# $^{18}O$ exchange between water reservoirs

Exercises Accompanying the Course Reaction Transport Modelling in the Hydrosphere

Lubos Polerecky, Utrecht University

February 2022

## Task 1 — solution

First, we define model parameters and empirical constants.

```
# model parameters
Aliq     <- 1           # liquid water area (m2)
Ptot     <- 101325      # total pressure (Pa)
v        <- 0           # wind speed (m h-1)
TK       <- 298         # temperature (K)

# empirical constants or parameters
Rgas     <- 8.314       # gas constant (J mol-1 K-1)
dHvap    <- 40.66e3     # molar enthalpy of vaporization of water (J mol-1)
c1       <- 0.0888*3600 # empirical constant in Eq. 2 (m h-1)
c2       <- 0.0783      # empirical constant in Eq. 2 (-)
xini     <- 0.002       # initial 18O atom fraction of liquid water
rho      <- 1e3         # water density (kg m-3)
MW       <- 18e-3       # molar weight of water (kg mol-1)
                        # rho/MW is assumed to be the same for 16O-H2O and 18O-H2O!
```

Then, we define auxiliary functions for calculating the equilibrium vapor pressure, amounts of water molecules in the gas and liquid phase, rate constants, and state variables.

```
Peq_vapor <- function(TK=300)                          # input temperature in K!
  101325 * exp( -40.66e3/8.314 * (1/TK - 1/373.15) ) # Clausius-Clapeyron equation

Nwater_vapor <- function(humid, TK, V){ # calculate Nvap from humidity, T, and volume
  humid*Peq_vapor(TK)*V/(TK*Rgas)       # based on the ideal gas law
}

Nwater_liquid <- function(V){           # calculate Nliq from volume
  rho*V/MW                              # using density and molar weight
}

kfbT <- function(TK, v=0, delta18O){    # rate constants
  kb16 <- Rgas*TK/dHvap*(c1+c2*v)       # water precipitation (m h-1)
  alpha <- 1/(delta18O*1e-3 + 1)        # convert delta18O (in permil!) to alpha
  kb18 <- kb16 * alpha                  # this is the source of overall fractionation!
  kf16 <- kb16 * Peq_vapor(TK)/(Rgas*TK)
  kf18 <- kf16                          # kf16 and kf18 assumed to be equal!
  return(c(kb16=kb16, kb18=kb18, kf16=kf16, kf18=kf18))
```

```
}

find_state <- function(Vtot, Vliq, humid, TK, x18vap, x18liq){
  Vgas <- Vtot - Vliq  # volume of the gas phase = Vtotal - Vliquid

  # total amounts of vapor, liquid, and air in the gas phase:
  Nvap <- Nwater_vapor(humid=humid, TK=TK, V=Vgas)
  Nliq <- Nwater_liquid(V=Vliq)
  Nair <- Ptot * Vgas/(Rgas*TK) - Nvap

  # output: state variables and parameters
  return( c(N18vap = Nvap*x18vap, N16vap = Nvap*(1-x18vap),
            N18liq = Nliq*x18liq, N16liq = Nliq*(1-x18liq),
            Nair   = Nair) ) # not changing but required
}
```

Finally, we define the model function based on the differential equations 8 and 9.

```
Water18Oexchange <-function(t, state, pars) {
  with (as.list(c(state, pars)),{

    # total amounts of vapor and liquid (mol)
    Nvap <- N18vap + N16vap
    Nliq <- N18liq + N16liq

    # current 18O atom fractions, isotope ratios, and delta vap vs. liq
    # note: ifelse() ensures no division by 0!
    x18vap           <- ifelse(Nvap>0,   N18vap/Nvap,           0)
    x18liq           <- ifelse(Nliq>0,   N18liq/Nliq,           0)
    R18vap           <- ifelse(N16vap>0, N18vap/N16vap,         0)
    R18liq           <- ifelse(N16liq>0, N18liq/N16liq,         0)
    delta_Vap_vs_Liq <- ifelse(R18liq>0, (R18vap/R18liq-1)*1e3, 0)

    # volume of the gas phase (air+vapor) at a given Nvap, Nair, TK, and Ptot
    Vgas <- Rgas*TK*(Nair+Nvap)/Ptot # based on the ideal gas law

    # process rates, mol h-1
    evap16 <- kf16*Aliq * (1-x18liq) * (Nliq/(Nliq+1e-5))
    evap18 <- kf18*Aliq *    x18liq  * (Nliq/(Nliq+1e-5))
    prec16 <- kb16*Aliq * (1-x18vap) * Nvap/Vgas
    prec18 <- kb18*Aliq *    x18vap  * Nvap/Vgas

    # time-derivatives
    dN18vap.dt <-  evap18 - prec18
    dN16vap.dt <-  evap16 - prec16
    dN18liq.dt <- -evap18 + prec18
    dN16liq.dt <- -evap16 + prec16
    dNair.dt   <- 0 # no gas dissolution in water!

    # extra output
    Pvap <- Nvap/Vgas * Rgas * TK  # partial pressure of vapor
    Pair <- Nair/Vgas * Rgas * TK  # partial pressure of air

    # return time-derivatives and ordinary variables as a list
```

```
    list(c(dN18vap.dt, dN16vap.dt, dN18liq.dt, dN16liq.dt, dNair.dt),

         # extra output quantities:

         # amounts of molecules
         Nwater = Nvap + Nliq,       # total amount of water (vapor+liquid)

         # volumes, m3
         Vgas = Vgas,                # gas phase
         Vliq = Nliq*MW/rho,         # liquid phase
         Vtot = Vgas+Nliq*MW/rho,    # total (gas+liquid)

         # pressures (Pa) and relative humidity
         Pvap = Pvap,                # partial pressure of vapor
         Pair = Pair,                # partial pressure of air
         Pgas = Pvap+Pair,           # total gas pressure (vapor+air)
         humid = Pvap/Peq_vapor(TK), # relative humidity

         # 18O atom fractions and delta vapor vs. liquid
         x18vap = x18vap,
         x18liq = x18liq,
         delta_Vap_vs_Liq = delta_Vap_vs_Liq # permil
       )
  })
}
```

## Model application

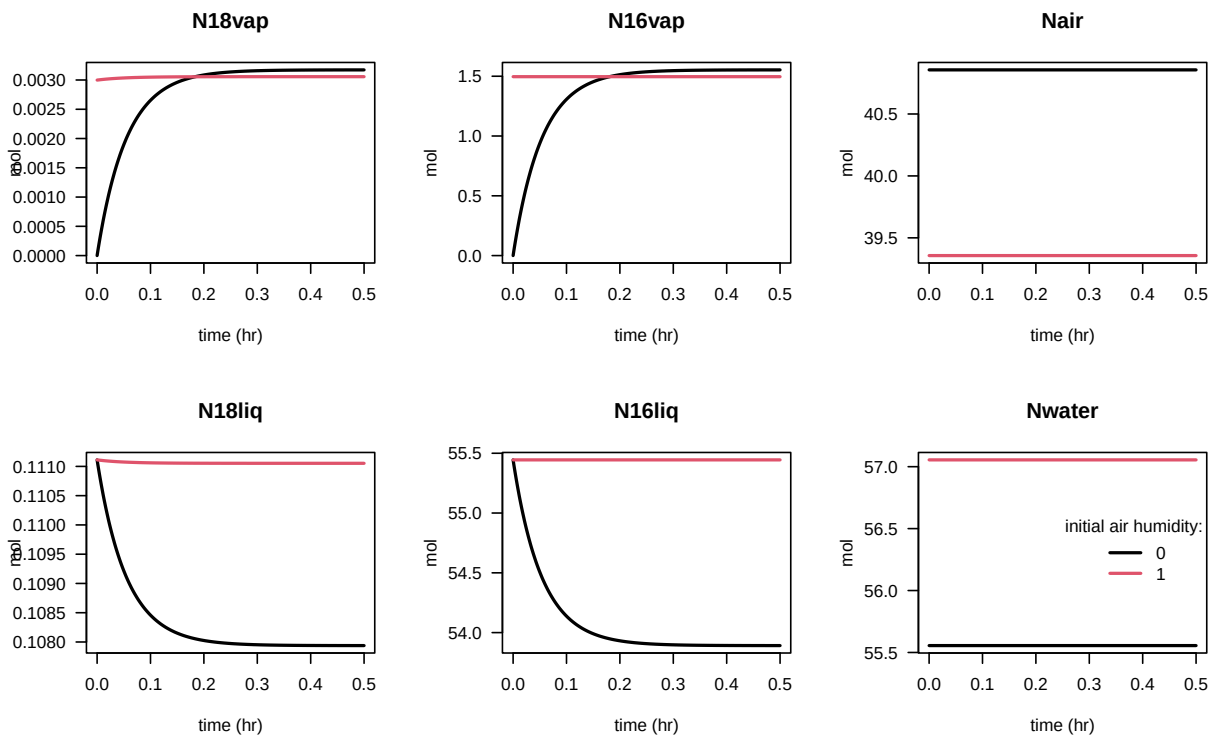
First, we use the above functions to evaluate the initial conditions and model parameters based on the definition of the problem. We solve the model for two initial values of the relative air humidity: 0 and 1.

```
# initial conditions
SVini1 <- find_state(Vtot=1, Vliq=1e-3, humid=1e-6, TK=TK, x18vap=0.002,     x18liq=0.002)
SVini2 <- find_state(Vtot=1, Vliq=1e-3, humid=1, TK=TK, x18vap=0.002, x18liq=0.002)

# model parameters
pars <- kfbT(TK=TK, delta18O=20)

# solve the model
require(deSolve)
outtimes <- seq(from = 0, to = 0.5, length.out = 400) # time in hr
out1 <- ode(y = SVini1, parms = pars, func = Water18Oexchange, times = outtimes)
out2 <- ode(y = SVini2, parms = pars, func = Water18Oexchange, times = outtimes)
```

## Results & Discussion

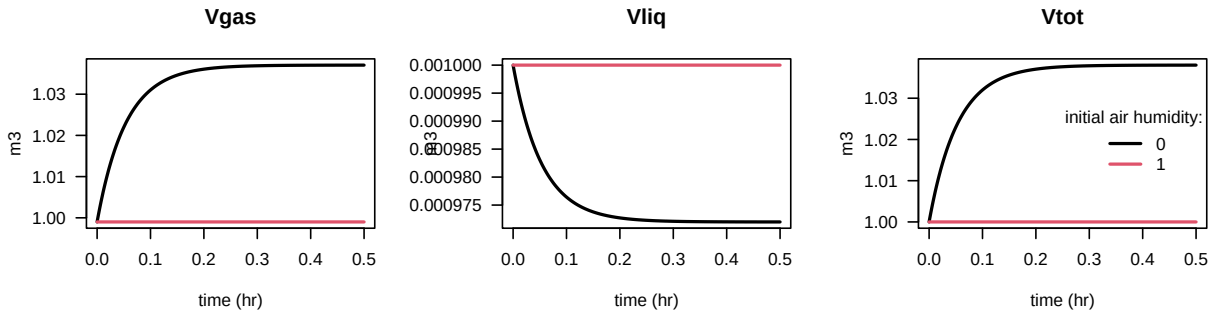
First, we plot the amounts of molecules to validate the mass balances. The results below show that the total amounts of water and air remain constant, as required. The total amount of water is slightly greater for the second run, since the air is initially humid and the initial volume of the liquid is the same as in the first run. The amount of air follows the opposite pattern.

```
plot(out1, out2,
     which=c("N18vap", "N16vap", "Nair", "N18liq", "N16liq", "Nwater"),
     xlab="time (hr)", lty=1, las=1, lwd=2, mfrow=c(2,3), ylab="mol")
legend("right", title="initial air humidity:", legend=c("0","1"),
       lty=1, lwd=2, col=1:2, bty="n")
```



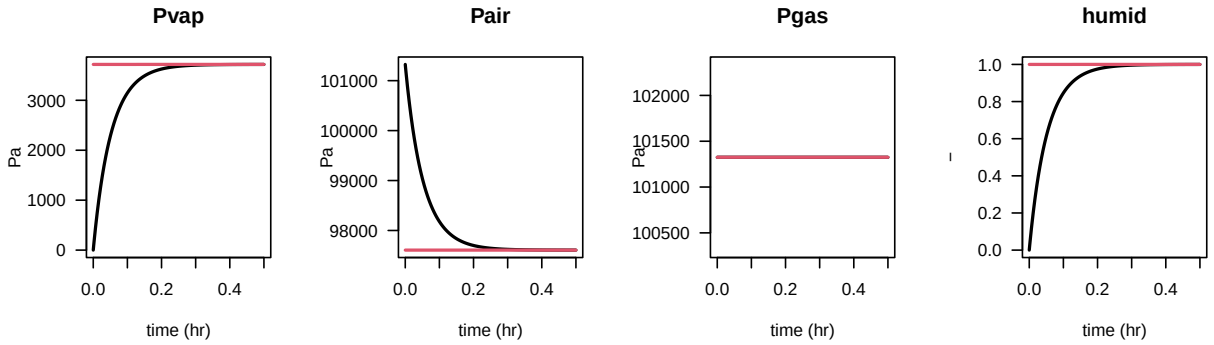

Next, we plot volumes (see graphs below). In the first scenario (initially dry air), the volume of the gas phase and the total volume increase, and the liquid water decreases, due to water evaporation occurring at a constant total pressure and temperature. There is no volume change in the second scenario because the air is initially fully saturated with water vapor, thus there is no net transfer of water between the liquid and gas phase.

```
plot(out1, out2,
     which=c("Vgas", "Vliq", "Vtot"),
     xlab="time (hr)", lty=1, las=1, lwd=2, mfrow=c(1,3), ylab="m3")
legend("right", title="initial air humidity:", legend=c("0","1"),
       lty=1, lwd=2, col=1:2, bty="n")
```

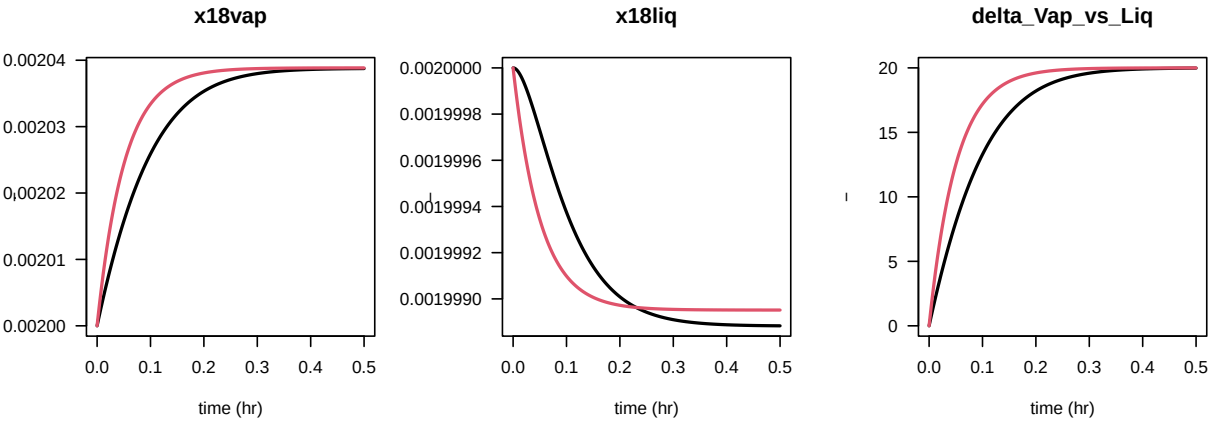


Next, we plot pressures and the relative air humidity to verify that the results make sense. In the first scenario, the partial pressure of water vapor increases, the partial pressure of air decreases, while the total gas pressure remains constant, as required. The relative air humidity increases from 0 to 1, as required. In the second scenario, there are no changes in the pressures or relative air humidity, as the initial condition corresponds to the equilibrium for the total amount of water vapor in the gas phase.

```
plot(out1, out2,
     which=c("Pvap", "Pair", "Pgas", "humid"),
     xlab="time (hr)", lty=1, las=1, lwd=2, mfrow=c(1,4),
     ylab=c(rep("Pa",3),"-"))
```




```
plot(out1, out2,
     which=c("x18vap", "x18liq", "delta_Vap_vs_Liq"),
     xlab="time (hr)", lty=1, las=1, lwd=2, mfrow=c(1,3), ylab="-")
```




## Conclusion

The model results illustrate that much more water evaporation and precipitation occurs than "meets the eye"! The exchange occurs on the time-scale of minutes, consistent with our daily experience.

# Task 2 — solution

## Parameters describing the experimental setup

```
V3    <- 1      # air volume (m3)
P     <- 101325 # air pressure (Pa)
Rgas  <- 8.314  # gas constant (SI units)
TK    <- 293.15 # temperature (K)
fH2O  <- 0.01   # molar fraction of water in air at 100% humidity
kb    <- 10     # rate constant of water precipitation (m d-1)
A1    <- 1      # area of the water reservoir 1 (m2)
                # (corresponds to 3*4 subcores, each with inner diameter 1.2 cm)
A2    <- 1      # area of the water reservoir 2 (m2)
                # square of 25x25 cm
h1    <- 1e-3   # height of the water reservoir 1 (m)
h2    <- 1e-3   # height of the water reservoir 2 (m)
xini  <- 0.002  # natural 18O atom fraction
x1ini <- 0.002  # initial 18O atom fraction in water reservoir 1
x2ini <- 0.002  # initial 18O atom fraction in water reservoir 2
x3ini <- 0.002  # initial 18O atom fraction of water vapor (=water reservoir 3)
rho   <- 1e3    # water density (kg m-3)
MW    <- 18e-3  # molar weight of water (kg mol-1)
                # rho/MW is assumed to be the same for 16O-H2O and 18O-H2O!
```

## Derived parameters

```
c3eq  <- P/(Rgas*TK) * fH2O # density of H2O molecules in air at 100% humidity (mol/m3)
V1    <- A1*h1              # volume of water in reservoir 1 (m3)
V2    <- A2*h2              # volume of water in reservoir 2 (m3)
# note: the density of particles (N/V=rho/MW) is assumed to be the same for 16O-H2O and 18O-H2O!
N1ini <- rho*V1/MW          # initial amount of H2O molecules in reservoir 1 (mol)
N2ini <- rho*V2/MW          # initial amount of H2O molecules in reservoir 2 (mol)
N3eq  <- c3eq*V3            # equilibrium amount of H2O molecules in reservoir 3 (mol)
```

## Model parameters

```
pars <- c(delta = 0, # delta = steady state (R3/R1-1)*1e3, in permil!
          f1    = 1, # consider reservoir 1
          f2    = 0  # consider reservoir 2
          )
```

## Initial conditions

```
# function to calculate initial state based on initial humidity of the air
humidityH2Oini <- function(humidity=1){
  # all values are in moles (16O-H2O and 18O-H2O isotopes separately)
  H2Oini <- c(N1_16 = N1ini*(1-x1ini),
              N1_18 = N1ini*x1ini,
```

```
              N2_16 = N2ini*(1-x2ini),
              N2_18 = N2ini*x2ini,
              N3_16 = N3eq*(1-x3ini) * humidity,
              N3_18 = N3eq*x3ini     * humidity
             )
  return(H2Oini)
}
```

## Model definition

```
WaterExchangeModel <-function(t, state, pars) {
  # t: time, state: state variables, pars: model parameters
  with (as.list(c(state, pars)),{

    # convert delta (in permil!) to alpha = kb_18/kb_16
    alpha <- 1/(delta*1e-3 + 1)

    # calculate all rate constants based on the rate constant
    # of water precipitation (kb), the fractionation factor (alpha),
    # and the equilibrium concentration c3eq
    kb_16 <- kb
    kb_18 <- kb * alpha
    kf_16 <- kb * c3eq
    kf_18 <- kb * c3eq

    # current amounts of H2O molecules in all reservoirs
    N1 <- N1_16+N1_18
    N2 <- N2_16+N2_18
    N3 <- N3_16+N3_18
    # current 18O atom fractions in all reservoirs
    x1 <- N1_18/N1
    x2 <- N2_18/N2
    x3 <- N3_18/N3

    # rate expressions [mol/d]

    # exchange between reservoir 1 (water) and 3 (water vapour)
    Evap1_16 <- kf_16 * (1-x1) * A1 * (N1>0)
    Evap1_18 <- kf_18 *    x1  * A1 * (N1>0)
    Cond1_16 <- kb_16 * (1-x3) * A1 * N3/V3
    Cond1_18 <- kb_18 *    x3  * A1 * N3/V3

    # exchange between reservoir 2 (water) and 3 (water vapour)
    Evap2_16 <- kf_16 * (1-x2) * A2 * (N2>0)
    Evap2_18 <- kf_18 *    x2  * A2 * (N2>0)
    Cond2_16 <- kb_16 * (1-x3) * A2 * N3/V3
    Cond2_18 <- kb_18 *    x3  * A2 * N3/V3

    # Time-derivatives: dN/dt = production - consumption [mol/d]
    dN1_16.dt <- (-Evap1_16 + Cond1_16)*f1
    dN1_18.dt <- (-Evap1_18 + Cond1_18)*f1
```

```
    dN2_16.dt <- (-Evap2_16 + Cond2_16)*f2
    dN2_18.dt <- (-Evap2_18 + Cond2_18)*f2
    dN3_16.dt <- ( Evap1_16 - Cond1_16)*f1 + (Evap2_16 - Cond2_16)*f2
    dN3_18.dt <- ( Evap1_18 - Cond1_18)*f1 + (Evap2_18 - Cond2_18)*f2

    # return time-derivatives and ordinary variables as a list
    list(c(dN1_16.dt, dN1_18.dt,
           dN2_16.dt, dN2_18.dt,
           dN3_16.dt, dN3_18.dt),

         # number of molecules
         Ntot_16 = N1_16+N2_16+N3_16, # total 16O-water
         Ntot_18 = N1_18+N2_18+N3_18, # total 18O-water
         N1 = N1_16 + N1_18,          # total water molecules, liquid 1
         N2 = N2_16 + N2_18,          # total water molecules, liquid 2
         N3 = N3_16 + N3_18,          # total water molecules, vapor
         Ntot = N1_16+N2_16+N3_16+N1_18+N2_18+N3_18, # all water molecules

         # 18O atom fractions
         x1 = N1_18/(N1_16+N1_18),
         x2 = N2_18/(N2_16+N2_18),
         x3 = N3_18/(N3_16+N3_18),

         # delta values (permil): delta_a_vs_b = (Ra/Rb-1)*1e3
         delta_Vap_vs_LiqA =  ( (N3_18/N3_16) / (N1_18/N1_16) - 1 ) * 1e3,
         delta_LiqB_vs_LiqA = ( (N2_18/N2_16) / (N1_18/N1_16) - 1 ) * 1e3
         )
  })
}
```

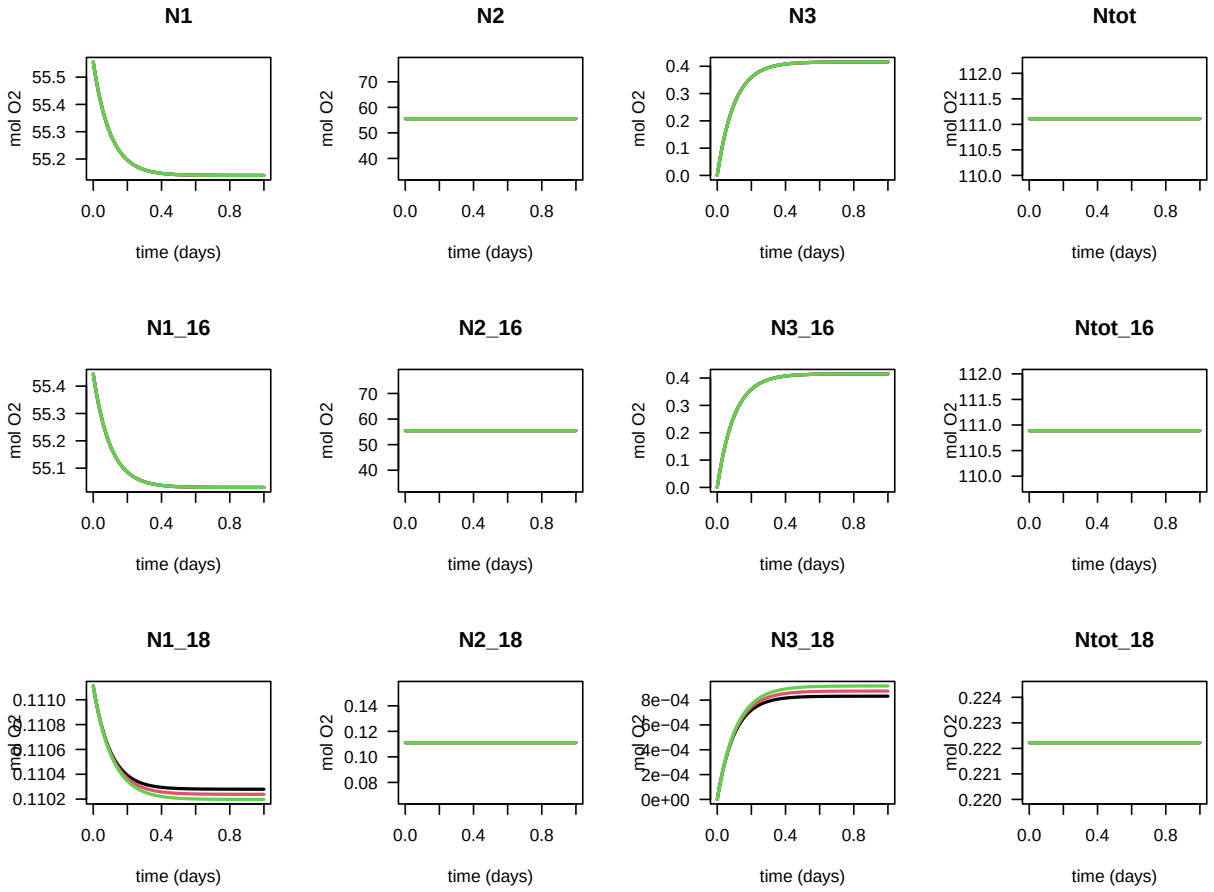
## Model solution

We calculate the dynamic solution for the initial humidity close to 0 and three values of the equilibrium fractionation factor $\delta$ (0, 50, and 100 permil):

```
require(deSolve)
outtimes <- seq(from = 0, to = 1, length.out = 100) # time in days
SVini <- humidityH2Oini(humidity = 1e-6)
pars1 <- pars2 <- pars
pars1["delta"] <- 50
pars2["delta"] <- 100
out0 <- ode(y = SVini, parms = pars,  func = WaterExchangeModel, times = outtimes)
out1 <- ode(y = SVini, parms = pars1, func = WaterExchangeModel, times = outtimes)
out2 <- ode(y = SVini, parms = pars2, func = WaterExchangeModel, times = outtimes)
```

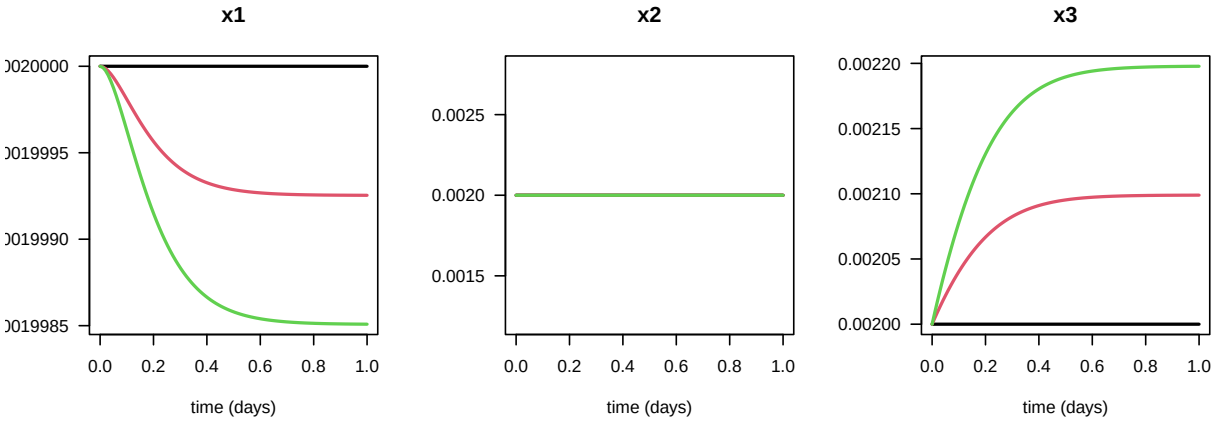
We plot the number of water molecules in the model reservoirs:

```
plot(out0, out1, out2,
     which=c("N1",   "N2",   "N3",   "Ntot",
             "N1_16","N2_16","N3_16","Ntot_16",
             "N1_18","N2_18","N3_18","Ntot_18"),
     xlab="time (days)", ylab="mol O2", lty=1, las=1, lwd=2, mfrow=c(3,4))
```




We plot the $^{18}O$ atom fractions:

```
plot(out0, out1, out2, which=c("x1","x2","x3"),
     xlab="time (days)", las=1, lwd=2, lty=1, mfrow=c(1,3))
```




Finally, we plot the $\delta$ values:

```
plot(out0, out1, out2,
     which=c("delta_LiqB_vs_LiqA","delta_Vap_vs_LiqA"),
     xlab="time (days)", las=1, lwd=2, lty=1, mfrow=c(1,2))
```

```
abline(h=pars1["delta"], lty=2, col=2)
abline(h=pars2["delta"], lty=2, col=3)
legend("bottomright", title="delta (permil):",
       legend=c(sprintf("%.1f",pars["delta"]),
                sprintf("%.1f",pars1["delta"]),
                sprintf("%.1f",pars2["delta"])),
       lty=1, lwd=2, col=1:3, bty="n")
```