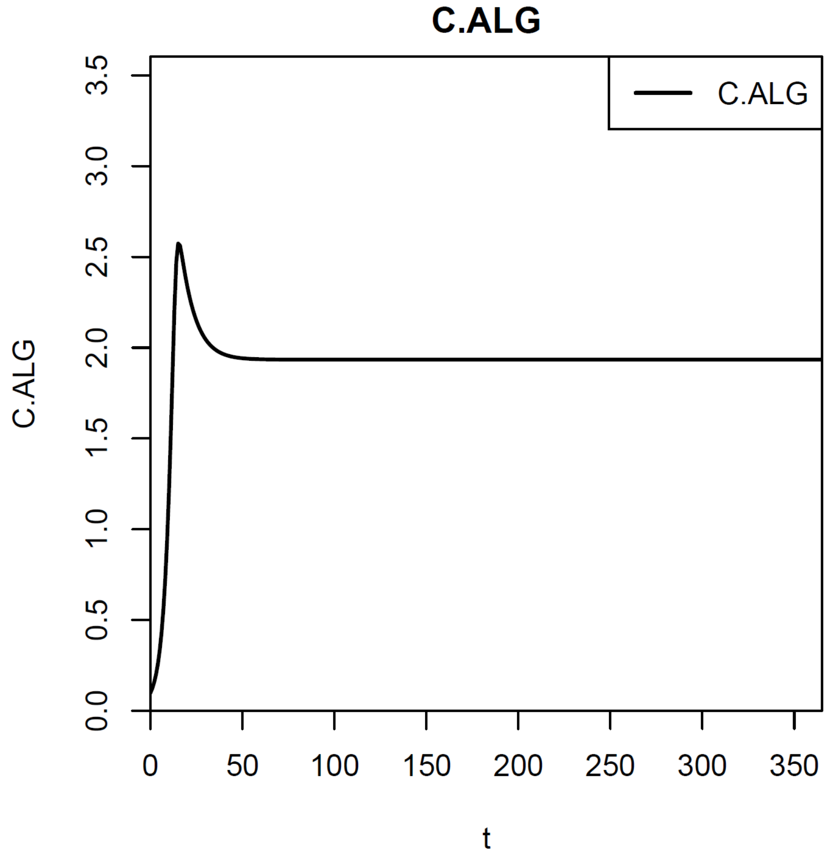
ODE result (1 year)



Task 3

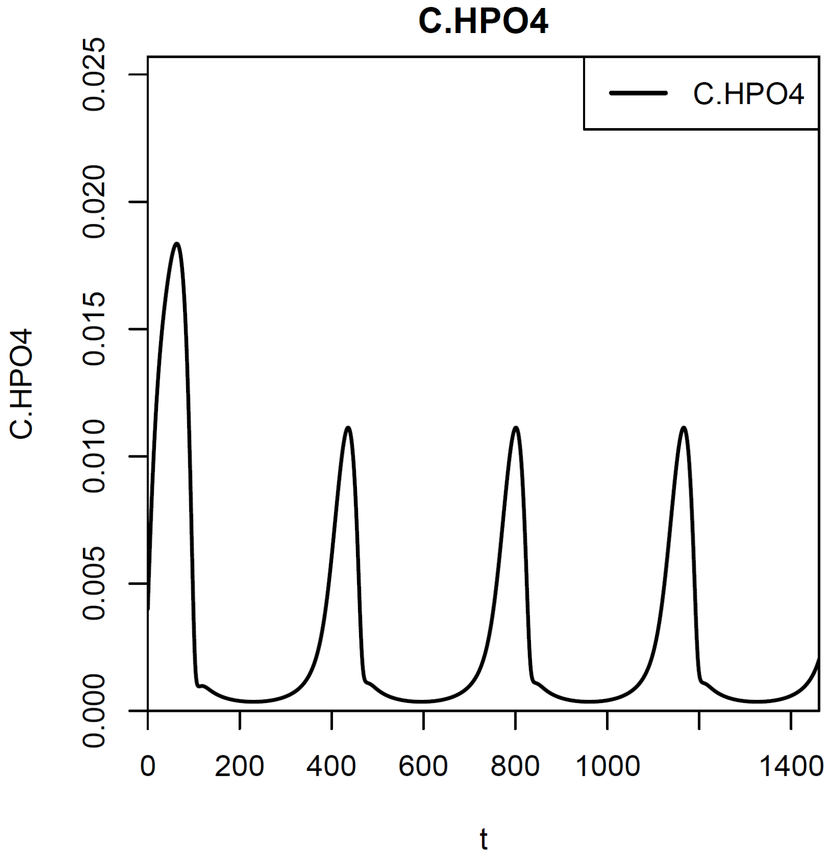
ecosim result (1 year)

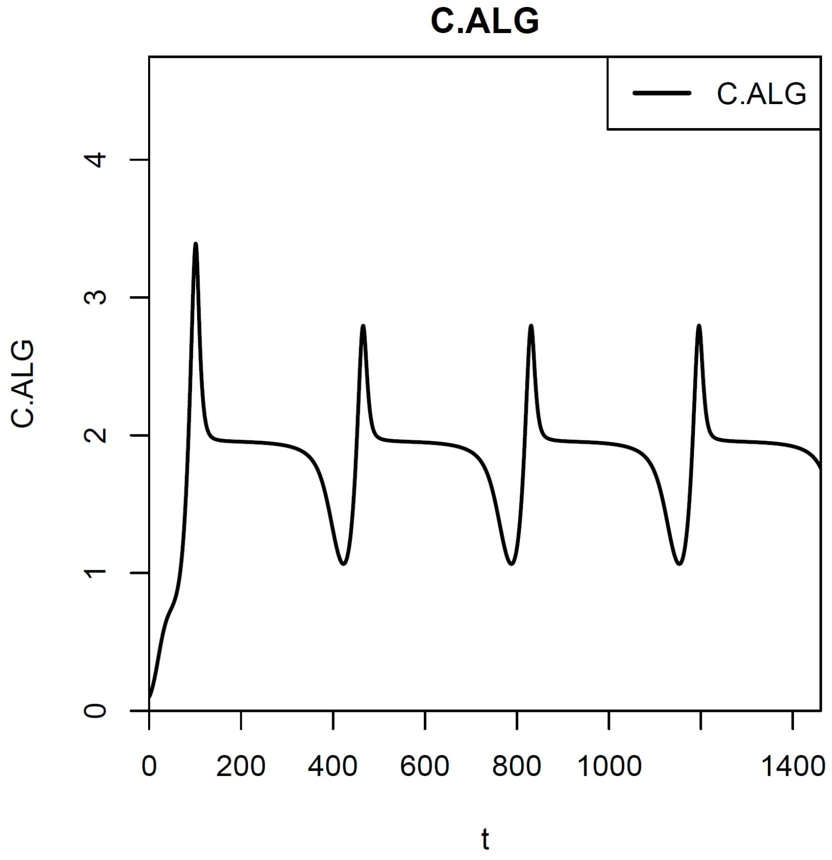




Task 4

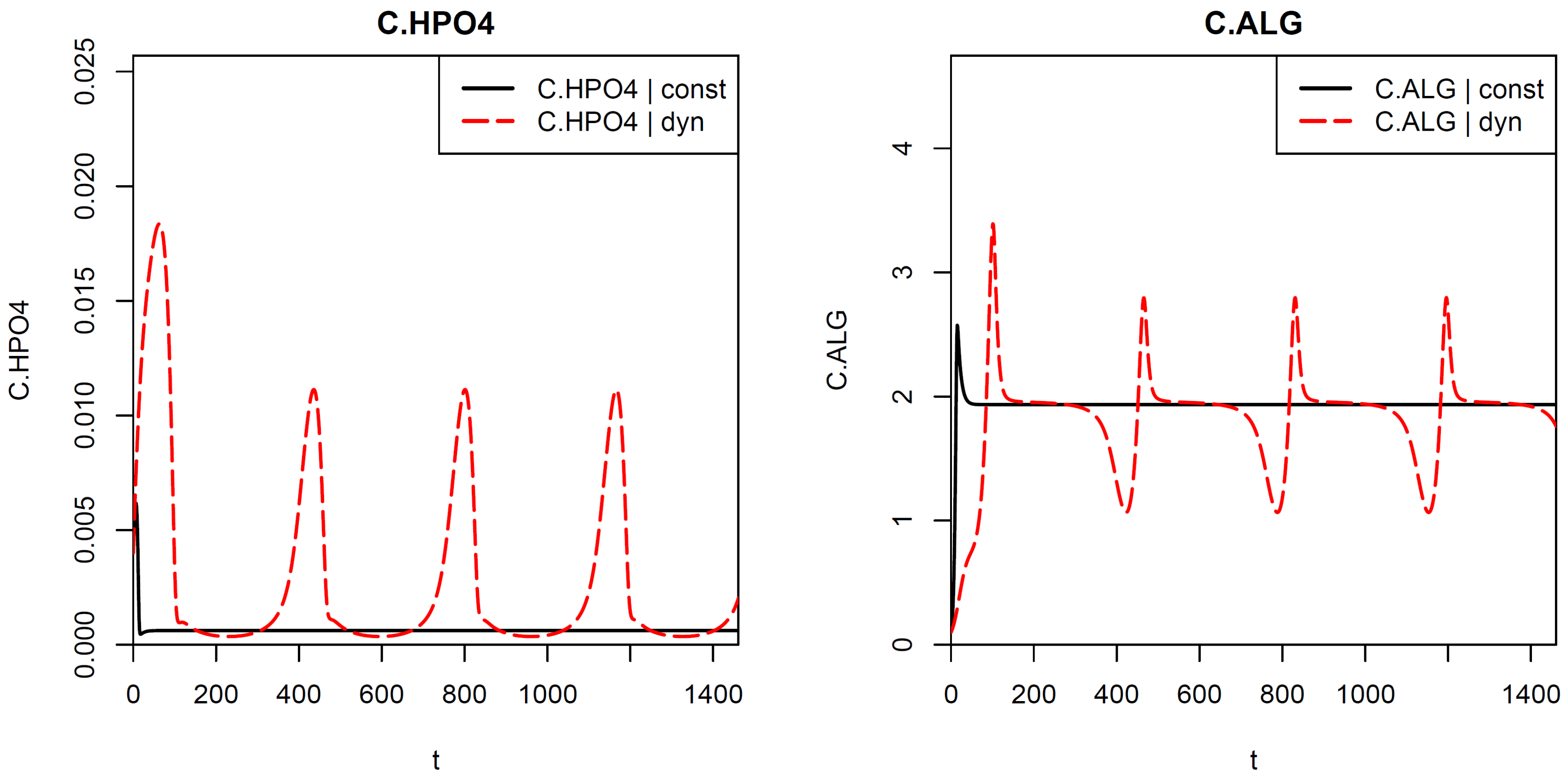
Periodic conditions
(4 years)





Task 4

Constant and periodic conditions
(4 years)



Homework: Task 5

- Using the function ‘`calcsens()`’ to perform sensitivity analysis.

- Sensitivity analysis supports our understanding of the importance of parameters on model results.

- Useful information: *Table 16.4* in Manuscript

Thanks for participating, see you next week!

Have a great day!