Exercise 3: General solution of stoichiometric equations

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

March 18, 2026

Goals:

- Understand the concepts of stoichiometric calculations introduced in section 4.3.
- Be able to do stoichiometric calculations based on the general solution of stoichiometric equations (section 4.3.3) using the R package `stoichcalc`.
- Learn to be attentive to the results and to error messages and to correct the code accordingly (execute the code piece by piece and proceed only if the results are as expected and everything was correct).

Task 1: Simple stoichiometry using the stoichcalc-package

Study the implementation of the simple stoichiometry of algae growth as formulated in exercise 1. First, load the package `stoichcalc` (and install if it is not yet installed).

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

Next, we construct a composition matrix by defining substances as named vectors with the elemental content, build a list of all substances, and call the `stoichcalc` function `calc.comp.matrix` to compile the composition matrix. Note that you have to be very careful to choose units consistently across the model (we will feed the calculated stoichiometric coefficients into processes in `ecosim` in the next exercise).

```
# Construct composition matrix:

HPO4   <- c(P      = 1)                    # gP/gHPO4-P
ALG    <- c(P      = 0.01)                 # gP/gALG

subst.comp <- list(HPO4   = HPO4,
                   ALG    = ALG)

alpha.1 <- calc.comp.matrix(subst.comp)

print(alpha.1)
```

Alternatively, it would be possible to formulate the composition matrix directly. However, the procedure above has the advantage that we only have to specify the elements that are contained in a substance and not the ones that are zero. The elements that are zero will be automatically added by the function `calc.comp.matrix`.

We then calculate process stoichiometries for growth and respiration of algae by providing the composition matrix, specifying a process name, providing the names of the substances involved in the process, and specifying which stoichiometric coefficient is set to a specific value (recall that process stoichiometry is only unique up to a multiplicative factor and that we make it unique by setting one of the coefficients to +1 or -1; this makes the process rate equal to plus or minus the conversion rate of the associated substance).

```
# Calculate stoichiometric coefficients of selected processes:

nu.gro.ALG <-
```

```
   calc.stoich.coef(alpha       = alpha.1,
                    name        = "growth.ALG",
                    subst       = c("HPO4","ALG"),
                    subst.norm  = "ALG",
                    nu.norm     = 1)


# complete the missing code for respiration of algae
nu.resp.ALG <-
   calc.stoich.coef( ... )
```

Note the message that the stoichiometric coefficients have been calculated successfully.

Finally, we bind the stoichiometric coefficients of the different processes together to the stoichiometric matrix nu:

```
nu.1 <- rbind(nu.gro.ALG,
              nu.resp.ALG)

print(nu.1)
```

Note that the stoichiometry of respiration is just the negative of that of primary production.

Task 2: Stoichiometry of a complex process model

For more complicated stoichiometries, it is worth to define the composition parameters explicitly. This makes it much easier to adapt the composition later by changing the parameters and re-evaluating the stoichiometry. Note that the parameters are evaluated for calculating the stoichiometry and not carried-through as parameters. Changing the parameter without re-evaluating the stoichiometry does thus not have the desired effect.

We first define the mass fractions of oxygen, hydrogen, nitrogen and phosphorus for all organic substances:

```
# Complex stoichiometry
# ~~~~~~~~~~~~~~~~~~~~


# Set parameter values:

param <- list(a.O.ALG     = 0.50,         # gO/gALG
              a.H.ALG     = 0.07,         # gH/gALG
              a.N.ALG     = 0.06,         # gN/gALG
              a.P.ALG     = 0.005,        # gP/gALG
              a.O.ZOO     = 0.50,         # gO/gZOO
              a.H.ZOO     = 0.07,         # gH/gZOO
              a.N.ZOO     = 0.06,         # gN/gZOO
              a.P.ZOO     = 0.01,         # gP/gZOO
              a.O.POM     = 0.40,         # gO/gPOM
              a.H.POM     = 0.07,         # gH/gPOM
              a.N.POM     = 0.04,         # gN/gPOM
              a.P.POM     = 0.007,        # gP/gPOM
              Y.ZOO       = 0.2,          # gZOO/gALG
              f.e         = 0.4)          # gPOM/gALG
```

Then we calculate the carbon content as the difference to unity. It makes sense to do this for carbon as this is the largest mass fraction and thus does not bear the risk of becoming negative (if you would do it

for phosphorus, which is only about 1% of the mass or even less, one could easily get negative values when modifying carbon that is about 50% of the mass).

```
# choose carbon fractions to guarantee that the fractions sum to unity (eq. 4.45):

param$a.C.ALG = 1-(param$a.O.ALG+param$a.H.ALG+param$a.N.ALG+param$a.P.ALG)
param$a.C.ZOO = 1-(param$a.O.ZOO+param$a.H.ZOO+param$a.N.ZOO+param$a.P.ZOO)
param$a.C.POM = 1-(param$a.O.POM+param$a.H.POM+param$a.N.POM+param$a.P.POM)
```

This completes the definition of the mass fractions.

Next we define the composition of all substances to be considered for the calculation of the stoichiometries of the growth, respiration and death processes. Note that we have to be careful with the units. The natural unit to measure the amount of inorganic chemical compounds is moles. This is not the case for organic compounds as they consist of a large number of different organic molecules and we just consider their mean composition. For this reason, we measure organic compounds as dry mass. As we have to consider conservation of charge as well as conservation of the elements, we need to add charge to the composition vectors:

```
# Construct composition matrix:

NH4    <- c(H      = 4,                    # molH/molNH4
            N      = 1,                    # molN/molNH4
            charge = 1)                    # chargeunits/molNH4
NO3    <- c(O      = 3,                    # molO/molNO3
            N      = 1,                    # molN/molNO3
            charge = -1)                   # chargeunits/molNO3
HPO4   <- c(O      = 4,                    # molO/molHPO4
            H      = 1,                    # molH/molHPO4
            P      = 1,                    # molP/molHPO4
            charge = -2)                   # chargeunits/molHPO4
HCO3   <- c(C      = 1,                    # molC/molHCO3
            O      = 3,                    # molO/molHCO3
            H      = 1,                    # molH/molHCO3
            charge = -1)                   # chargeunits/molHCO3
O2     <- c(O      = 2)                    # molO/molO2
H      <- c(H      = 1,                    # molH/molH
            charge = 1)                    # chargeunits/molH
H2O    <- c(O      = 1,                    # molO/molH2O
            H      = 2)                    # molH/molH2O
ALG    <- c(C      = param$a.C.ALG/12,     # molC/gALG
            O      = param$a.O.ALG/16,     # molO/gALG
            H      = param$a.H.ALG,        # molH/gALG
            N      = param$a.N.ALG/14,     # molN/gALG
            P      = param$a.P.ALG/31)     # molP/gALG
ZOO    <- c(C      = param$a.C.ZOO/12,     # molC/gZOO
            O      = param$a.O.ZOO/16,     # molO/gZOO
            H      = param$a.H.ZOO,        # molH/gZOO
            N      = param$a.N.ZOO/14,     # molN/gZOO
            P      = param$a.P.ZOO/31)     # molP/gZOO
POM    <- c(C      = param$a.C.POM/12,     # molC/gPOM
            O      = param$a.O.POM/16,     # molO/gPOM
            H      = param$a.H.POM,        # molH/gPOM
            N      = param$a.N.POM/14,     # molN/gPOM
            P      = param$a.P.POM/31)     # molP/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(NH4    = NH4,
                   NO3    = NO3,
                   HPO4   = HPO4,
                   HCO3   = HCO3,
                   O2     = O2,
                   H      = H,
                   H2O    = H2O,
                   ALG    = ALG,
                   ZOO    = ZOO,
                   POM    = POM)

alpha.2 <- calc.comp.matrix(subst.comp)

print(alpha.2)
```

Note that this function just complemented the definitions specified above for all substances by zeros for those elements that were not explicitly listed in the definition.

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

param$Y.ALG.death = min(1,param$a.N.ALG/param$a.N.POM,param$a.P.ALG/param$a.P.POM)
param$Y.ZOO.death = min(1,param$a.N.ZOO/param$a.N.POM,param$a.P.ZOO/param$a.P.POM)

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

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

```
# Calculate stoichiometric coefficients of selected processes:

nu.gro.ALG.NH4 <-
   calc.stoich.coef(alpha       = alpha.2,
                    name        = "gro.ALG.NH4",
                    subst       = c("NH4","HPO4","HCO3","O2","H","H2O","ALG"),
                    subst.norm  = "ALG",
                    nu.norm     = 1)

nu.gro.ALG.NO3 <-
   calc.stoich.coef(alpha       = alpha.2,
                    name        = "gro.ALG.NO3",
                    subst       = c("NO3","HPO4","HCO3","O2","H","H2O","ALG"),
                    subst.norm  = "ALG",
                    nu.norm     = 1)

nu.resp.ALG <-
   calc.stoich.coef(alpha       = alpha.2,
                    name        = "resp.ALG",
                    subst       = c("NH4","HPO4","HCO3","O2","H","H2O","ALG"),
                    subst.norm  = "ALG",
                    nu.norm     = -1)

nu.death.ALG <-
```

```
   calc.stoich.coef(alpha       = alpha.2,
                    name        = "death.ALG",
                    subst       = c("NH4","HPO4","HCO3","O2","H","H2O","ALG","POM"),
                    subst.norm  = "ALG",
                    nu.norm     = -1,
                    constraints = list(c("ALG" = param$Y.ALG.death,
                                         "POM" = 1)))

nu.gro.ZOO <-
   calc.stoich.coef(alpha       = alpha.2,
                    name        = "gro.ZOO",
                    subst       = c("NH4","HPO4","HCO3","O2","H","H2O","ALG","ZOO","POM"),
                    subst.norm  = "ZOO",
                    nu.norm     = 1,
                    constraints = list(c("ZOO" = 1,
                                         "ALG" = param$Y.ZOO),
                                       c("POM" = 1,
                                         "ALG" = param$f.e)))

nu.resp.ZOO <-
   calc.stoich.coef(alpha       = alpha.2,
                    name        = "resp.ZOO",
                    subst       = c("NH4","HPO4","HCO3","O2","H","H2O","ZOO"),
                    subst.norm  = "ZOO",
                    nu.norm     = -1)

nu.death.ZOO <-
   calc.stoich.coef(alpha       = alpha.2,
                    name        = "death.ZOO",
                    subst       = c("NH4","HPO4","HCO3","O2","H","H2O","ZOO","POM"),
                    subst.norm  = "ZOO",
                    nu.norm     = -1,
                    constraints = list(c("ZOO" = param$Y.ZOO.death,
                                         "POM" = 1)))
```

Finally, we bind the stoichiometries together for the full stoichiometric matrix:

```
nu.2 <- rbind(nu.gro.ALG.NH4,
              nu.gro.ALG.NO3,
              nu.resp.ALG,
              nu.death.ALG,
              nu.gro.ZOO,
              nu.resp.ZOO,
              nu.death.ZOO)

print(round(nu.2,3))
```

This stoichiometric matrix should be checked in particular for the signs of the stoichiometric coefficients. As we chose the yields for death accordingly above, death of neither algae or zooplankton consumes phosphate or ammonium. However, we also have to check that it does not consume oxygen, which is not so easy to avoid. If it would consume oxygen, we would have to switch to the approach of using multiple fractions of dead organic matter (we do not have this problem if the composition of dead organic matter is the same as that of the dying organism).

Finally, we demonstrate the most frequently occurring errors. This was the correctly calculated stoichiometry

for zooplankton growth (rounding of the output makes it more pleasant to read):

```
# Correct stoichiometry of growth of zooplankton:

round(calc.stoich.coef(alpha       = alpha.2,
                       name        = "gro.ZOO",
                       subst       = c("NH4","HPO4","HCO3","O2","H","H2O","ALG","ZOO","POM"),
                       subst.norm  = "ZOO",
                       nu.norm     = 1,
                       constraints = list(c("ZOO" = 1,
                                            "ALG" = param$Y.ZOO),
                                          c("POM" = 1,
                                            "ALG" = param$f.e))),3)
```

If we forget a constraint, the program cannot calculate a unique solution:

```
# Missing constraint error for the stoichiometry of growth of zooplankton:

round(calc.stoich.coef(alpha       = alpha.2,
                       name        = "gro.ZOO",
                       subst       = c("NH4","HPO4","HCO3","O2","H","H2O","ALG","ZOO","POM"),
                       subst.norm  = "ZOO",
                       nu.norm     = 1,
                       constraints = list(c("ZOO" = 1,
                                            "ALG" = param$Y.ZOO))),3)
```

The same happens, if we include a substance that does not allow for a unique solution. In this case, including ammonium and nitrate does not allow the program to resolve in which form nitrogen should be released when respiring part of the algae:

```
# Too many substances error for the stoichiometry of growth of zooplankton:

round(calc.stoich.coef(alpha       = alpha.2,
                       name        = "gro.ZOO",
                       subst       = c("NH4","NO3","HPO4","HCO3","O2","H","H2O","ALG","ZOO","POM"),
                       subst.norm  = "ZOO",
                       nu.norm     = 1,
                       constraints = list(c("ZOO" = 1,
                                            "ALG" = param$Y.ZOO),
                                          c("POM" = 1,
                                            "ALG" = param$f.e))),3)
```

The other kind of problem occurs if we forget a substance, in this example ammonium:

```
# Missing substance error for the stoichiometry of growth of zooplankton:

round(calc.stoich.coef(alpha       = alpha.2,
                       name        = "gro.ZOO",
                       subst       = c("HPO4","HCO3","O2","H","H2O","ALG","ZOO","POM"),
                       subst.norm  = "ZOO",
                       nu.norm     = 1,
                       constraints = list(c("ZOO" = 1,
                                            "ALG" = param$Y.ZOO),
                                          c("POM" = 1,
                                            "ALG" = param$f.e))),3)
```

There is no solution as there is no substance available for the release of the respired nitrogen.

Questions

1. How do you find out, whether you need additional constraints to elemental mass balance and charge? What could be a drawback of the simplified approach to this question?
2. Where to get the required additional constraints from (if needed)?
3. Why do we add H^+, but not OH^- to the compounds considered for the calculation of stoichiometric coefficients?

Task 3: Homework: Extend the process stoichiometry to sulfur

Extend the process model implemented in task 2 to the consideration of sulfur (S), assuming the same content of 0.3% of S in algae, zooplankton and particulate organic matter (decrease the carbon content by 0.3%) and the uptake and release of sulfur in the form of sulfate SO_4^{2-} (release may also be in the form of hydrogen sulfide, H_2S, which later is oxidized to sulfate; we aggregate these processes).