Exercise 1: Lake Phytoplankton Model

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

February 28, 2024

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.

```r
# 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

Task 2: Simple lake phytoplankton model with constant driving forces - Direct implementation using an ODE solver package

Carefully study the implementation of the model described in **section 11.1** given below for the case of constant driving forces.

2.1 Define the system and its parameters We investigate the system by solving the corresponding differential equations (11.8 and 11.9 in the manuscript) with the package `deSolve`. **Fill in the missing terms in the second equation (11.9).** *Hint:* Follow the structure of equation 11.8 as shown below and in the manuscript.

Pay attention to the different objects you add to your environment while you complete and run the following chunks.

We define a named list of model parameters:

```
# Model with constant driving forces
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# definition of model parameters:

param    <- list(k.gro.ALG   = 0.5,      # 1/d
                 k.death.ALG = 0.1,      # 1/d
                 K.HPO4      = 0.002,    # gP/m3
                 alpha.P.ALG = 0.003,    # gP/gDM
                 A           = 5e+006,   # m2
                 h.epi       = 5,        # m
                 Q.in        = 5,        # m3/s
                 C.HPO4.in   = 0.04,     # gP/m3
                 C.HPO4.ini  = 0.004,    # gP/m3
                 C.ALG.ini   = 0.1)      # gDM/m3
```

In a second step, we define a function `rhs` that will be passed to the solver to integrate the differential equations of our simple phytoplankton model. The solver passes the following arguments to the function to be potentially used to formulate the right-hand side of the differential equations:

- an argument for the time, here `t`, only used if there is explicit time-dependence in the right-hand side of the differential equations,
- an argument for the state variables, in our example `y=c("C.HPO4","C.ALG")`,
- an argument for the parameters, passed from the solver call to this function.

```
# 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*par$A) * y["C.ALG"] +
                  par$k.gro.ALG * y["C.HPO4"] / (par$K.HPO4 + y["C.HPO4"]) * y["C.ALG"] -
                  par$k.death.ALG * y["C.ALG"]

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

Note that the function is returning a list containing the vector of the derivatives of the state variables with respect to time.

Note also that the inflow `par$Q.in` is given as m^3/s and we would like to run simulations with uding days as the time unit, that is why we convert it by multiplying by $60 \cdot 60 \cdot 24 = 86400$.

2.2 Perform simulations We now study how to do simulations once a model has been defined and how to plot the results with the function `plot`.

First we solve the differential equations for 1 year with the solver `ode` of the package `deSolve`.

```
# read help file of the solver to understand the arguments needed:

?ode
```

```
## starting httpd help server ... done
```

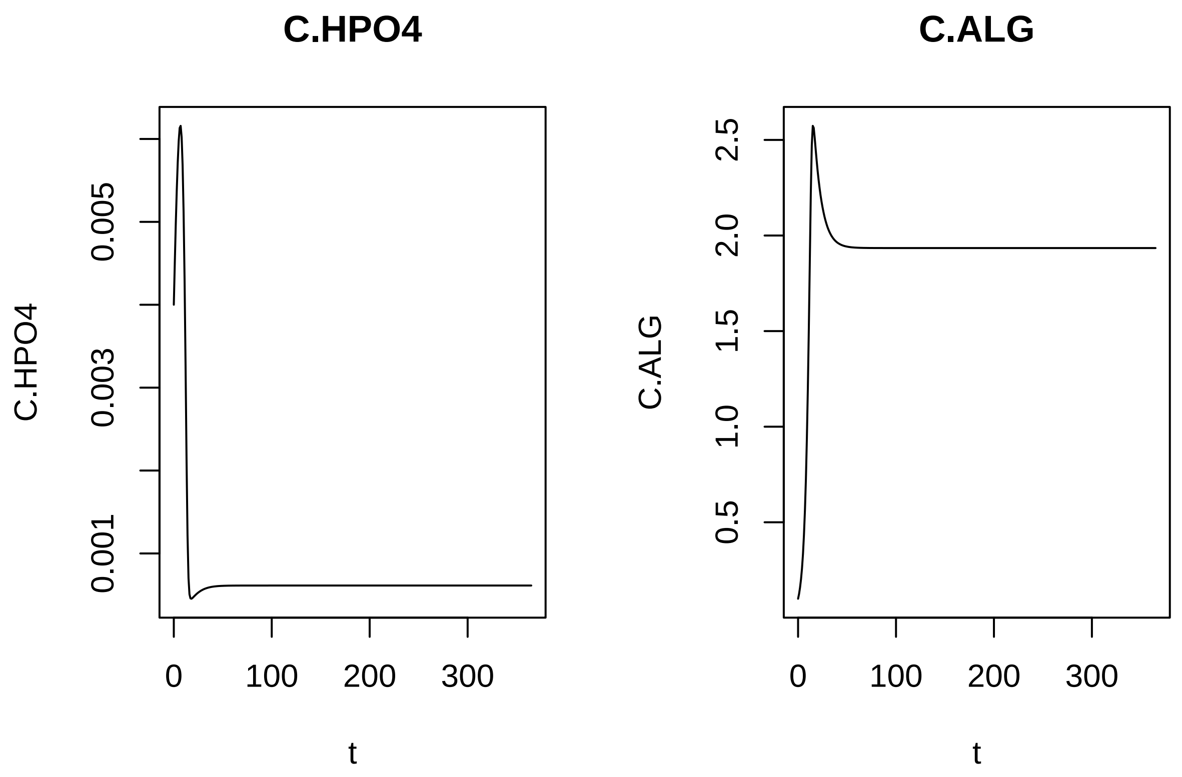
```
# solve differential equations:

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)
```

After reading the documentation of the ODE solver function `ode`, we understand how to use it to solve the equations of our model:

- as initial conditions, we give the initial concentrations of HPO4 and ALG defined in our parameter list at the beginning, be careful not to confuse it with the concentration in the inflow,
- then, we would like to run a simulation for one year and we would like to get daily outputs, so we pass a time sequence of 365 steps (`seq(0,365,by=1)`),
- finally, we fill in the function that defines the right-hand side of the differential equations we want to integrate and its parameters.

Then, we plot the results.

```
# plot results:

par(mfrow=c(1,2))
plot(res[,"time"],res[,"C.HPO4"],type="l",xlab="t",ylab="C.HPO4",main="C.HPO4")
plot(res[,"time"],res[,"C.ALG"] ,type="l",xlab="t",ylab="C.ALG" ,main="C.ALG")
```



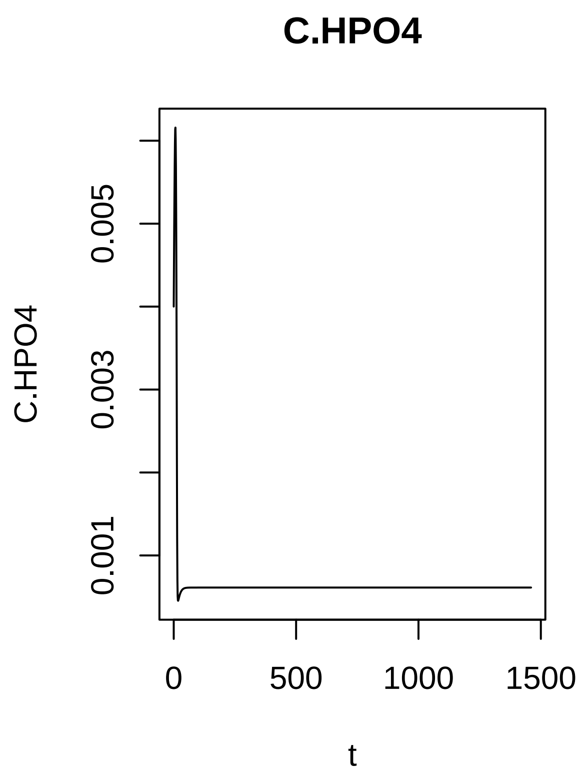
```
# plot results to a file:

file.name <- "exercise_1_results_a1_deSolve.pdf"
# open a pdf file to store the plots
pdf(file.name, paper = 'special', width = 10, height = 5, onefile = TRUE)

par(mfrow=c(1,2))
plot(res[,"time"],res[,"C.HPO4"],type="l",xlab="t",ylab="C.HPO4",main="C.HPO4")
plot(res[,"time"],res[,"C.ALG"] ,type="l",xlab="t",ylab="C.ALG" ,main="C.ALG")

dev.off() # close the pdf file
```

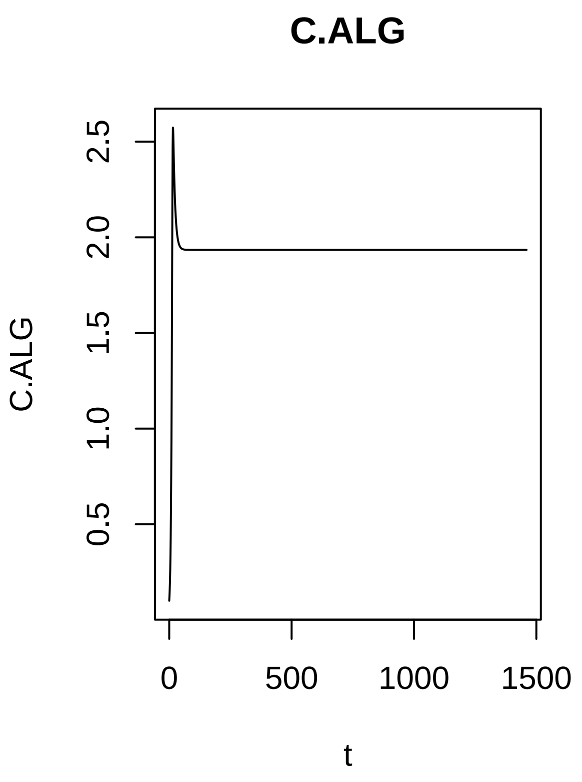
```
## pdf
##   2
```

Finally, **fill in the missing terms**, calculate and plot the solution for 4 years.

```
# change simulation time to 4 years:

res.4y <- ode(y=c(C.HPO4=param$C.HPO4.ini,C.ALG=param$C.ALG.ini),
              times=seq(0,4*365,by=1),func=rhs,par=param)

# plot results:

par(mfrow=c(1,2))
```

```
plot(res.4y[,"time"],res.4y[,"C.HPO4"],type="l",xlab="t",ylab="C.HPO4",main="C.HPO4")
plot(res.4y[,"time"],res.4y[,"C.ALG"] ,type="l",xlab="t",ylab="C.ALG" ,main="C.ALG")
```





Task 3: Implementation of a simple lake phytoplankton model with constant driving forces - `ecosim`

Now, we investigate the same problem using the `ecosim` package.

3.0 Introduction to the package ecosim See presentation, documentation in section 16 of the manuscript and the manual ecosim.pdf at http://cran.rproject.org/package=ecosim.

3.1 Define the system We don't need to define the parameters again, since it was done in Task 2.1. Become familiar with the creation of objects of the classes `process`, `reactor`, and `system` and link their entries to the system description provided in **section 11.1**. In particular, check the implementation of the object of the class `process` with the process table notation introduced in the course.

We define the processes of growth and death of algae as objects of the class `process` of the package `ecosim`. Each process is defined by its name, rate, and stoichiometry. The rate is defined as an expression that can use parameters (defined for the object of class `system` below), concentrations defined in objects of class `reactor` that are part of the object of class `system`. To define the stoichiometry a named list of expressions must be provided that identifies the substance or organism concentrations as the names and contains the stoichiometric coefficients as expressions.

```
# 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
```

Next, we define the mixed box describing the epliminion of the lake as an object of the class `reactor` of the package `ecosim`.

```
# 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))
```

Finally, we combine the reactor, the parameters, and the desired output times in an object of class `system` of the package `ecoval`.

```
# 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))
```

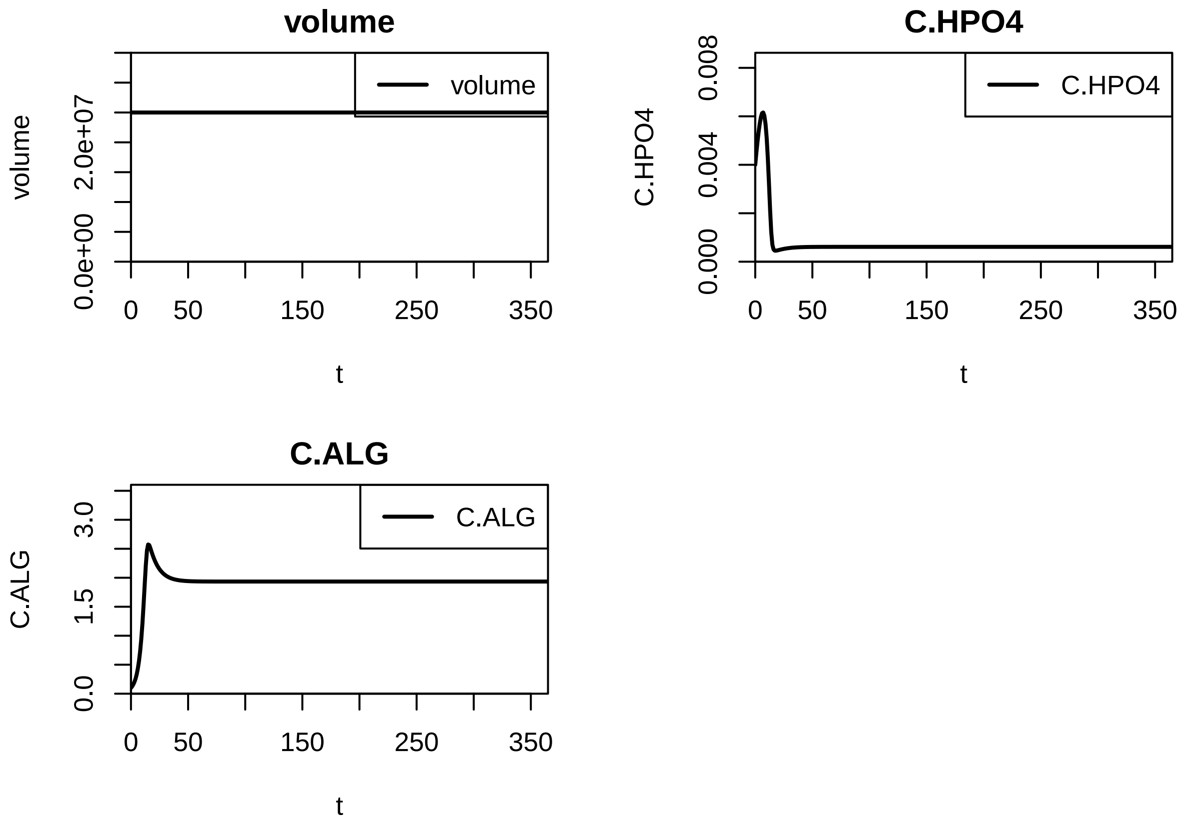
Note that this object contains all definitions of the configuration of reactors (in this case just a single one), the processes active in each reactor, the model parameters, and the output time points. Any simulations carried out will refer to the definitions in this object, and not to the external variables that we used to set up the elements of the system.

3.2 Perform simulations In a second step, we run simulations of the model as an object of the class `system` from the package `ecosim`. Perform simulations with the model and **try different plotting options** based on the code below and on your own ideas. Interpret the results.

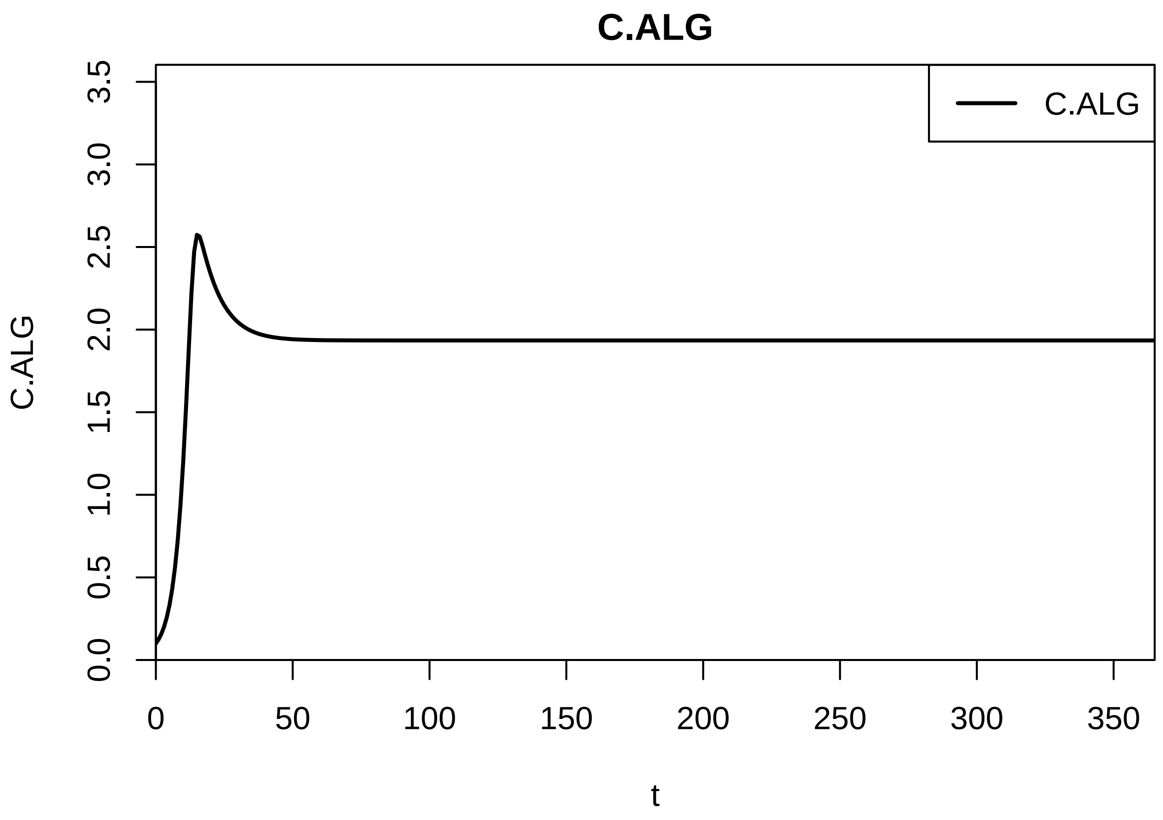
```r
# perform simulation:

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

Be aware that the function `calcres` of the `ecosim` package also uses `ode` to calculate the solution of the differential equations.

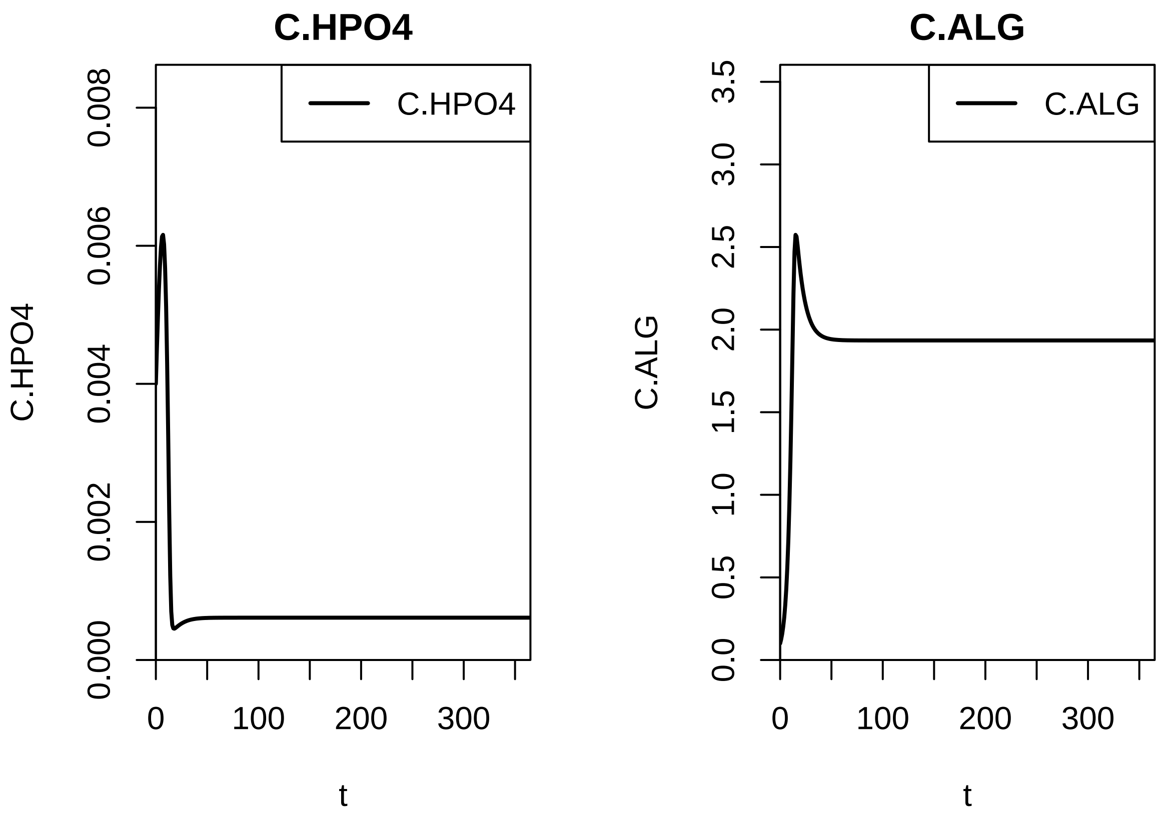
```r
# plot results whit default options:

plotres(res.11.1.a)
```



```r
# plot only results for the concentration of algae:

plotres(res=res.11.1.a,colnames="C.ALG")
```



```
# plot results for phosphate and algae:

plotres(res=res.11.1.a,colnames=list("C.HPO4","C.ALG"))
```

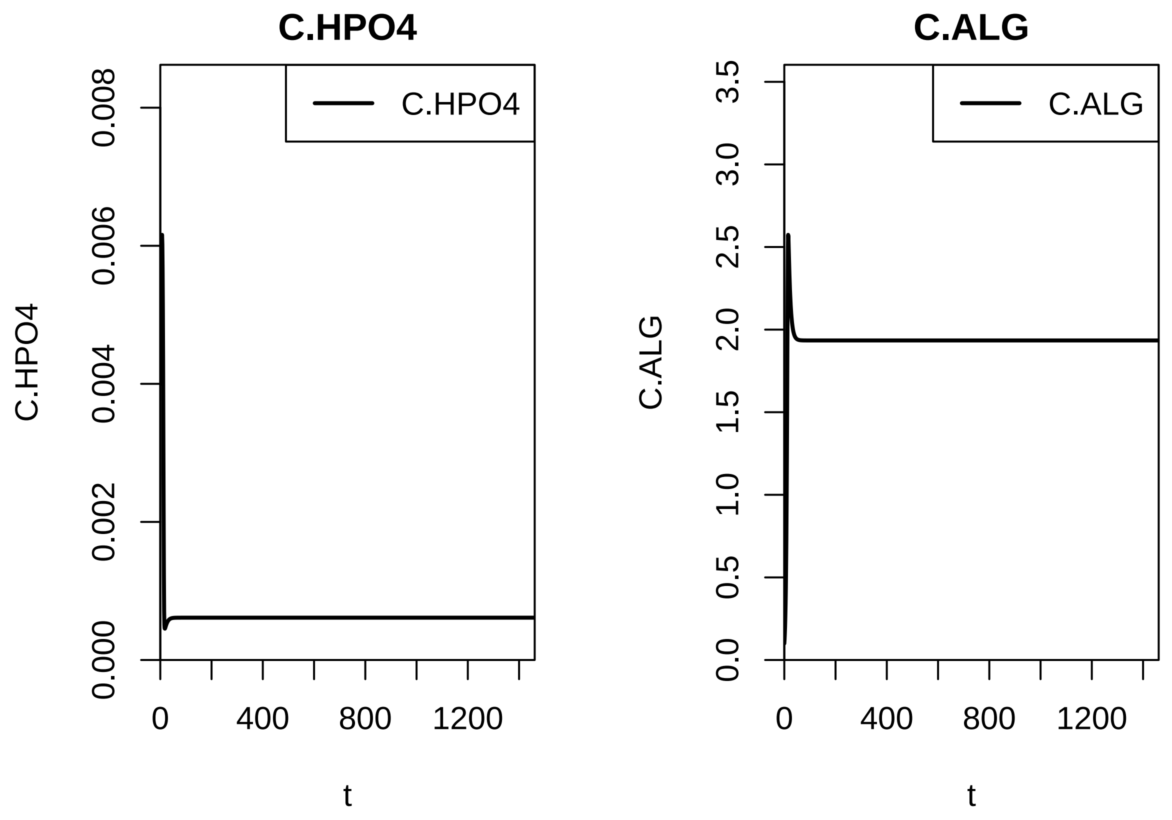


```
# plot results to a file:

plotres(res       = res.11.1.a,
        colnames = list("C.HPO4","C.ALG"),
        file      = "exercise_1_results_a1.pdf",
        width     = 10,
        height    = 5)
```

```
## pdf
##   2
```

Simulate for 4 years and plot results:

```
# change simulation time to 4 years:

system.11.1.a@t.out <- seq(0,4*365.25,by=1)

# calculate results for the system with modified simulation time:

res.11.1.a.4y <- calcres(system.11.1.a)

# plot results

plotres(res=res.11.1.a.4y,colnames=list("C.HPO4","C.ALG"))
```



```r
# plot results tp pdf

plotres(res      = res.11.1.a.4y,
        colnames = list("C.HPO4","C.ALG"),
        file     = "exercise_1_results_a2.pdf",
        width    = 10,
        height   = 5)
```

```
## pdf
##   2
```

3.3 Comparison of model implementation Now that you investigated the same system in two different ways (by defining the right hand side directly in R and solving it with `deSolve` and by using the package `ecosim`), what do you think are the advantages and disadvantages of each implementation? How might this change when the model becomes more complex?

Task 4: Extend the model to periodic driving forces and do simulations

From now on we will only work with the package `ecosim`.

Study and run the extension of the model to periodic driving forces as documented below. **Fill in the missing terms,** perform simulations, and plot and interpret the results.

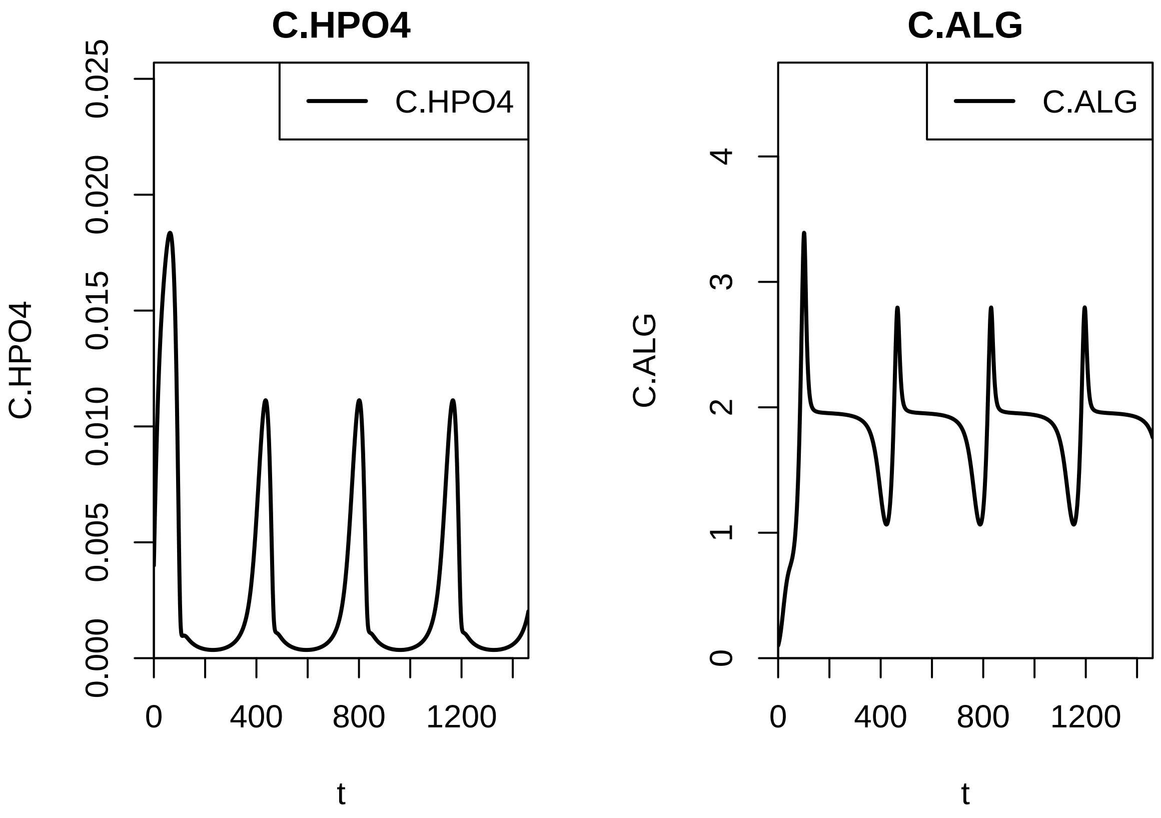
```
# Model with seasonally varying driving forces
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# extend system definitions:

system.11.1.b <- system.11.1.a

# extend growth of algae by environmental factors:

gro.ALG.ext <-
   new(Class  = "process",
       name   = "Growth of algae extended",
       rate   = expression(k.gro.ALG
                           *exp(beta.ALG*(T-T0))
                           *C.HPO4/(K.HPO4+C.HPO4)
                           *log((K.I+I0)
                                /(K.I+I0*exp(-(lambda.1+lambda.2*C.ALG)*h.epi)))
                            /((lambda.1+lambda.2*C.ALG)*h.epi)
                           *C.ALG),
       stoich = list(C.ALG  = 1,                              # gDM/gDM
                     C.HPO4 = expression(-alpha.P.ALG)))     # gP/gDM

# re-define processes in the reactor "epilimnion":

epilimnion@processes <- list(gro.ALG.ext,death.ALG)

# make environmental conditions (light and temperature) time dependent:

epilimnion@cond <- list(I0 = expression(0.5*(I0.min+I0.max)+
                                        0.5*(I0.max-I0.min)*
                                        cos(2*pi/365.25*(t-t.max))),   # W/m2
                        T  = expression(0.5*(T.min+T.max)+
                                        0.5*(T.max-T.min)*
                                        cos(2*pi/365.25*(t-t.max))))  # degC
```

```
# re-define the reactor "epilimnion" in the system definition:

system.11.1.b@reactors <- list(epilimnion)

# extend model parameters:

param  <- c(param,
            list(beta.ALG    = 0.046,    # 1/degC
                 T0          = 20,       # degC
                 K.I         = 30,       # W/m2
                 lambda.1    = 0.10,     # 1/m
                 lambda.2    = 0.10,     # m2/gDM
                 t.max       = 230,      # d
                 I0.min      = 25,       # W/m2
                 I0.max      = 225,      # W/m2
                 T.min       = 5,        # degC
                 T.max       = 25))      # degC

# increase algal growth rate to compensate for new limitations:
```

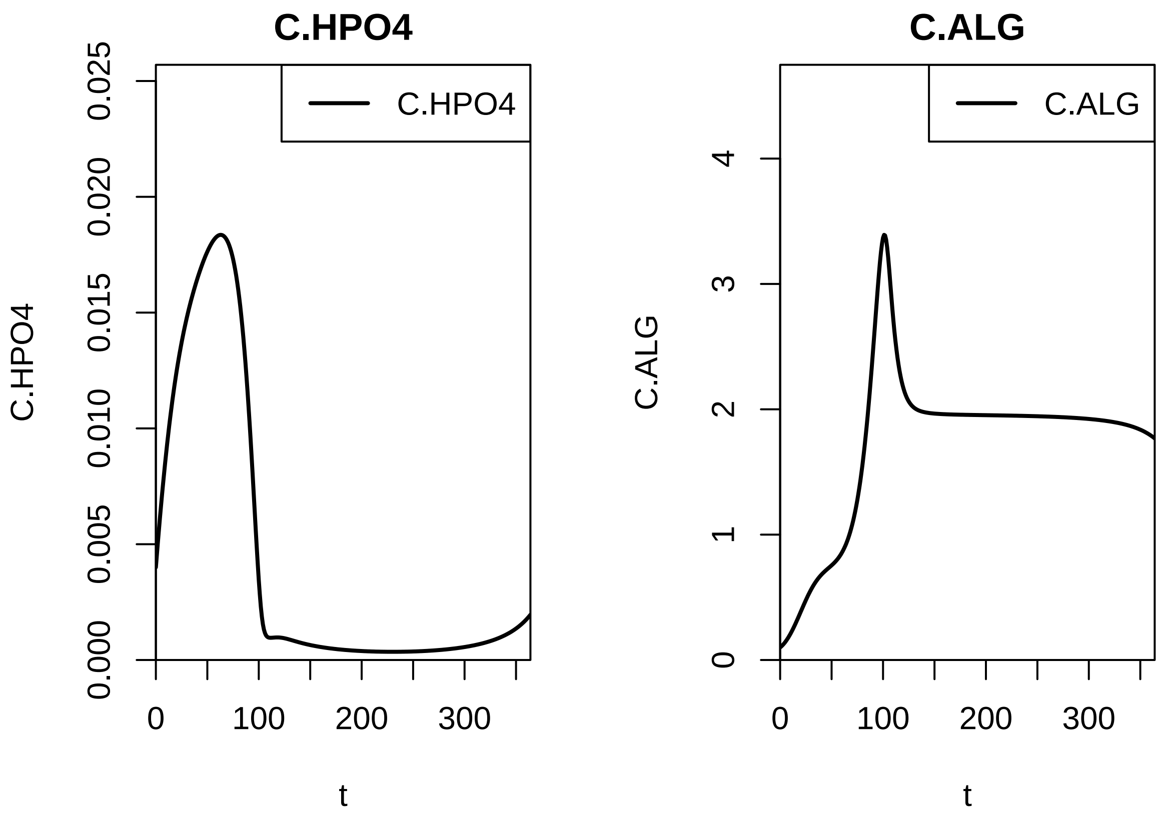
```
param$k.gro.ALG <- 0.8

# replace parameters in the system definition:

system.11.1.b@param <- param
```

Redo simulations and plot results:

```
# redo simulations and plot results:

res.11.1.b <- calcres(system.11.1.b)

plotres(res=res.11.1.b, colnames=list("C.HPO4","C.ALG")) # observe the time frame of this simulation
```



```
# plot only the first year of the simulation res.11.1.b
plotres(res=res.11.1.b[1:365,], colnames=list("C.HPO4","C.ALG"))
```



```
plotres(res      = res.11.1.b,     # plot to pdf file
        colnames = list("C.HPO4","C.ALG"),
        file     = "exercise_1_results_b1.pdf",
        width    = 10,
        height   = 5)
```

```
## pdf
##   2
```

```
# comparison of the two simulations:

plotres(res      = list(const=res.11.1.a.4y,dyn=res.11.1.b),
        colnames = list("C.HPO4","C.ALG"),
        file     = "exercise_1_results_ab.pdf",
        width    = 10,
        height   = 5)
```

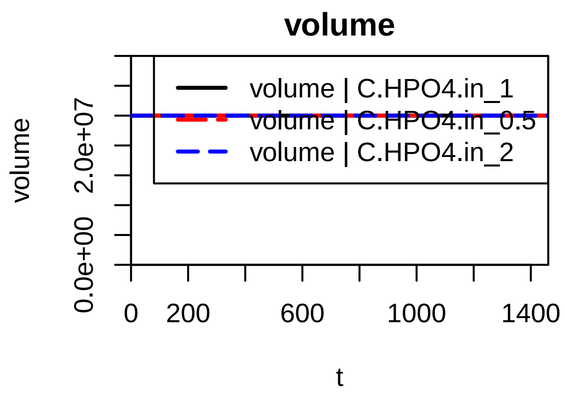
```
## pdf
##   2
```

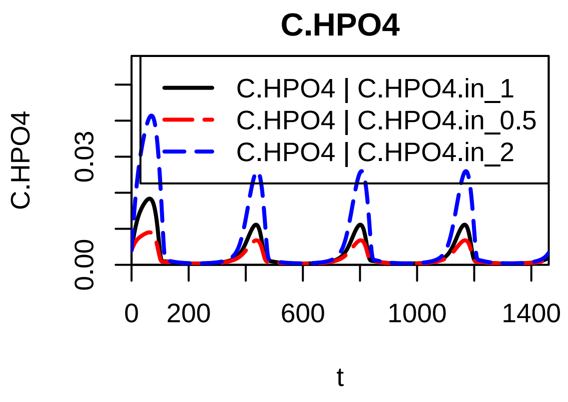
Task 5 - Homework: Simple sensitivity analysis

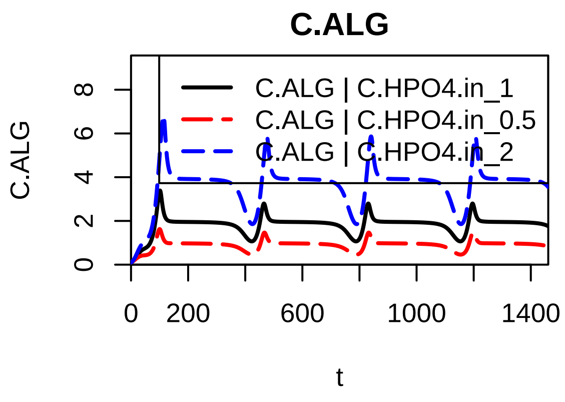
A sensitivity analysis is an analysis of how sensitive the model results are to changes in the parameter values. The simplest way of doing this is a so called “local sensitivity analysis”, where we change just one parameter at a time and keep the other parameters fixed, run the model and plot and analyse the results.

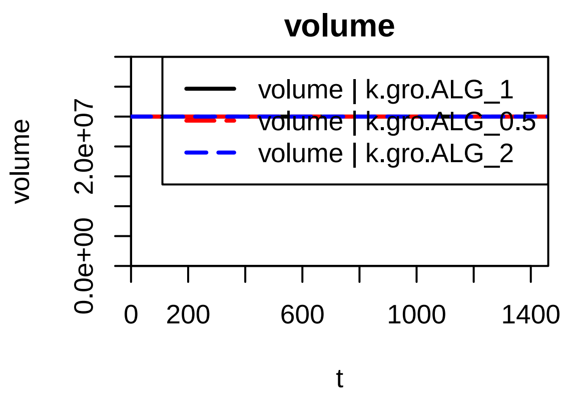
Do simulations with modified values of the parameters $C_{in,HPO_4^{2-}}$, $k_{gro,ALG}$, $k_{death,ALG}$, and $K_{HPO_4^{2-},ALG}$ using the function `calcsens()`. Type `?calcsens` to get the help file for this function and to see the default values. By default each parameter given in the list of `param.sens` is increased by a factor of 2 and decreased by a factor of 1/2. Try to understand the model responses to changes in the model input ($C_{in,HPO_4^{2-}}$) and process parameters ($k_{gro,ALG}$, $k_{death,ALG}$, $K_{HPO_4^{2-},ALG}$) under constant environmental conditions.

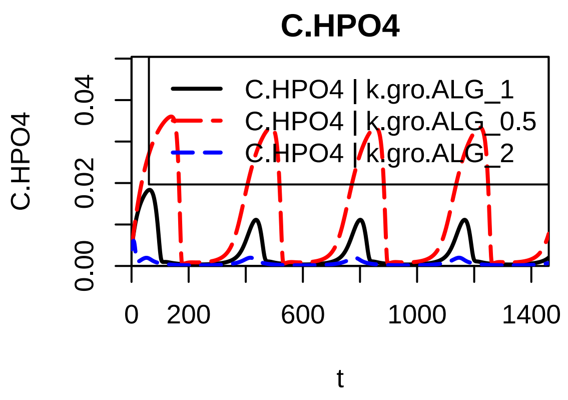
```
?calcsens # get help file for this function

# perform sensitivity analysis: TO BE COMPLETED

sens.res <- calcsens(system = system.11.1.b,
                     param.sens = c("C.HPO4.in","k.gro.ALG", "k.death.ALG", "K.HPO4"),
                     scaling.factors = c(1,0.5,2))

# plot the results

plotres(sens.res)
```

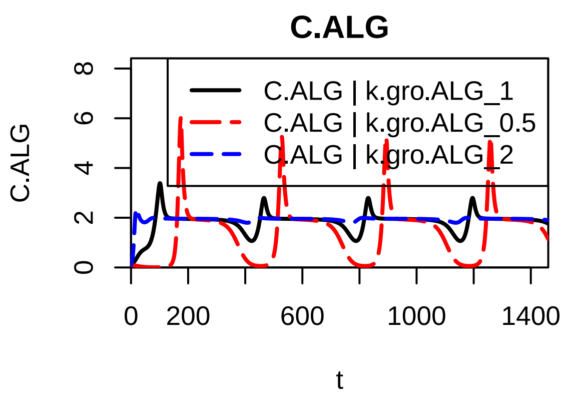


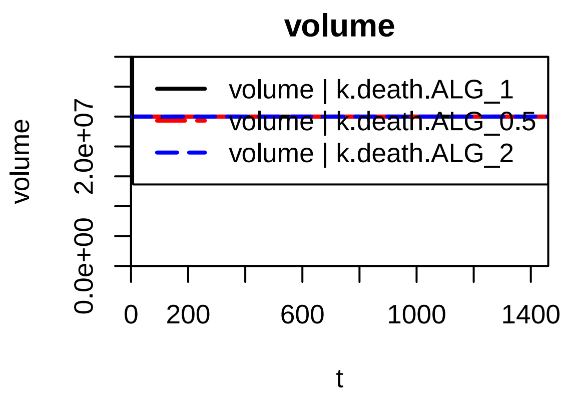


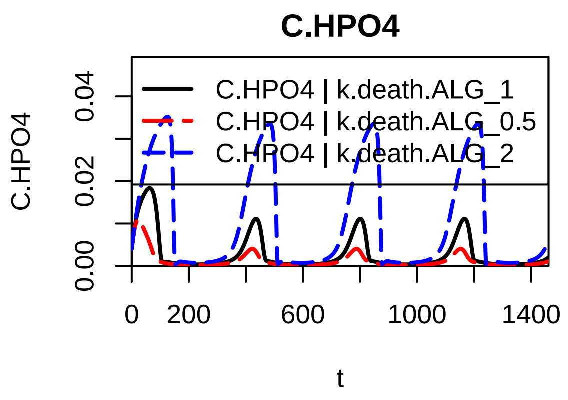


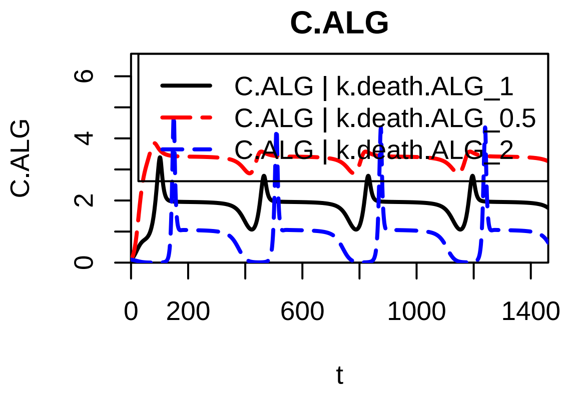

volume
volume | k.gro.ALG_1
volume | k.gro.ALG_0.5
volume | k.gro.ALG_2
volume
0.0e+00
2.0e+07
0
200
600
1000
1400
t

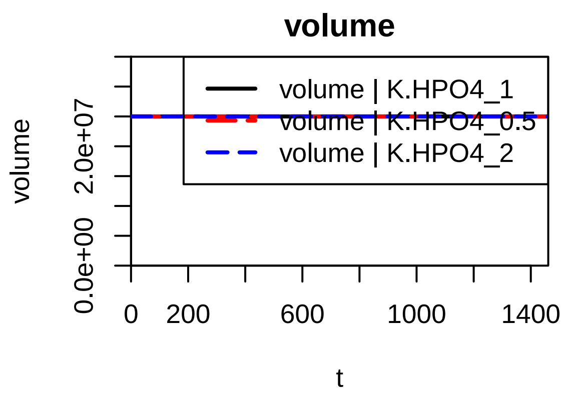

C.HPO4
C.HPO4 | k.gro.ALG_1
C.HPO4 | k.gro.ALG_0.5
C.HPO4 | k.gro.ALG_2
C.HPO4
0.00
0.02
0.04
0
200
600
1000
1400
t

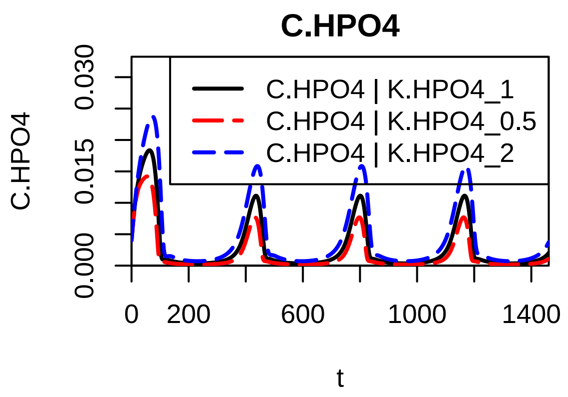

C.ALG
C.ALG | k.gro.ALG_1
C.ALG | k.gro.ALG_0.5
C.ALG | k.gro.ALG_2
C.ALG
0
2
4
6
8
0
200
600
1000
1400
t

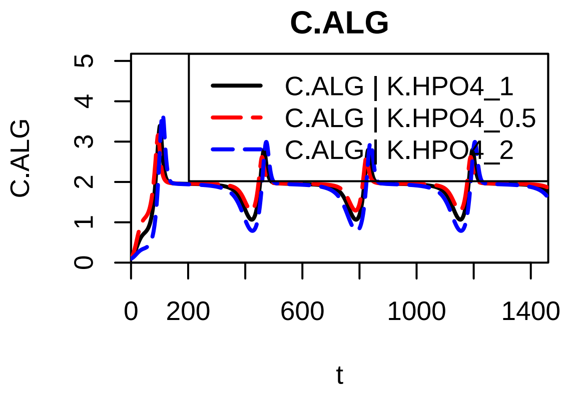

volume
volume | k.death.ALG_1
volume | k.death.ALG_0.5
volume | k.death.ALG_2
volume
0.0e+00
2.0e+07
0
200
600
1000
1400
t


C.HPO4
C.HPO4 | k.death.ALG_1
C.HPO4 | k.death.ALG_0.5
C.HPO4 | k.death.ALG_2
C.HPO4
0.00
0.02
0.04
0
200
600
1000
1400
t


C.ALG
C.ALG | k.death.ALG_1
C.ALG | k.death.ALG_0.5
C.ALG | k.death.ALG_2
C.ALG
0
2
4
6
0
200
600
1000
1400
t







Theory questions

1. How can you derive the total (net) transformation rate of $C_{HPO_4^{2-}}$ and C_{ALG} from the process table (Table 11.1) and the process rates (Table 11.2)? *Hint:* see equation (4.1) in the manuscript. What are the units?

2. Look at the state variables $C_{HPO_4^{2-}}$ and C_{ALG}. Which of them is more sensitive to the parameter $K_{HPO_4^{2-},ALG}$ and which of them is more sensitive to $C_{in,HPO_4^{2-}}$? Do you understand why?