Fractionation during decomposition : theoretical simulations

Lucas Deschamps, Jennifer Paillassa, Vincent Maire, Esther Lévesque

2024-10-08

```
library(tidyverse) # To manipulate and plot data
library(readxl) # To import xlsx data frames
library(ggtext) # To write beautiful axis with scientific notation
library(deSolve) # To numerically solve differential equations
```

Variation of C/N with decomposition

The first model presented describes the differential loss of C and N during decomposition. It is represented by a system of two differential equations.

$$\frac{\delta C}{\delta t} = -p_C \cdot k \cdot C \tag{1}$$

$$\frac{\delta N}{\delta t} = -p_N \cdot k \cdot N \tag{2}$$

With C and N being respectively the size of the C and N pools ($g \cdot kg^{-1}$), k ($year^{-1}$) the decomposition rate, p_C and p_N (0-1) the proportion of decomposed C and N not returned to OM during decomposition, lost by respiration and mineralization, respectively. The solution to such equation is an exponential decay :

$$C(t) = C_0 \cdot e^{-p_C \cdot k \cdot t} \tag{3}$$

$$N(t) = N_0 \cdot e^{-p_N \cdot k \cdot t} \tag{4}$$

Where C_0 and N_0 are the initial C and N stocks in $g \cdot kg^{-1}$, respectively.

Such equation can be implemented like this :

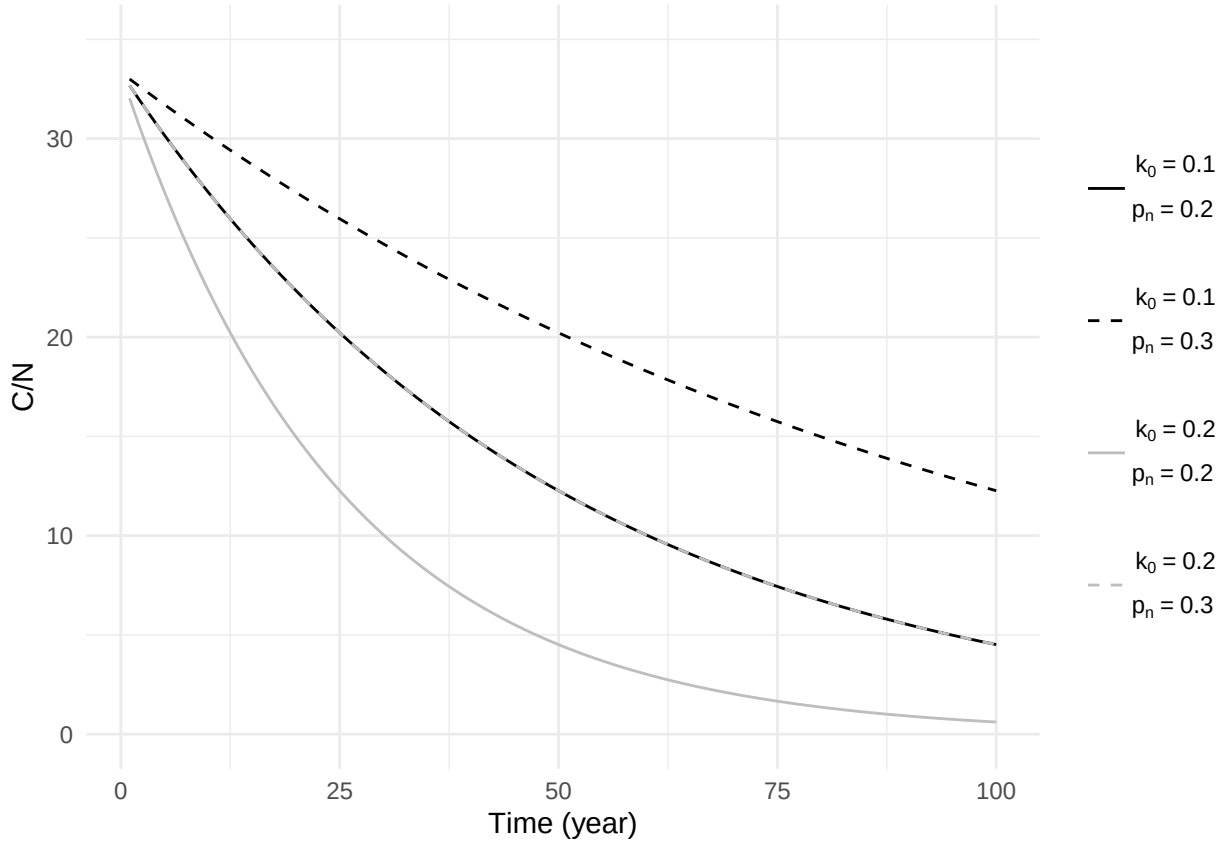
```
exponential.decay <- function(X_0, k, p, t){
  X_0 * exp(-p*k*t)
}
```

We can compute the results by creating a data frame containing different parameters values. We choose decomposition rates of 0.1 and 0.2 (`k`), 0.4 as the proportion of C loss (`p_C`), and 0.2 and 0.3 as the proportion of N loss (`p_N`). Initial stocks of C (`C_0`) and N (`N_0`) were 100 and 3 $g \cdot kg^{-1}$, with a C/N of approximately 33.

```r
## Create a data frame to store results
D_t_CN <- tibble(
  t = rep(1:100, 4),
  k = rep(c(0.1, 0.1, 0.2, 0.2), each = 100),
  p_N = rep(c(0.2, 0.3, 0.2, 0.3), each = 100)
) %>%
  mutate(C_0 = 100, N_0 = 3, p_C = 0.4,
         cat = paste("k=", k, "\npn=",p_N)) %>%
  mutate(C = exponential.decay(C_0, k = k, p = p_C, t = t),
         N = exponential.decay(N_0, k = k, p = p_N, t = t),
         CN = C/N)
```

We can then plot the results:

```r
(p_t_CN <- D_t_CN %>% ggplot(aes(x = t, y = CN)) +
   ## Add a line by values of parameters
  geom_line(aes(color = cat, linetype = cat)) +
   ## Change theme
  theme_minimal() +
  ## Adjust y axis
  ylim(0,35) +
  ## Name the axes
  labs(x = "Time (year)", y = "C/N") +
  ## Adjust the legend
  theme(legend.title = element_blank(),
        legend.text = element_text(margin = margin(t = 0.4, b = 0.4, unit = "cm"))) +
  ## Create a clear legend
  scale_color_manual(values = c("black", "black", "grey", "grey"),
                     labels = expression(
                       atop(~k[0]~"="~ 0.1,p[n]~"="~0.2),
                       atop(~k[0]~"="~ 0.1,p[n]~"="~0.3),
                       atop(~k[0]~"="~ 0.2,p[n]~"="~0.2),
                       atop(~k[0]~"="~ 0.2,p[n]~"="~0.3))) +
    scale_linetype_manual(values = c(1,2,1,2),
                          labels = expression(
                            atop(~k[0]~"="~ 0.1,p[n]~"="~0.2),
                            atop(~k[0]~"="~ 0.1,p[n]~"="~0.3),
                            atop(~k[0]~"="~ 0.2,p[n]~"="~0.2),
                            atop(~k[0]~"="~ 0.2,p[n]~"="~0.3),)
    )
)
```



Variation of C/N with depth

We can then adapt this model to add a advective term and simulate the variation of C/N with soil depth (z).

$$\frac{\delta C}{\delta t} = -v \cdot \frac{\delta C}{\delta z} - p_C \cdot k \cdot C \tag{5}$$

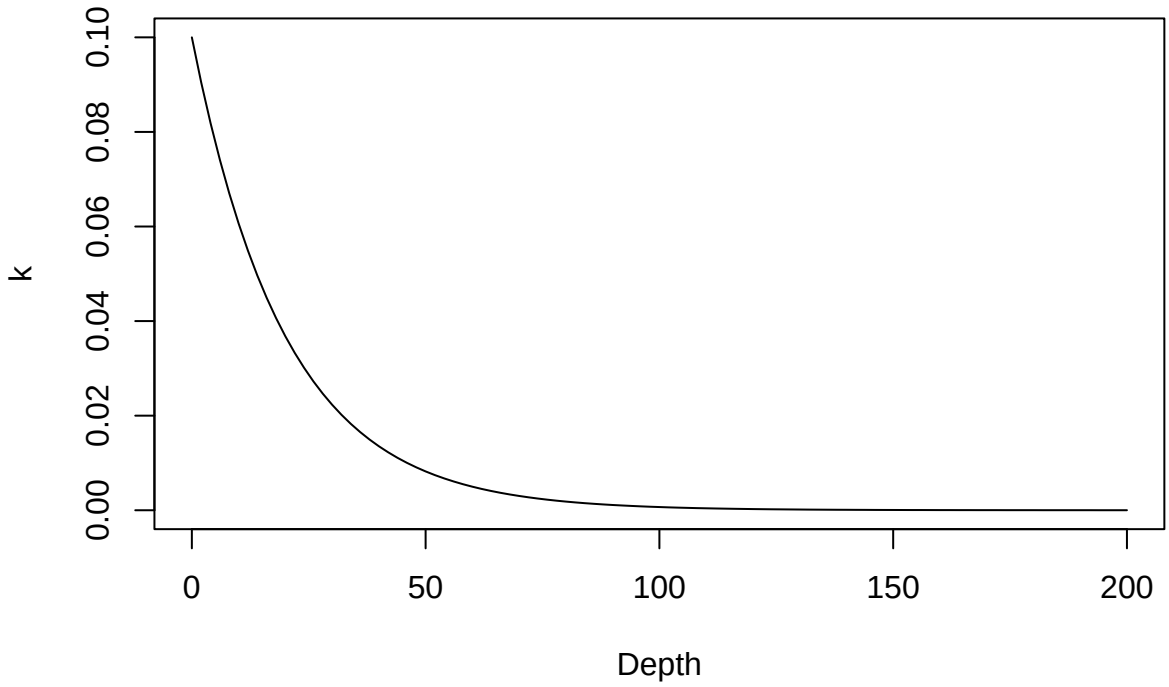
$$\frac{\delta N}{\delta t} = -v \cdot \frac{\delta N}{\delta z} - p_N \cdot k \cdot N \tag{6}$$

With v being the advection rate ($cm \cdot year^{-1}$). To accommodate the decreasing of decomposition with depth, which almost stops in the permafrost, k becomes a function of depth :

$$k = k_0 \cdot e^{-\frac{z}{z_h}}$$

With k_0 being the basal decomposition rate, and z_h a depth control parameter, representing the depth at which k_0 is halved. The function can be easily visualized that way.

```
k_func <- function(x, k_0, z_h) k_0 * exp(-x/z_h)
curve(k_func(x, 0.1, 20), from = 0, to = 200, ylim = c(0, 0.1), xlab = "Depth", ylab = "k")
```



This system of *partial differential equations* can be solved numerically using the algorithms in the `deSolve` package. For this we first need to define a model (the system of differential equations).

The advection process is approximated through dicretization, dividing the first meter of the soil by 100 1 cm wide boxes. The flux value at the boundary of the uppermost cell is the input flux `flux_X_fun(t)`, for which we will define constants. The value of fluxes between other cells is equal to the value of the above cell times the advection rate parameter `v`. Model code is inspired from sections 6.6.4 and 6.7.3 in Soetaert *et al.* (2009).

```
# Model equation
model_CN <-function(t, State, Pars)
{
  with(as.list(c(State, Pars)), {
    # Extract states and k
    C <- State[1:numboxes]
    N <- State[(numboxes+1):(2*numboxes)]
    k <- pars[2:(2+99)]
    # Compute vertical fluxes
    Flux_C   <- c(flux_C_fun(t), v*C)
    Flux_N <- c(flux_N_fun(t), v*N)
    # Compute rate of change
    dC      <- -diff(Flux_C)/delx  - k*p_C*C
    dN      <- -diff(Flux_N)/delx  - k*p_N*N
    return(list(c(dC, dN)))     # result
  })
}  # end of model
```

We will define the tickness and the number of boxes :

```
delx      <- 1    # cm      thickness of boxes
numboxes  <- 100

# depth to air-sea interface of centre of boxes, m
Depth  <- seq(from=0,by=delx,length.out=numboxes)  # sequence, 1 cm intervals
```

And then define the constant boundary conditions, that is C (`C_B`) and N (`N_B`) input from plants at the surface. We will consider plant input is equal to $100gC \cdot year^{-1}$, with a C/n of 30. We will simulate the evolution of the soil column during 1000 years. The boundary conditions needs to be defined as a function returning a value for any timestep.

```
# Vector of timesteps
times <- 1:1000
# Vector and function of C input
flux_C <- rep(100, times = length(times))
flux_C_fun <- approxfun(x = times, y = flux_C)
# Vector and function of N input
flux_N <- flux_C/30
flux_N_fun <- approxfun(x = times, y = flux_N)
# What does function returns?
flux_C_fun(1:5)
```

```
## [1] 100 100 100 100 100
```

```
flux_N_fun (1:5)
```

```
## [1] 3.333333 3.333333 3.333333 3.333333 3.333333
```

We then need to define the initial C and N concentrations along the soil columns in a vector. We will consider a bare soil, with no C and N.

```
state <- c(rep(0, times=numboxes), # Initial C concentration
           rep(0,times=numboxes)) # Initial N Concentration
```

We then have to define the parameters. We will explore the consequences of different value of k_0 and p_N, so we will store different sets of parameters in a list.

```
# Define the values of k_0 and p_N to test
k_0_vec <- c(0.1, 0.2, 0.1, 0.2)
p_N_vec <- c(0.2, 0.2, 0.3, 0.3)
# Create a list to store parameters
Params <- list()

# Create a loop to store parameters values
for(i in 1:length(k_0_vec)){
  Params[[i]] <- c(
  v = 0.5,
  k = k_func(Depth, k_0_vec[i], 20),
  p_C = 0.4,
  p_N = p_N_vec[i]
)
}
```

And estimate the solution using the default solver. We will use a loop to go through the differens values of parameters.

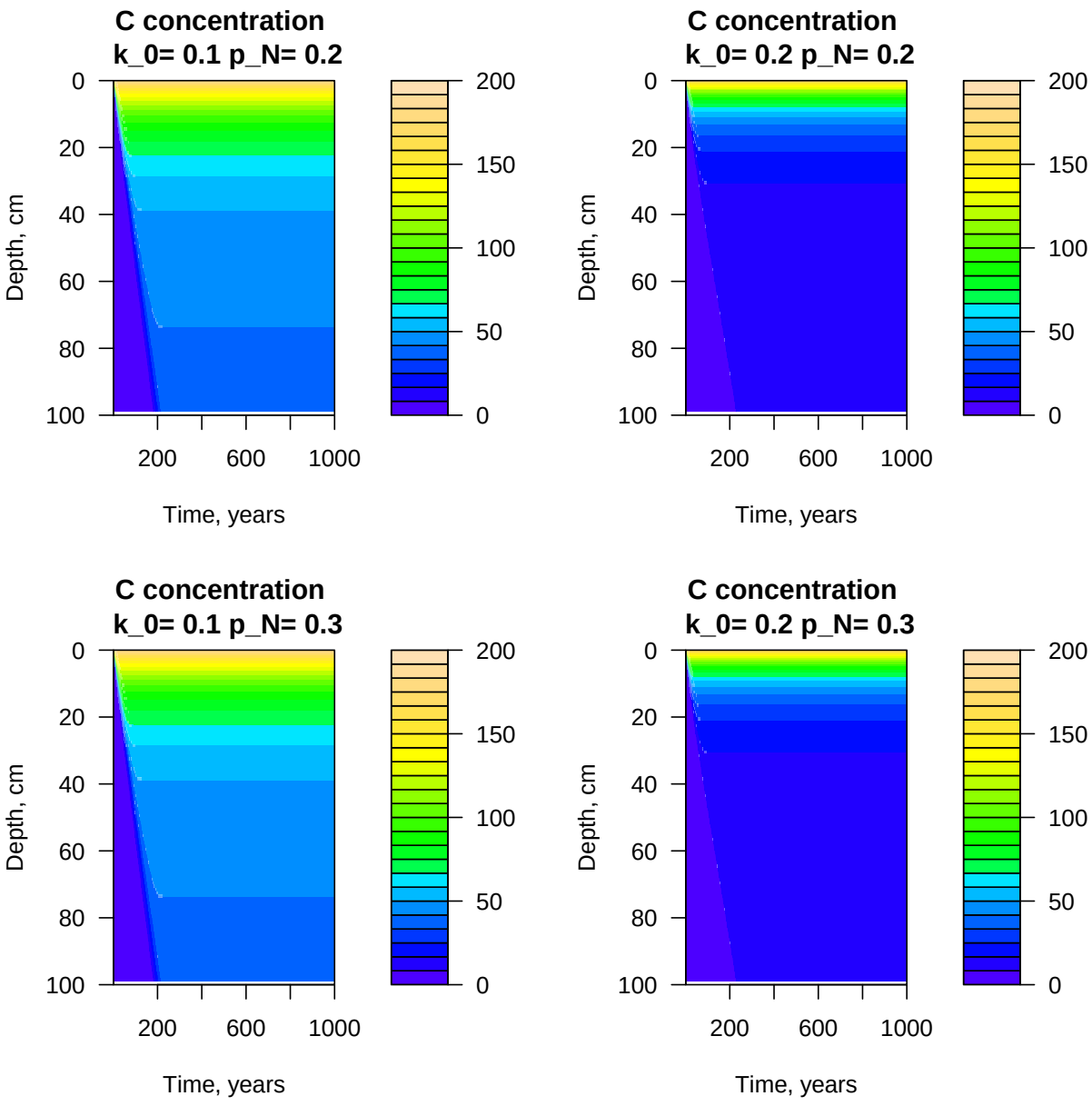
```
# Create a list to store simulation results
out <- list()
for(i in 1:length(Params)){
  pars <- Params[[i]]
  out[[i]] <- ode.1D(y = state, times = times, func = model_CN, parms = pars,
             dimens = numboxes)
}
```

We can gather the C and N concentration in lists from the big matrix output, and compute C/N. The first column of the output is the time step, so concentrations start at columns 2.

```
# Create lists to store results
Conc_C <- list()
Conc_N <- list()
CN <- list()

for(i in 1:length(out)){
  Conc_C[[i]] <- out[[i]][,2:(numboxes+1)]
  Conc_N[[i]] <- out[[i]][,((numboxes+2)):(2*numboxes+1)]
  CN[[i]] <- Conc_C[[i]]/Conc_N[[i]]
}
```

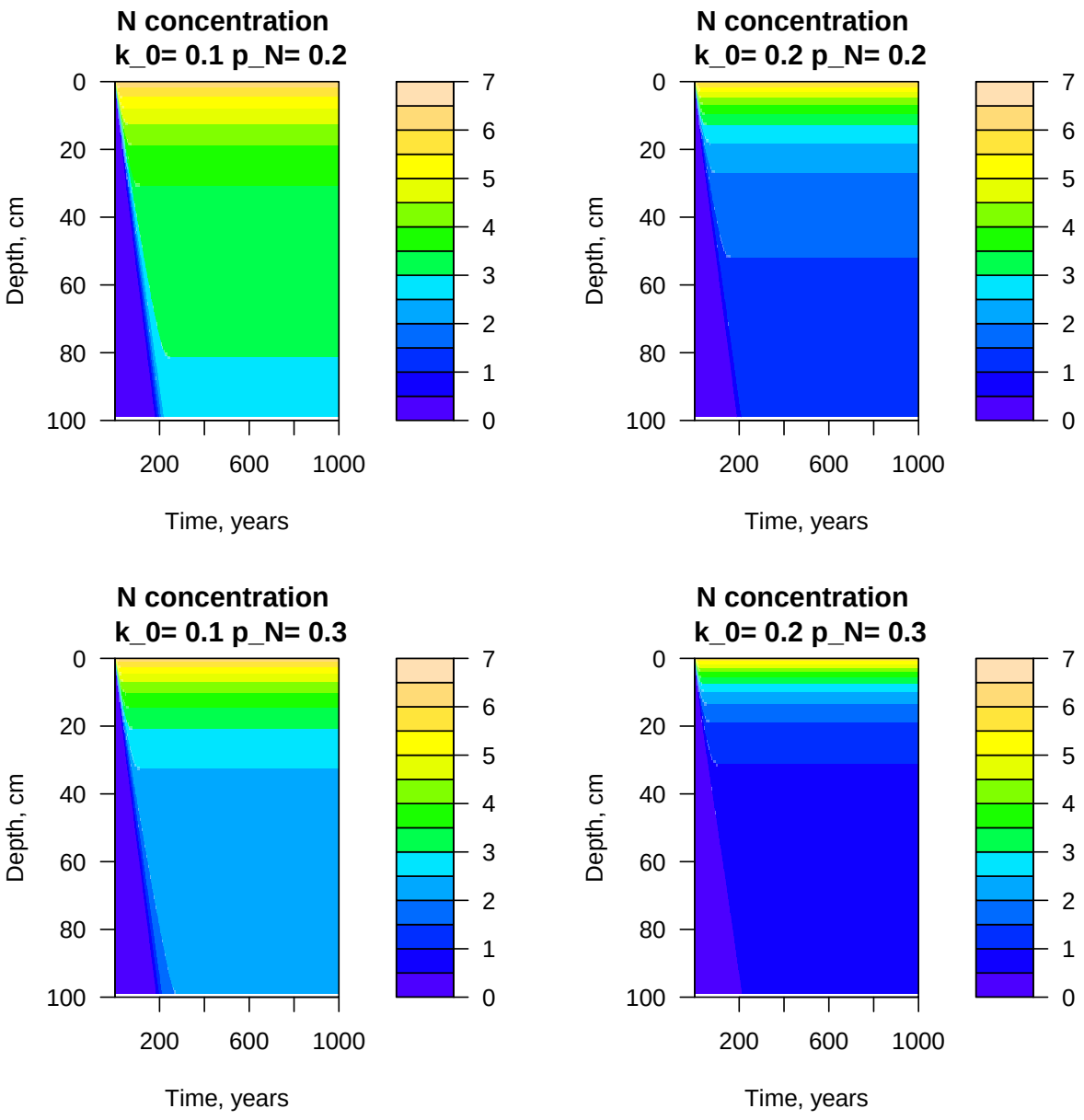
We can then plot the evolution of C with time and depth at the different parameters value. As we see, the system becomes stable after approximately 250 years. C concentration decreases with k_0, but is not impacted by p_N.

```
# Define colors
col <-   topo.colors
# Store default graphical parameters
par <- par()
# Increase vertical margins
par(mar = c(5.1, 4.1, 3.1, 2.1))
# Define color levels
levels <- seq(0, 200, length.out = 25)
# Draw the plots
for(i in 1:length(out)){
  title <- paste("C concentration", "\n", "k_0=", k_0_vec[i], "p_N=", p_N_vec[i])
  filled.contour(x=times,y=Depth, Conc_C[[i]], color= col, ylim=c(100,0),
                 levels = levels,
                 xlab="Time, years", ylab= "Depth, cm",
                 main=title)
}
par(par)
```



Let's now observe the evolution of N concentrations.

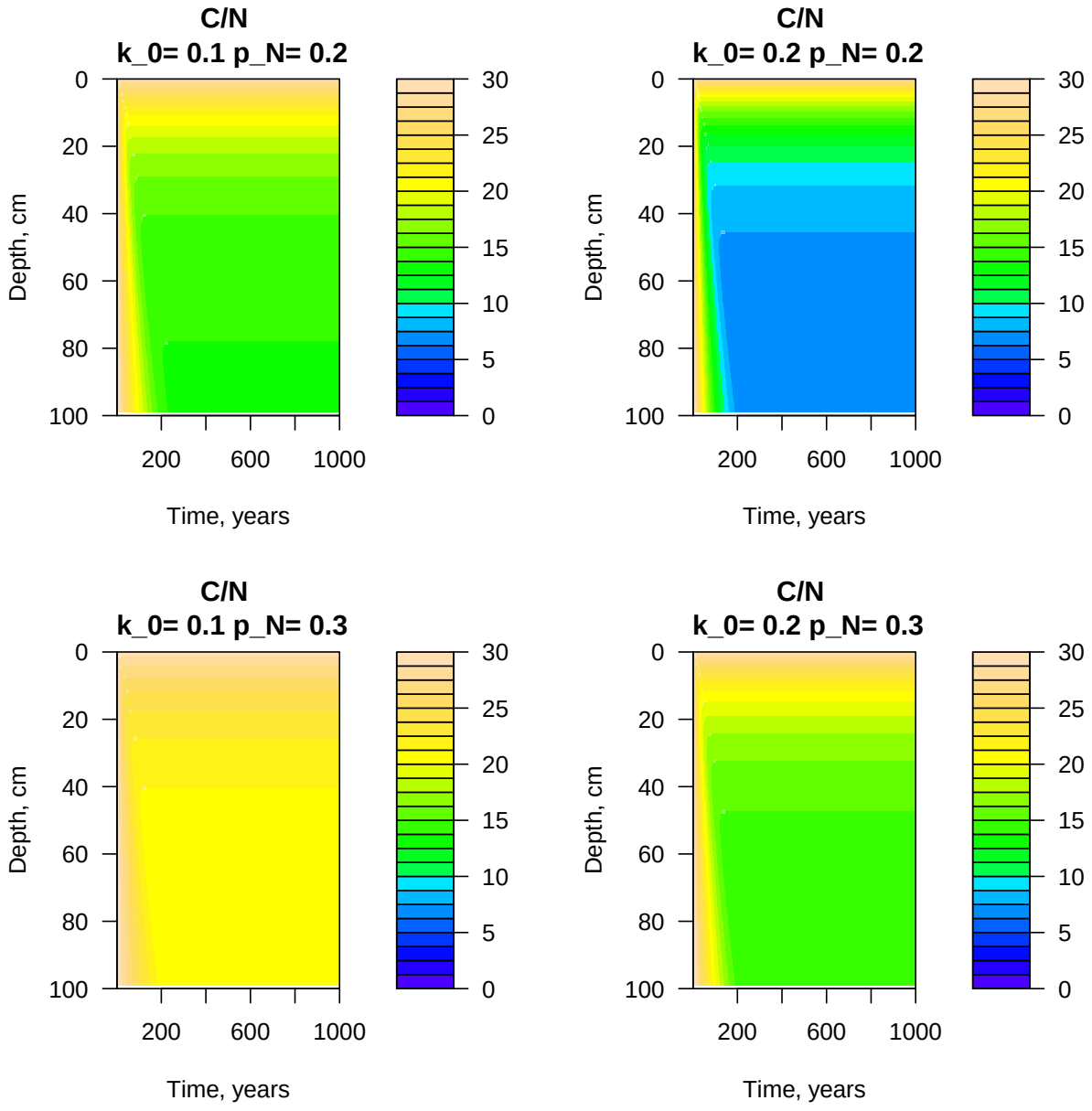
```
# Draw the plots
par(mar = c(5.1, 4.1, 3.1, 2.1))
# Define color levels
levels <- seq(0, 7, length.out = 15)
for(i in 1:length(out)){
  title <- paste("N concentration", "\n", "k_0=", k_0_vec[i], "p_N=", p_N_vec[i])
  filled.contour(x=times,y=Depth, Conc_N[[i]], color= col, ylim=c(100,0),
                 levels = levels,
                 xlab="Time, years", ylab= "Depth, cm",
                 main=title)
}
par(par)
```



And the C/N. We can see the distribution of C/N is highly sensitive to both parameters, the C/N being highest through all of the soil columns when k_0 is low and p_N is high.

```
# Draw the plots
par <- par()
par(mar = c(5.1, 4.1, 3.1, 2.1))
# Define color levels
levels <- seq(0, 30, length.out = 25)
for(i in 1:length(out)){
  title <- paste("C/N", "\n", "k_0=", k_0_vec[i], "p_N=", p_N_vec[i])
  filled.contour(x=times,y=Depth, CN[[i]], color= col, ylim=c(100,0),
                 levels = levels,
                 xlab="Time, years", ylab= "Depth, cm",
                 main=title)
```

```
}
par(par)
```



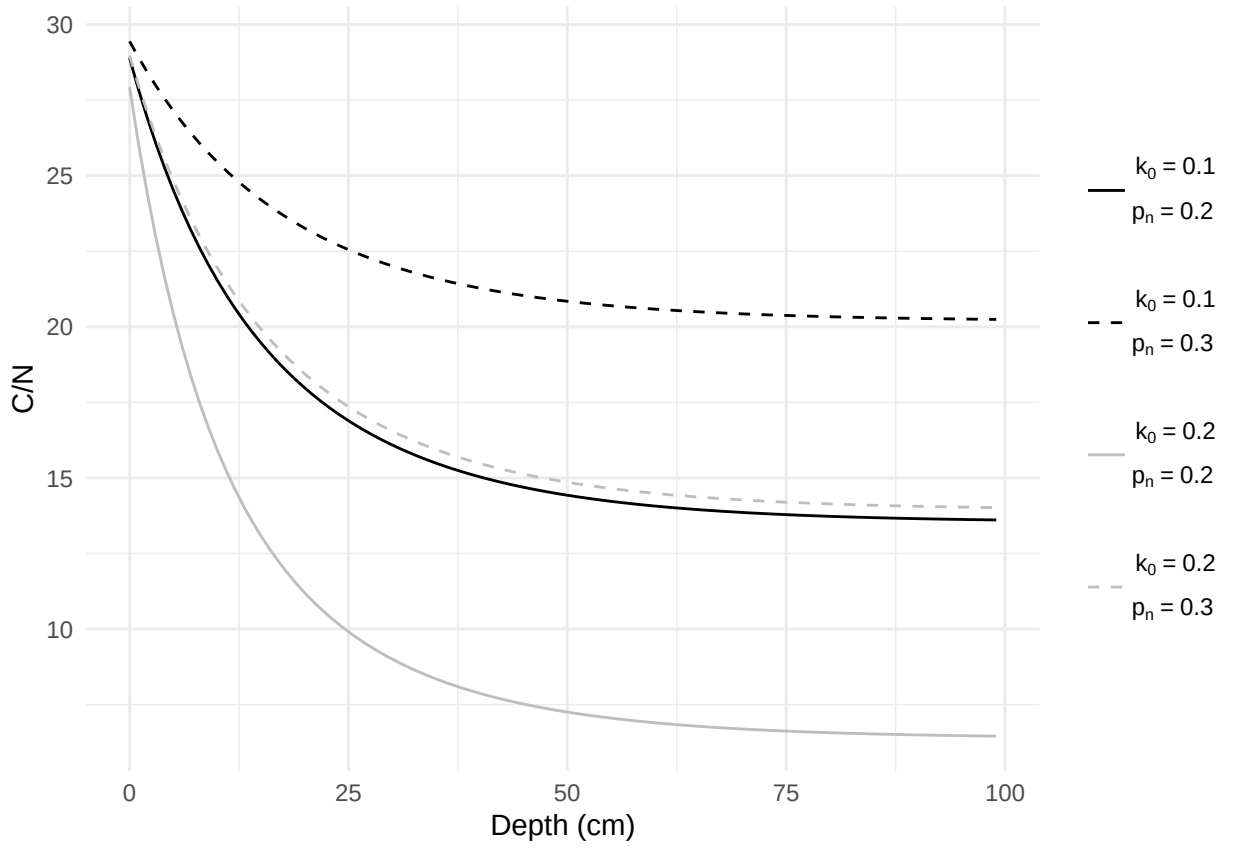
We will then gather the C/N profiles at the last timestep.

```
for(i in 1:length(CN)){
  Temp <- tibble(
    z = Depth,
    CN = CN[[i]][nrow(CN[[i]])-1,],
    k_0 = as.character(k_0_vec[i]),
    p_N = as.character(p_N_vec[i])
  ) %>%
    mutate(cat = paste("k_0=", k_0, "\npn=",p_N))
```

```
  if(i == 1) D_z_CN <- Temp
  else D_z_CN <- bind_rows(D_z_CN, Temp)
}
```

And plot the relationship between C/N and depth for the different values of k_0 and p_N.

```
(p_z_CN <- D_z_CN %>% ggplot(aes(y = CN, x = z)) +
  geom_line(aes(color = cat, linetype = cat)) +
  labs(y = "C/N", x = "Depth (cm)") +
  theme_minimal() +
  theme(legend.title = element_blank(),
        legend.text = element_text(margin = margin(t = 0.4, b = 0.4, unit = "cm"))) +
  scale_color_manual(values = c("black", "black", "grey", "grey"),
                     labels = expression(
                       atop(~k[0]~"="~ 0.1,p[n]~"="~0.2),
                       atop(~k[0]~"="~ 0.1,p[n]~"="~0.3),
                       atop(~k[0]~"="~ 0.2,p[n]~"="~0.2),
                       atop(~k[0]~"="~ 0.2,p[n]~"="~0.3))) +
  scale_linetype_manual(values = c(1,2,1,2),
                        labels = expression(
                          atop(~k[0]~"="~ 0.1,p[n]~"="~0.2),
                          atop(~k[0]~"="~ 0.1,p[n]~"="~0.3),
                          atop(~k[0]~"="~ 0.2,p[n]~"="~0.2),
                          atop(~k[0]~"="~ 0.2,p[n]~"="~0.3))
                        )
)
```



Variation of ^{15}N with depth and C/N

First of all, let's create function to convert between isotopic ratio to delta value and reverse. We will in the same way set the ratio of the standards (delta values of zero).

```
# Function for isotopic computations ------------------------------------
dtoR <- function(delta, standard) (1000+delta) * standard / 1000
Rtod <- function(R, standard) ((R - standard) / standard ) * 1000

Belemnita <- 0.0112372
AIR <- 0.003676
```

As suggested in the paper, we can model the variation of ^{15}N with depth by adding a fractionation constant (α) to the general model :

$$\frac{\delta^{13}C}{\delta t} = -v \cdot \frac{\delta^{13}C}{\delta z} - p_C \cdot k \cdot \alpha_C \cdot^{13} C \tag{7}$$

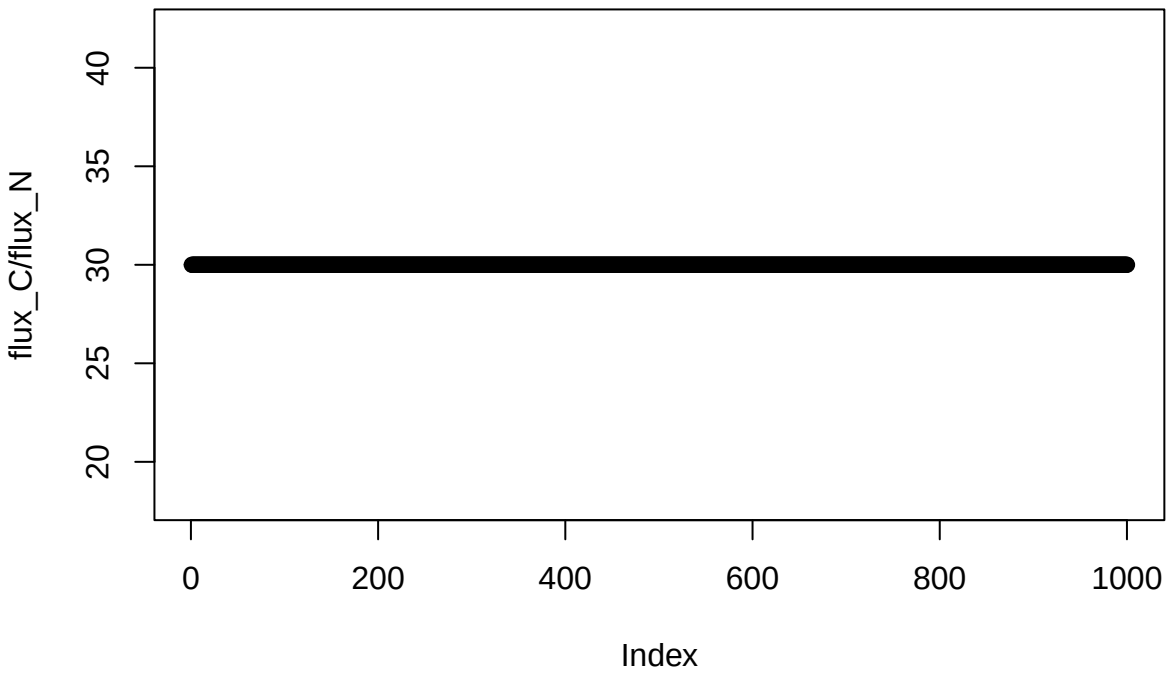
$$\frac{\delta^{15}N}{\delta t} = -v \cdot \frac{\delta^{15}N}{\delta z} - p_N \cdot k \cdot \alpha_N \cdot^{15} N \tag{8}$$

It is straightforward to modify the `deSolve` model above to include isotopes.

```
# Model equation
model_iso <-function(t, State, Pars)
{
  with(as.list(c(State, Pars)), {
    # Extract states and k
    C <- State[1:numboxes]
    N <- State[(numboxes+1):(2*numboxes)]
    C13 <- State[(2*numboxes+1):(3*numboxes)]
    N15 <- State[(3*numboxes+1):(4*numboxes)]
    k <- pars[2:(2+99)]
    # Compute vertical fluxes
    Flux_C   <- c(flux_C_fun(t), v*C)
    Flux_N <- c(flux_N_fun(t), v*N)
    Flux_C13 <- c(flux_C13_fun(t), v*C13)
    Flux_N15 <- c(flux_N15_fun(t), v*N15)
    # Compute rate of change
    dC      <- -diff(Flux_C)/delx  - k*p_C*C
    dN      <- -diff(Flux_N)/delx  - k*p_N*N
    dC13      <- -diff(Flux_C13)/delx  - a_C*k*p_C*C13
    dN15      <- -diff(Flux_N15)/delx  - a_N*k*p_N*N15
    return(list(c(dC, dN, dC13, dN15)))     # result
  })
}  # end of model
```

We then need to define the isotopic ratio of inputs. We will consider a constant flux of plants material with $\delta^{13}C = -27‰$ and $\delta^{15}N = 0‰$.

```
# Vector of timesteps
times <- 1:1000
# Vector and function of C input
flux_C <- rep(100, times = length(times))
flux_C_fun <- approxfun(x = times, y = flux_C)
# Vector and function of N input
flux_N <- flux_C/30
flux_N_fun <- approxfun(x = times, y = flux_N)
plot(flux_C/flux_N)
```




```
# Vector and function of 13C input
flux_C13 <- flux_C*dtoR(-27, Belemnita)
flux_C13_fun <- approxfun(x = times, y = flux_C13)
# Vector and function of 15N input
flux_N15 <- flux_N*dtoR(0, AIR)
flux_N15_fun <- approxfun(x = times, y = flux_N15)
```

We have to slightly modify the initial states to include the two new variables.

```
state <- c(rep(0, times=numboxes), # Initial C concentration
           rep(0,times=numboxes), # Initial N concentration
           rep(0,times=numboxes), # Initial 13C concentration
           rep(0,times=numboxes)) # Initial 15N Concentration
```

We will define the parameters with two correlated values of p_N and α_n.

```
# Define the values of k_0 and p_N to test
p_N_vec <- c(0.2, 0.3)
a_N_vec <- c(0.999, 0.998)
# Create a list to store parameters
Params <- list()

# Create a loop to store parameters values
for(i in 1:length(a_N_vec)){
  Params[[i]] <- c(
```

```
  v = 0.5,
  k = k_func(Depth, 0.2, 20),
  p_C = 0.4,
  p_N = p_N_vec[i],
  a_C = 1, # No fractionation for C isotope
  a_N = a_N_vec[i]
)
}
```

And estimate the solution. We will use a loop to go through the different values of parameters.

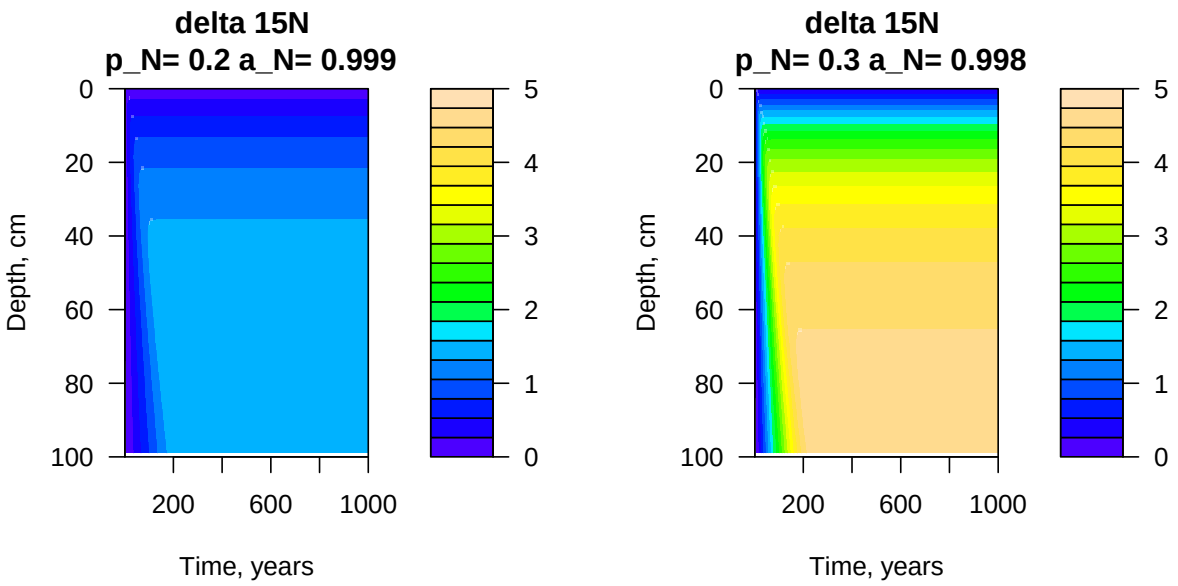
```
# Create a list to store simulation results
out <- list()
for(i in 1:length(Params)){
  pars <- Params[[i]]
  out[[i]] <- ode.1D(y = state, times = times, func = model_iso, parms = pars,
             dimens = numboxes)
}
```

We can then extract the concentrations from the big matrix output, and compute C/N and $\delta^{15}N$.

```
# Create lists to store results
Conc_C <- list()
Conc_N <- list()
Conc_C13 <- list()
Conc_N15 <- list()
CN <- list()
delta_N15 <- list()

for(i in 1:length(out)){
  Conc_C[[i]] <- out[[i]][,2:(numboxes+1)]
  Conc_N[[i]] <- out[[i]][,((numboxes+2)):(2*numboxes+1)]
  Conc_C13[[i]] <- out[[i]][,((2*numboxes+2)):(3*numboxes+1)]
  Conc_N15[[i]] <- out[[i]][,((3*numboxes+2)):(4*numboxes+1)]
  CN[[i]] <- Conc_C[[i]]/Conc_N[[i]]
  delta_N15[[i]] <- Rtod(Conc_N15[[i]]/Conc_N[[i]], AIR)
}
```

We can observe the variation of $\delta^{15}N$ with depth and time.

```
# Draw the plots
par(mar = c(5.1, 4.1, 3.1, 2.1))
# Define color levels
levels <- seq(0, 5, length.out = 20)
for(i in 1:length(out)){
  title <- paste("delta 15N", "\n", "p_N=", p_N_vec[i], "a_N=", a_N_vec[i])
  filled.contour(x=times,y=Depth, delta_N15[[i]], color= col, ylim=c(100,0),
                 levels = levels,
                 xlab="Time, years", ylab= "Depth, cm",
                 main=title)
}
par(par)
```



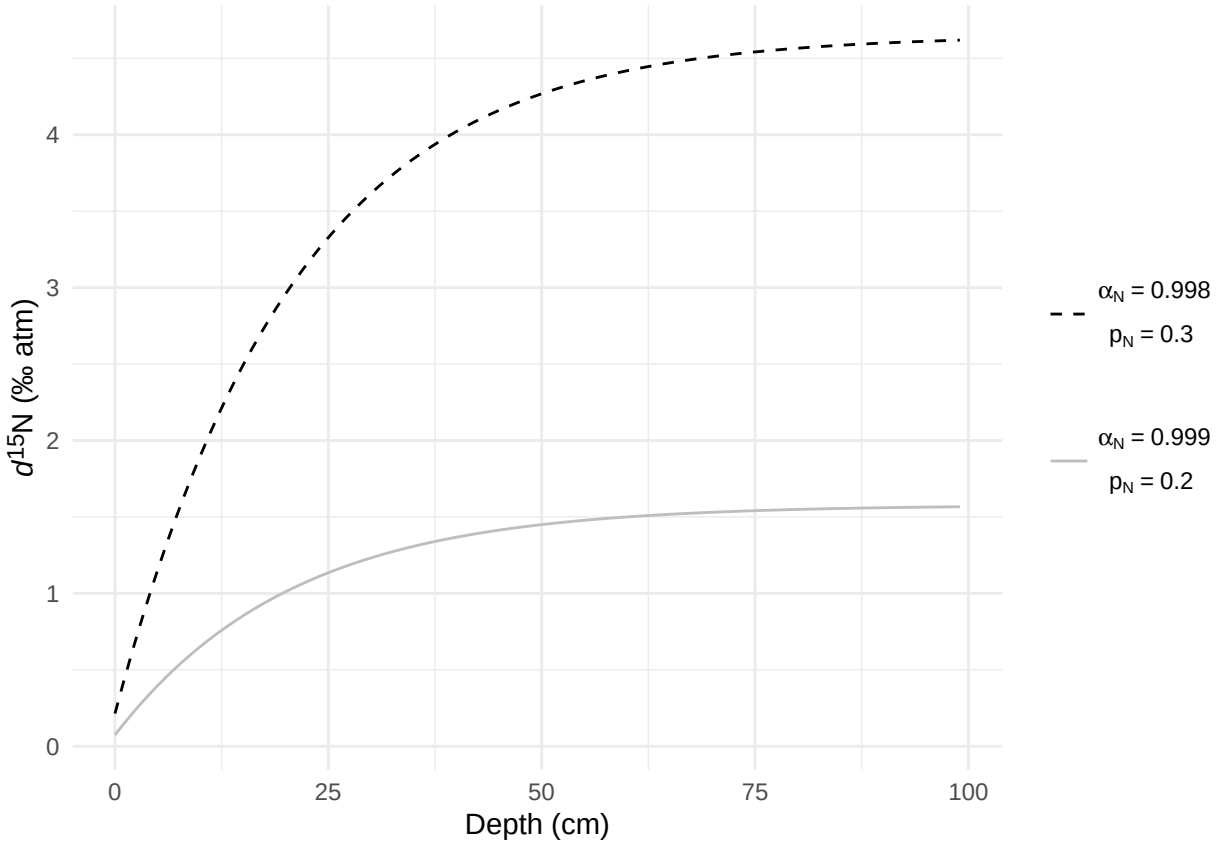
We will then gather the C/N and ^{15}N profiles at the last time step.

```
for(i in 1:length(delta_N15)){
  Temp <- tibble(
    z = Depth,
    CN = CN[[i]][nrow(CN[[i]])-1,],
    d15N = delta_N15[[i]][nrow(delta_N15[[i]])-1,],
    p_N = as.character(p_N_vec[i]),
    a_N = as.character(a_N_vec[i])
  ) %>%
    mutate(cat = paste("a_N=", a_N, "\npn=",p_N))

  if(i == 1) D_15N <- Temp
  else D_15N <- bind_rows(D_15N, Temp)
}
```

And then plot the result as a function of depth.

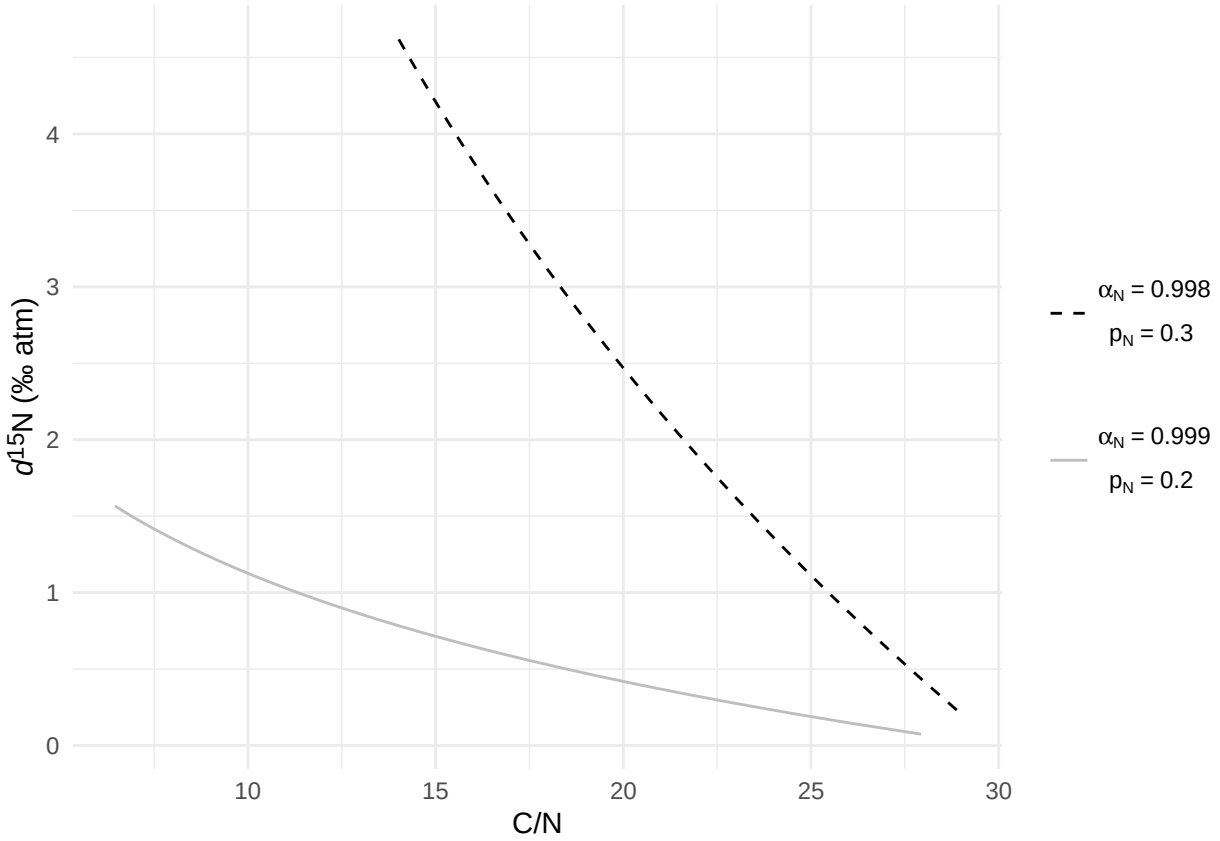
```
(p_d_15N <- D_15N %>% ggplot(aes(x = z, y = d15N)) +
  geom_line(aes(color = cat, linetype = cat)) +
  # geom_jitter(aes(color = cat), width = 0.1)+
  theme_minimal() +
    theme(legend.text = element_text(margin = margin(t = 0.5, b = 0.5, unit = "cm"))) +
    scale_color_manual(values = c("black", "grey"),
                       labels = expression(
                         atop(~alpha[N]~"="~ 0.998,p[N]~"="~0.3),
                         atop(~alpha[N]~"="~ 0.999,p[N]~"="~0.2))) +
    scale_linetype_manual(values = c(2,1,2,1),
                          labels = expression(
                            atop(~alpha[N]~"="~ 0.998,p[N]~"="~0.3),
                            atop(~alpha[N]~"="~ 0.999,p[N]~"="~0.2))
    ) +
  labs(x = "Depth (cm)",
       y = "*&delta;*<sup>15</sup>N (&permil; atm)") +
```

```
    theme(legend.title = element_blank(),
          axis.title.y = element_markdown())
)
```



And plot it again as a function of C/N.

```
(p_CN_15N <- D_15N %>% ggplot(aes(x = CN, y = d15N)) +
  geom_line(aes(color = cat, linetype = cat)) +
  # geom_jitter(aes(color = cat), height = 0.1)+
  theme_minimal() +
    theme(legend.text = element_text(margin = margin(t = 0.5, b = 0.5, unit = "cm"))) +
    scale_color_manual(values = c("black", "grey", "black", "grey"),
                       labels = expression(
                         atop(~alpha[N]~"="~ 0.998,p[N]~"="~0.3),
                         atop(~alpha[N]~"="~ 0.999,p[N]~"="~0.2))) +
    scale_linetype_manual(values = c(2,1),
                          labels = expression(
                            atop(~alpha[N]~"="~ 0.998,p[N]~"="~0.3),
                            atop(~alpha[N]~"="~ 0.999,p[N]~"="~0.2))
    ) +
  labs(x = "C/N",
       y = "*&delta;*<sup>15</sup>N (&permil; atm)") +
  theme(legend.title = element_blank(),
        axis.title.y = element_markdown())
)
```



Variation of ^{13}C with C/N

Constant isotopic ratio of the input

We can reuse the same model to simulate the variation of $\delta^{13}C$ with C/N and depth, considering a constant isotopic value of plant input (-27‰). We will define a set of three different fractionation factors.

```
# Define the values of a_C to test
a_C_vec <- c(0.999, 1, 1.001)
# Create a list to store parameters
Params <- list()

# Create a loop to store parameters values
for(i in 1:length(a_C_vec)){
  Params[[i]] <- c(
  v = 0.5,
  k = k_func(Depth, 0.2, 20),
  p_C = 0.4,
  p_N = 0.3,
  a_C = a_C_vec[i], # No fractionation for C isotope
  a_N = 1
)
}
```

And solve the model.

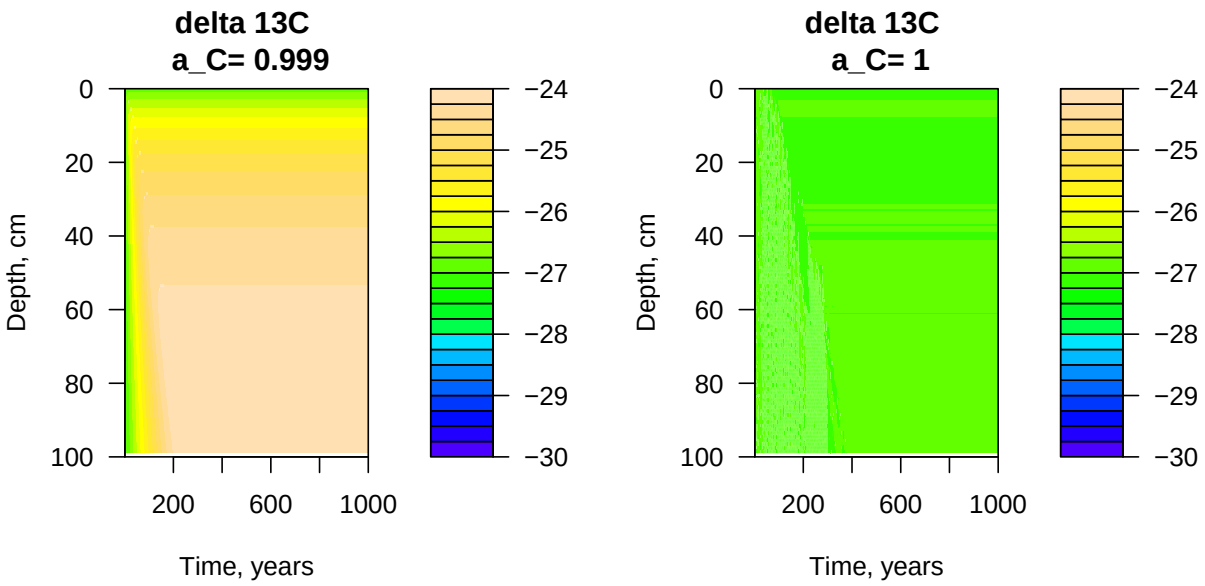
```
# Create a list to store simulation results
out <- list()
for(i in 1:length(Params)){
  pars <- Params[[i]]
  out[[i]] <- ode.1D(y = state, times = times, func = model_iso, parms = pars,
             dimens = numboxes)
}
```

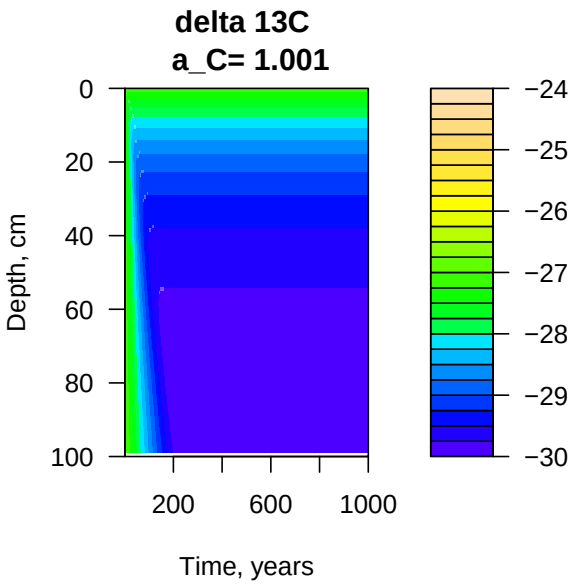
We can then extract the concentrations from the big matrix output, and compute C/N and $\delta^{13}C$.

```
# Create lists to store results
Conc_C <- list()
Conc_N <- list()
Conc_C13 <- list()
Conc_N15 <- list()
CN <- list()
delta_C13 <- list()

for(i in 1:length(out)){
  Conc_C[[i]] <- out[[i]][,2:(numboxes+1)]
  Conc_N[[i]] <- out[[i]][,((numboxes+2)):(2*numboxes+1)]
  Conc_C13[[i]] <- out[[i]][,((2*numboxes+2)):(3*numboxes+1)]
  Conc_N15[[i]] <- out[[i]][,((3*numboxes+2)):(4*numboxes+1)]
  CN[[i]] <- Conc_C[[i]]/Conc_N[[i]]
  delta_C13[[i]] <- Rtod(Conc_C13[[i]]/Conc_C[[i]], Belemnita)
}
```

We can observe the variation of $\delta^{13}C$ with depth and time.

```
# Draw the plots
par(mar = c(5.1, 4.1, 3.1, 2.1))
# Define color levels
levels <- seq(-30, -24, length.out = 25)
for(i in 1:length(out)){
  title <- paste("delta 13C", "\n", "a_C=", a_C_vec[i])
  filled.contour(x=times,y=Depth, delta_C13[[i]], color= col, ylim=c(100,0),
                 levels = levels,
                 xlab="Time, years", ylab= "Depth, cm",
                 main=title)
}
par(par)
```



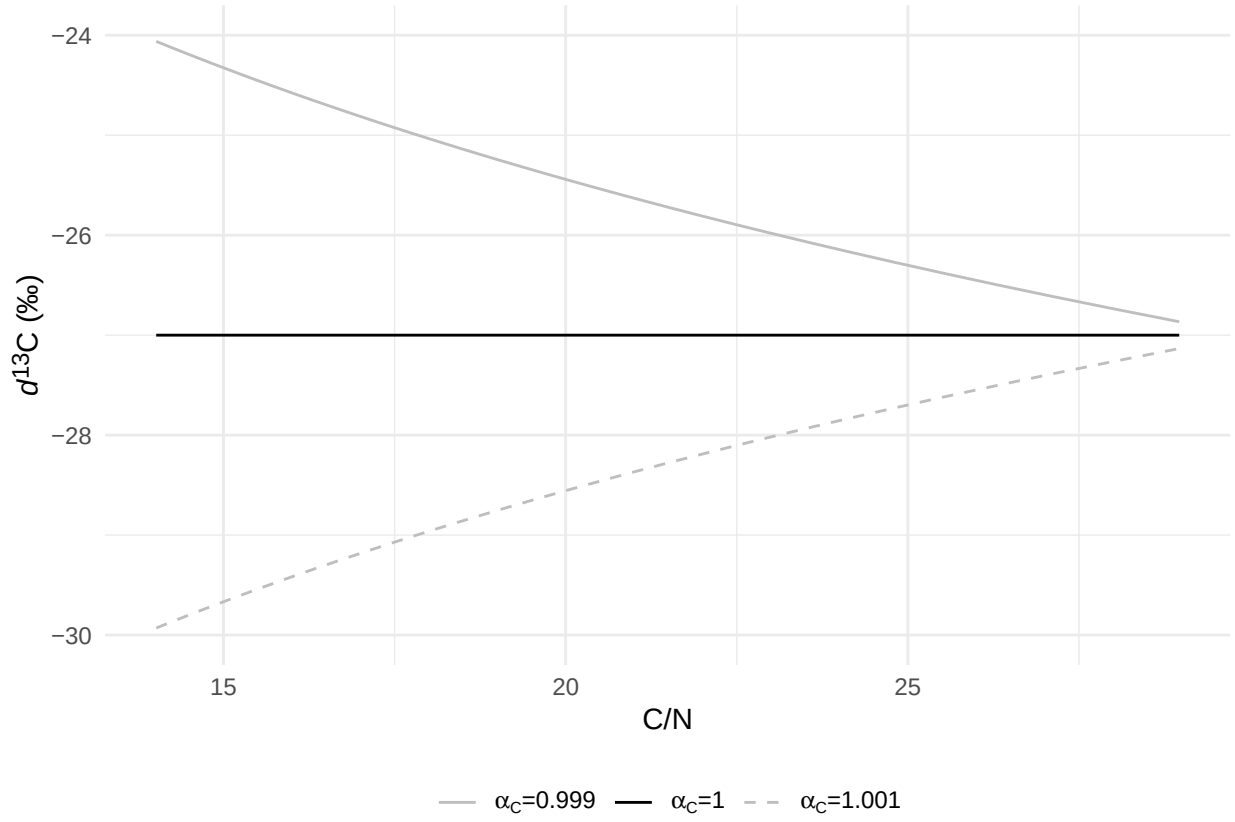


We will then gather the C/N and ^{13}C profiles at the last time step.

```
for(i in 1:length(delta_C13)){
  Temp <- tibble(
    z = Depth,
    CN = CN[[i]][nrow(CN[[i]])-1,],
    d13C = delta_C13[[i]][nrow(delta_C13[[i]])-1,],
    a_C = as.character(a_C_vec[i])
  ) %>%
    mutate(cat = paste("a_C=", a_C))

  if(i == 1) D_13C <- Temp
  else D_13C <- bind_rows(D_13C, Temp)
}
```

And plot the results.

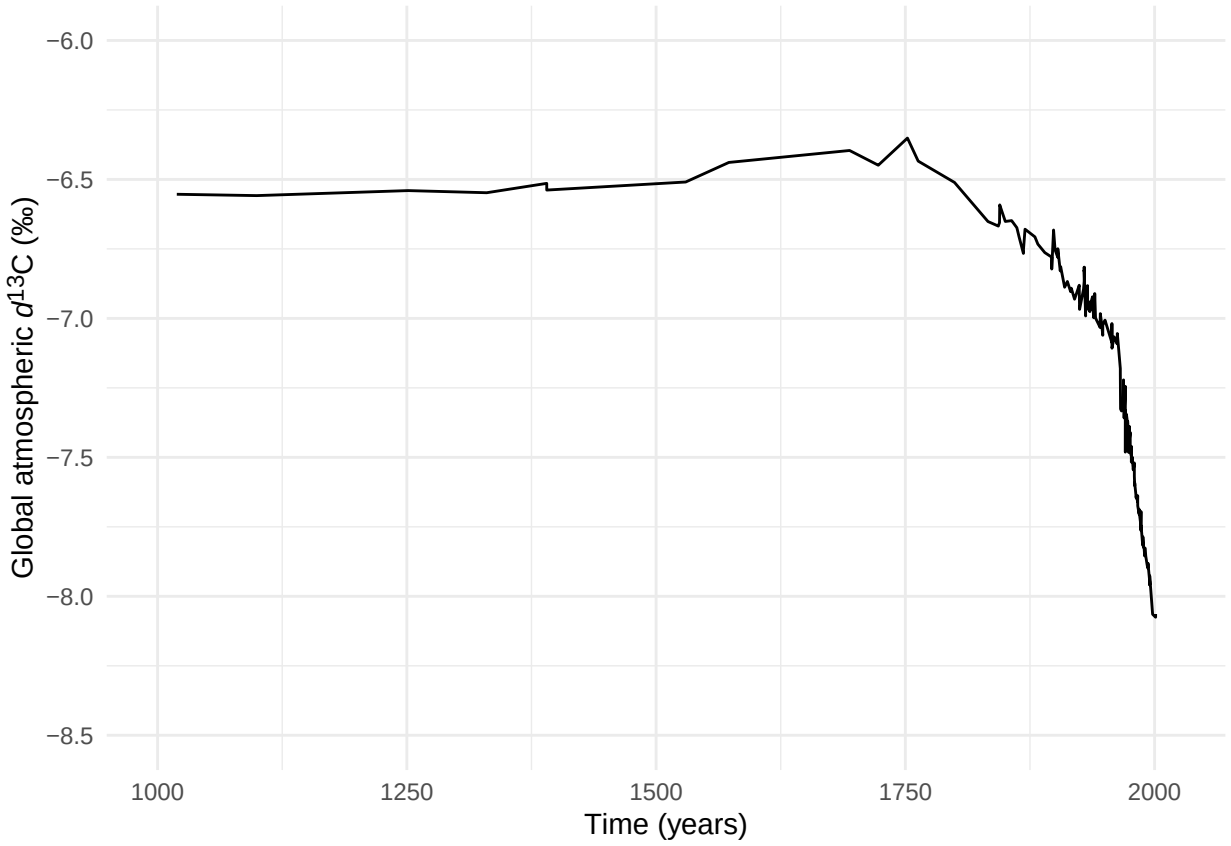
```
(p_d13C_CN_eq <- D_13C %>%
    ggplot(aes(x = CN, y = d13C)) +
    geom_line(aes(color = cat, linetype = cat)) +
    theme_minimal() +
    theme(legend.position = "bottom",
          legend.title = element_blank(),
          axis.title.y = element_markdown()) +
    scale_color_manual(values = c("grey", "black", "grey"),
                       labels = expression(
                         paste(alpha[C], "=", 0.999),
                         paste(alpha[C], "=", 1),
                         paste(alpha[C], "=", "1.001"))) +
    scale_linetype_manual(values = c(1,1,2),
                          labels = expression(
                            paste(alpha[C], "=", 0.999),
                            paste(alpha[C], "=", 1),
                            paste(alpha[C], "=", "1.001"))
    ) +
    labs(x = "C/N",
         y = "*&delta;*<sup>13</sup>C (&permil;)") +
    ylim(-30, -24)
)
```



Variable isotopic ratio of the input

δ^13^C of the atmosphere decreased since the beginning of the industrial era, because of fossil fuel emissions. This decrease can be visualize using the dataset made available by *Rubino et al.* (2013).

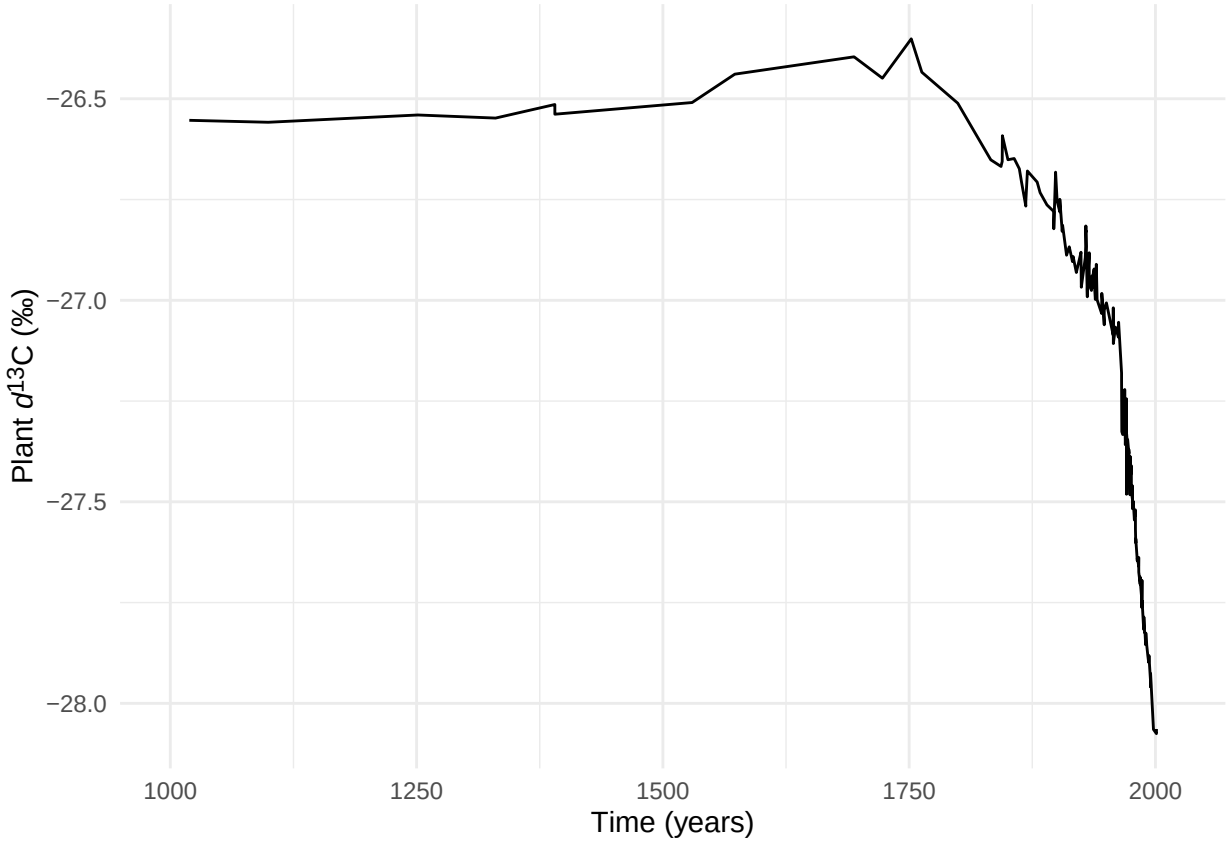
```
# Plot the old atmosphere ------------------------------------------------
TS_old <- read_xls("C:/Users/deschaml/OneDrive/Projets/Deschamps_2022_Isotopes/Data/Rubino_Fancey_2013_

(p_atm <- TS_old %>%
    ggplot(aes(x = `effective age [AD] for CO2`,
               y = `d13C-CO2 corrected for blank, gravity and diffusion, vPDB [%]`)) +
    geom_line() +
    theme_minimal() +
  labs(x = "Time (years)",
       y = "Global atmospheric *&delta;*<sup>13</sup>C (&permil;)")  +
  lims(x = c(1000, 2020),
       y = c(-8.5, -6)) +
  theme(axis.title.y = element_markdown())
  )
```



We will define the δ^13^C of plant input roughly, by substracting 20 to the atmospheric signature. As before, we will define a $_{12}$C flux of $100g \cdot year^{-1}$.

```
TS <- TS_old %>%
  ## Create a column for plant signature
  mutate(Plant_d13C = `d13C-CO2 corrected for blank, gravity and diffusion, vPDB [%]` - 20,
```

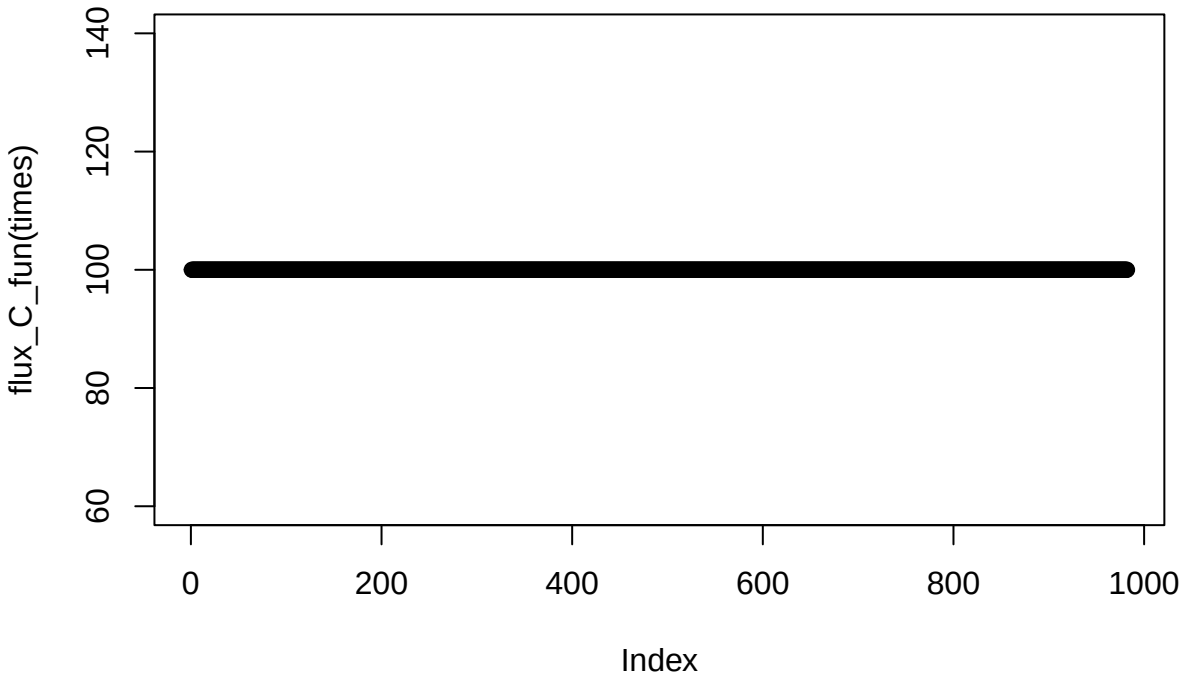
```
         Plant_R13C = dtoR(Plant_d13C, Belemnita),
         Plant_C = 100,
         Plant_13C = Plant_C * Plant_R13C,
         Year = `effective age [AD] for CO2`) %>%
  filter(complete.cases(Year))

TS %>%
    ggplot(aes(x = Year,
               y = Plant_d13C)) +
    geom_line() +
    theme_minimal() +
  labs(x = "Time (years)",
       y = "Plant *&delta;*<sup>13</sup>C (&permil;)")  +
  lims(x = c(1000, 2020)) +
  theme(axis.title.y = element_markdown())
```



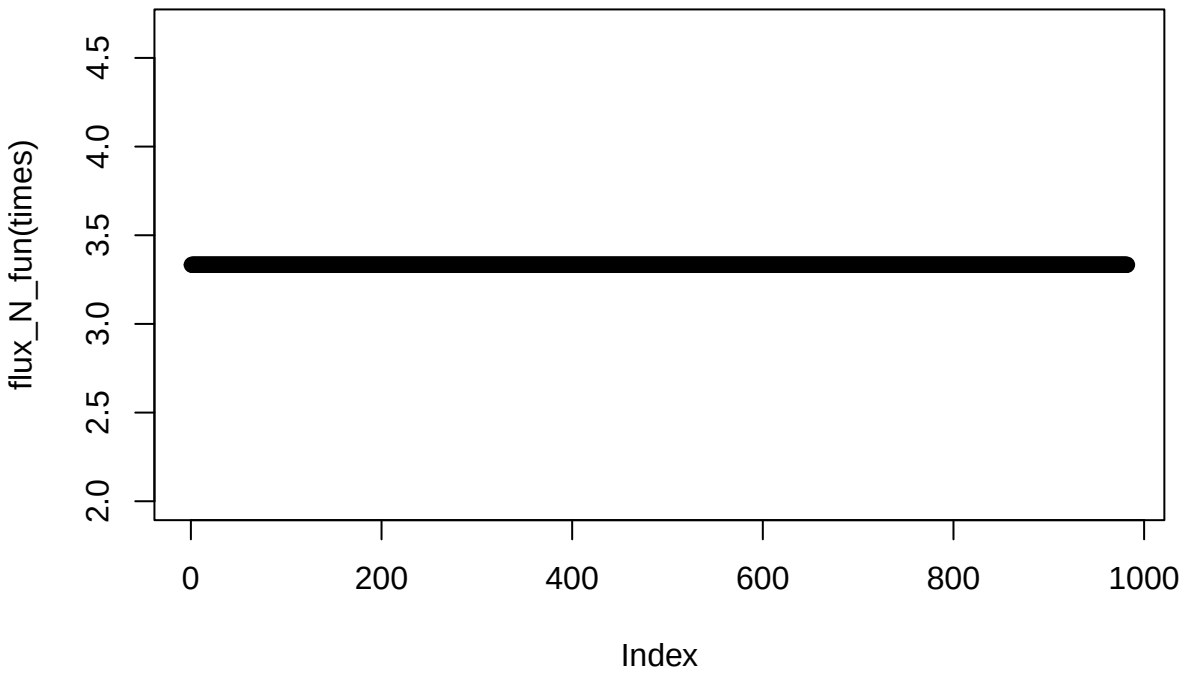
To accomodate such changes in input flux (*boundary condition*), we will reuse the same model but changing the input to a function reproducing the evolution of atmospheric (and plant) baseline $\delta^{13}C$.

Then, we extract the boundary conditions from the `TS` dataset, and approximate them using a function.

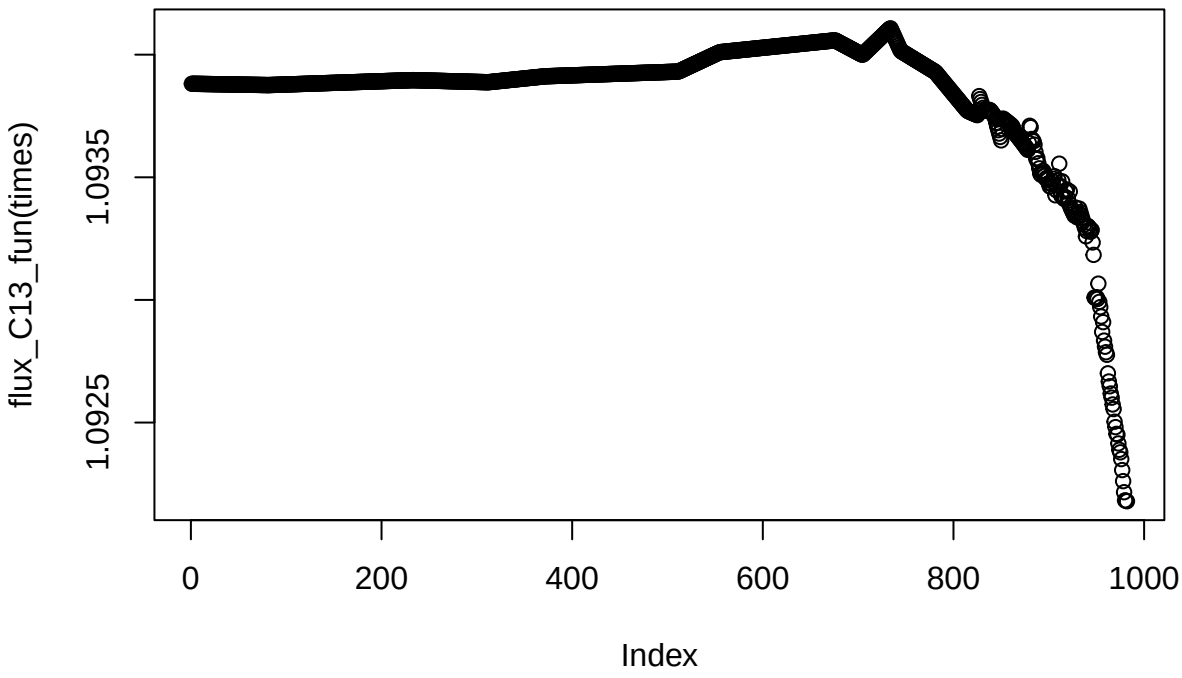
```
# Vector of timesteps
times <- min(TS$Year):max(TS$Year)   # output wanted at these time intervals`
# Vector and function of C input
flux_C <- TS$Plant_C
flux_C_fun <- approxfun(x = TS$Year, y = flux_C)
plot(flux_C_fun(times))
```



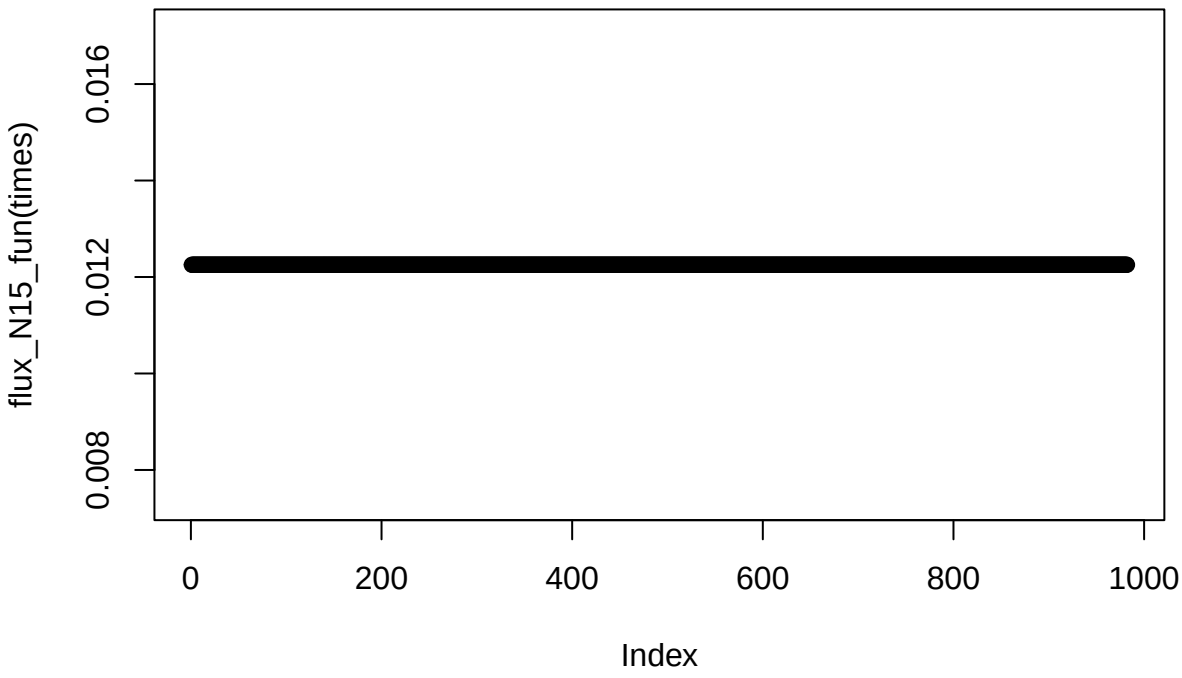

```
# Vector and function of N input
flux_N <- flux_C/30
flux_N_fun <- approxfun(x = TS$Year, y = flux_N)
plot(flux_N_fun(times))
```




```
# Vector and function of 13C input
flux_C13 <- TS$Plant_13C
flux_C13_fun <- approxfun(x = TS$Year, y = flux_C13)
plot(flux_C13_fun(times))
```




```
# Vector and function of 15N input
flux_N15 <- flux_N*dtoR(0, AIR)
flux_N15_fun <- approxfun(x = TS$Year, y = flux_N15)
plot(flux_N15_fun(times))
```



And solve the model.

```
# Create a list to store simulation results
out <- list()
for(i in 1:length(Params)){
  pars <- Params[[i]]
  out[[i]] <- ode.1D(y = state, times = times, func = model_iso, parms = pars,
             dimens = numboxes)
}
```

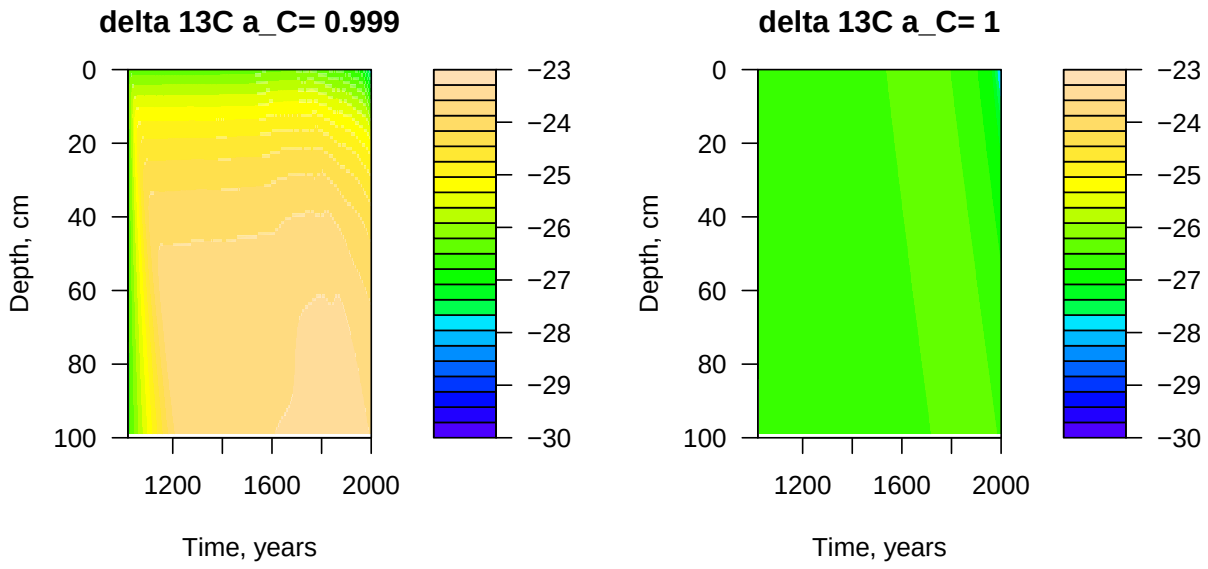
We can then extract the concentrations from the big matrix output, and compute C/N and $\delta^{13}C$.

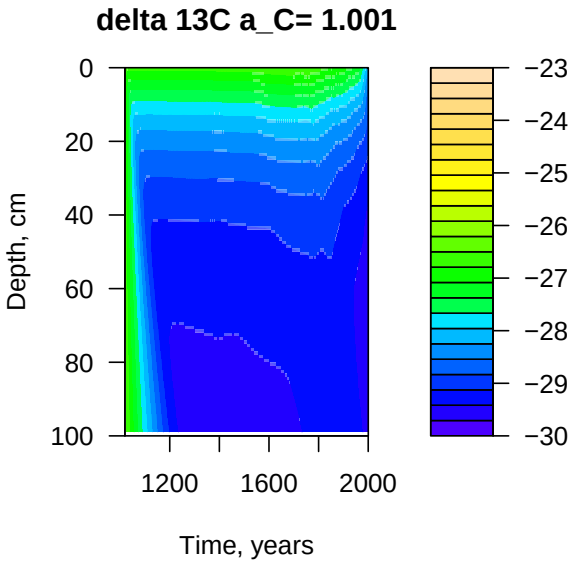
```
# Create lists to store results
Conc_C <- list()
Conc_N <- list()
Conc_C13 <- list()
Conc_N15 <- list()
CN <- list()
delta_C13 <- list()

for(i in 1:length(out)){
  Conc_C[[i]] <- out[[i]][,2:(numboxes+1)]
  Conc_N[[i]] <- out[[i]][,((numboxes+2)):(2*numboxes+1)]
  Conc_C13[[i]] <- out[[i]][,((2*numboxes+2)):(3*numboxes+1)]
  Conc_N15[[i]] <- out[[i]][,((3*numboxes+2)):(4*numboxes+1)]
  CN[[i]] <- Conc_C[[i]]/Conc_N[[i]]
```

```
  delta_C13[[i]] <- Rtod(Conc_C13[[i]]/Conc_C[[i]], Belemnita)
}
```

We can observe the variation of $\delta^{13}C$ with depth and time. We can see that dpletion of atmospheric input create an enrichment in 13C with depth, older, more decomposed OM coming for ^{13}C enriched plant material..

```
# Draw the plots
par(mar = c(5.1, 4.1, 3.1, 2.1))
# Define color levels
levels <- seq(-30, -23, length.out = 25)
for(i in 1:length(out)){
  title <- paste("delta 13C", "a_C=", a_C_vec[i])
  filled.contour(x=times,y=Depth, delta_C13[[i]], color= col, ylim=c(100,0),
                 levels = levels,
                 xlab="Time, years", ylab= "Depth, cm",
                 main=title)
}
par(par)
```





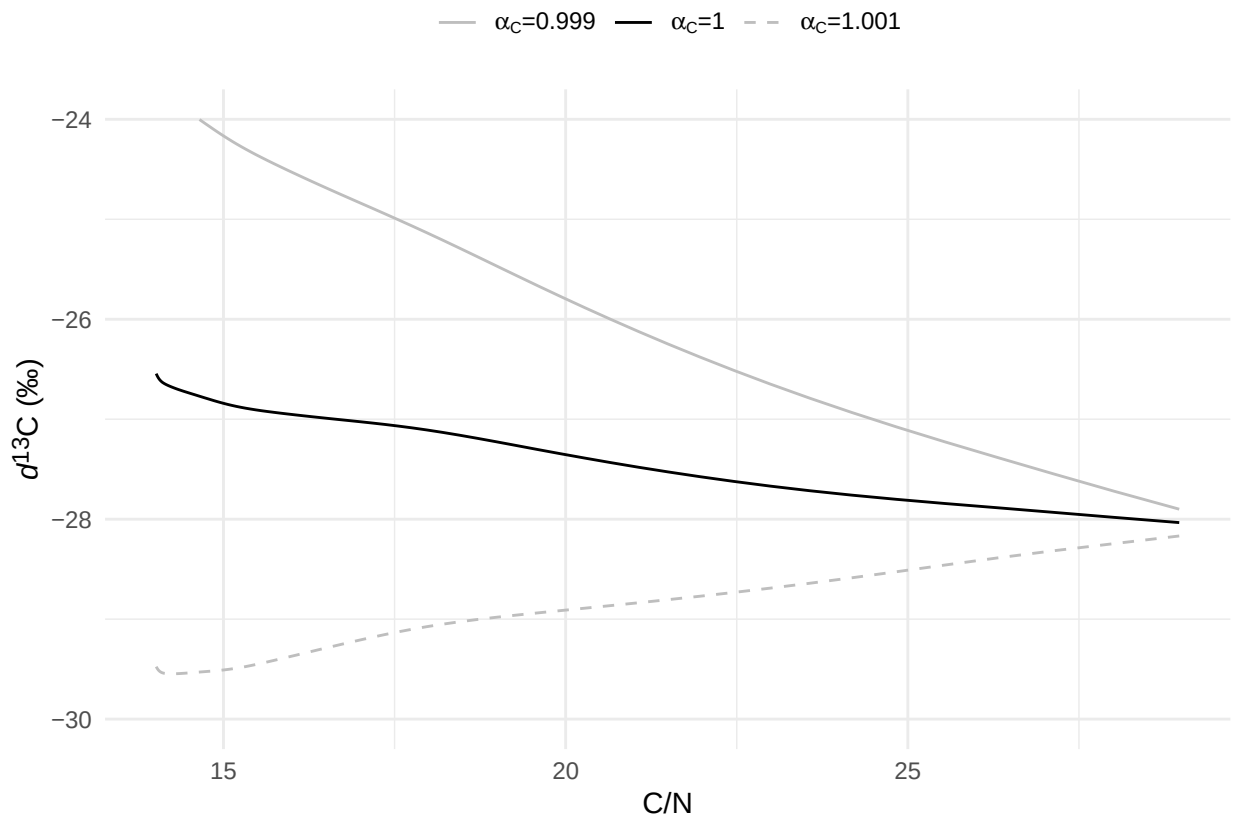
We will then gather the C/N and ^{13}C profiles at the last time step.

```
for(i in 1:length(delta_C13)){
  Temp <- tibble(
    z = Depth,
    CN = CN[[i]][nrow(CN[[i]])-1,],
    d13C = delta_C13[[i]][nrow(delta_C13[[i]])-1,],
    a_C = as.character(a_C_vec[i])
  ) %>%
    mutate(cat = paste("a_C=", a_C))

  if(i == 1) D_13C_atm <- Temp
  else D_13C_atm <- bind_rows(D_13C_atm, Temp)
}
```

And plot the results. Even without fractionation, depletion of plant inputs lead to a negative relationship between C/N and $\delta^{13}C$, meaning that Om becomes more concentred in ^{13}C while decomposition advances and C/N decrease.

```
(p_d13C_CN_atm <- D_13C_atm %>%
    ggplot(aes(x = CN, y = d13C)) +
    geom_line(aes(color = cat, linetype = cat)) +
    theme_minimal() +
    theme(legend.position = "top",
          legend.title = element_blank(),
          axis.title.y = element_markdown()) +
    scale_color_manual(values = c("grey", "black", "grey"),
                       labels = expression(
                         paste(alpha[C], "=", 0.999),
                         paste(alpha[C], "=", 1),
                         paste(alpha[C], "=", "1.001"))) +
    scale_linetype_manual(values = c(1,1,2),
                          labels = expression(
```

```
                            paste(alpha[C], "=", 0.999),
                            paste(alpha[C], "=", 1),
                            paste(alpha[C], "=", "1.001"))
    ) +
    labs(x = "C/N",
         y = "*&delta;*<sup>13</sup>C (&permil;)") +
    ylim(-30, -24)
)
```



Final ^{13}C graph

```
# Final plot
library(cowplot)
my_legend <- get_legend(p_d13C_CN_atm)
(p_13C <- plot_grid(my_legend,
                    p_atm,
                    plot_grid(p_d13C_CN_eq + theme(legend.position="none"),
                              p_d13C_CN_atm + ylab(NULL) + theme(legend.position = "none"),
                              ncol = 2, labels = c("b", "c"),
                              hjust = -5, vjust = 1),
          ncol = 1, labels = c("", "a", ""), align = "h", rel_heights = c(.1, 1, 1),
          hjust = -5, vjust = 1))
# Save the plot
```

```
# ggsave("C:/Users/gosse/OneDrive - Université du Québec à Trois-Rivières/Projects/Active_projects/Desc
#        width = 7.5, height = 5, bg = "white")
```

References

Rubino, M., Etheridge, D. M., Trudinger, C. M., Allison, C. E., Battle, M. O., Langenfelds, R. L., Steele, L. P., Curran, M., Bender, M., White, J. W. C., Jenk, T. M., Blunier, T., & Francey, R. J. (2013). A revised 1000 year atmospheric 13 C-CO2 record from Law Dome and South Pole, Antarctica. Journal of Geophysical Research Atmospheres, 118(15), 8482–8499. https://doi.org/10.1002/jgrd.50668

Soetaert, K., & Herman, P. M. J. (2009). A pratical guide to ecological modelling using R as a simulation platform. Springer. https://doi.org/https://doi.org/10.1111/j.1442-9993.2010.02180.x