The CNPDIA package - adding perturbations to the C, N and P cycle

Karline Soetaert

20-november-2017

```r
require(CNPDIA)
```

CNPDIAperturb

The CNPDIA model is a 1-D model of Carbon, nitrogen, phosphorus and oxygen diagenesis in a marine sediment. It is based on the OMEXDIA model (Soetaert et al., 1996a, b), extended with simple P dynamics.

The model describes twelve state variables, in 100 layers:

- 2 fractions of organic carbon (FDET,SDET): fast and slow decaying, solid substance.
- Oxygen (O2), dissolved substance.
- Nitrate (NO3), dissolved substance.
- Nitrite (NO2), dissolved substance.
- Ammonium (NH3), dissolved substance.
- Oxygen demand unit (ODU), dissolved substance, as a lump sum of all reduced substances other than ammonium.
- Dissolved inorganic carbon (DIC), dissolved substance
- Phosphate (PO4), dissolved substance
- Iron-bound P (FeP), P bound to iron oxides, solid substance
- Ca-bound P (CaP), apatite, solid substance
- Adsorbed P (Pads), solid substance

It contains functions to perturb sediment properties, e.g. mimicking resuspension, deposition or erosion events.

Main functions

The main functions allow to solve the model to steady state (*CNPDIAsolve*), to run it dynamically (*CNPDIAdyna*), or to add perturbations (*CNPDIAperturb*) to dynamic simulations.

Steady-state solution, function CNPDIAsolve

Function *CNPDIAsolve* finds the steady-state solution of the CNPDIA model. Its arguments are:

```r
args(CNPDIAsolve)
```

```
function (parms = list(), yini = NULL, gridtype = 1, Grid = NULL,
    porosity = NULL, bioturbation = NULL, irrigation = NULL,
    surface = NULL, diffusionfactor = NULL, dynamicbottomwater = FALSE,
    ratefactor = NULL, verbose = FALSE, ...)
NULL
```

here *parms* is a list with a subset of the CNPDIA parameters (see main vignette for what they mean and their default values). If unspecified, then the default parameters are used.

Dynamic run, function CNPDIAdyna

Function *CNPDIAdyna* runs the model dynamically without perturbations. Its arguments are:

```
args(CNPDIAdyna)
```

```
function (parms = list(), times = 0:365, spinup = NULL, yini = NULL,
    gridtype = 1, Grid = NULL, porosity = NULL, bioturbation = NULL,
    irrigation = NULL, surface = NULL, diffusionfactor = NULL,
    dynamicbottomwater = FALSE, CfluxForc = NULL, FePfluxForc = NULL,
    CaPfluxForc = NULL, O2bwForc = NULL, NO3bwForc = NULL, NO2bwForc = NULL,
    NH3bwForc = NULL, ODUbwForc = NULL, PO4bwForc = NULL, DICbwForc = NULL,
    wForc = NULL, biotForc = NULL, irrForc = NULL, rFastForc = NULL,
    rSlowForc = NULL, pFastForc = NULL, MPBprodForc = NULL, gasfluxForc = NULL,
    HwaterForc = NULL, ratefactor = NULL, verbose = FALSE, ...)
NULL
```

Perturbation run, function CNPDIAperturb

Function *CNPDIAperturb* runs the CNPDIA model for a specific time interval while adding perturbations at requested times.

Its arguments are:

```
args(CNPDIAperturb)
```

```
function (parms = list(), times = 0:365, spinup = NULL, yini = NULL,
    gridtype = 1, Grid = NULL, porosity = NULL, bioturbation = NULL,
    irrigation = NULL, surface = NULL, diffusionfactor = NULL,
    dynamicbottomwater = FALSE, perturbType = "mix", perturbTimes = seq(from = 0,
        to = max(times), by = 365), perturbDepth = 5, concfac = 1,
    CfluxForc = NULL, FePfluxForc = NULL, CaPfluxForc = NULL,
    O2bwForc = NULL, NO3bwForc = NULL, NO2bwForc = NULL, NH3bwForc = NULL,
    ODUbwForc = NULL, PO4bwForc = NULL, DICbwForc = NULL, wForc = NULL,
    biotForc = NULL, irrForc = NULL, rFastForc = NULL, rSlowForc = NULL,
    pFastForc = NULL, MPBprodForc = NULL, gasfluxForc = NULL,
    HwaterForc = NULL, ratefactor = NULL, verbose = FALSE, ...)
NULL
```

The functions to run the model dynamically also allow for several external conditions to be either constants or to vary in time. Thus, they can be set by a parameter or as a forcing function.

These conditions are:

- the flux of carbon, CaP and FeP (parameters *Cflux, CaPflux, FePflux*, forcings *CfluxForc, CaPfluxForc, FePfluxForc*),
- the bottom water concentrations (parameters *O2bw, NO3bw, NO2bw, NH3bw, ODUbw, PO4bw, DICbw*, forcings *O2bwForc, NO3bwForc, NO2bwForc, NH3bwForc, ODUbwForc, PO4bwForc, DICbwForc*)
- the sedimentation, bioturbation and bio-irrigation rates (*w, biot, irr*), (*w, biot, irr*)

- the decay rates of organic matter (parameters *rFast, rSlow*, forcings *rFastForc, rSlowForc*) and the fraction fast organic matter present in the flux (parameter *pFast*, forcing *pFastForc*)
- the air-sea exchange rate when exposed to the air (parameter *gasflux*, forcings *gasfluxForc*)
- the height of the overlying water (parameter *Hwater*, forcing function *HwaterForc*), effective only if *dynamicbottomwater* is *TRUE*.
- *ratefac* is a (time series or a constant) multiplication factor, that is multiplied with all biogeochemical rates. It can be used to impose temperature dependency.

These forcing functions are either prescribed as a list that contains a data series (*list (data = ...)*) or as a list that specifies a periodic signal, defined by the amplitude (*amp*), *period*, *phase*, a coefficient that defines the strength of the periodic signal (*power*) and the minimum value (*min*) : the default settings are: *list(amp = 0, period = 365, phase = 0, pow = 1, min = 0)*. The mean value in the sine function is given by the corresponding parameter.

For instance, for the C flux, the seasonal signal would be defined as: $max(min, Cflux*(1+(amp*sin((times-phase)/period*2*pi))^pow)$.

Three types of perturbations are possible (argument *perturb*):

- *mixing* straightens the profiles over a certain depth
- *erosion* removes part of the surficial sediment
- *deposition* adds sediment on top.

These perturbations are implemented as events, and need input of the perturbation times (*perturbTimes*), and the depth (*perturbDepth*). For deposition events, the factor of increase/decrease of the solid fraction concentration can also be inputted (*concfac*).

Accessory functions

The default values of the parameters, and their units can be interrogated:

```
P  <- CNPDIAparms()
head(P)
```

```
##                 parms       units                description
## Cflux   5.000000e+02 nmolC/cm2/d total organic C deposition
## pFast   9.000000e-01           -   part FDET in carbon flux
## FePflux 0.000000e+00 nmolP/cm2/d     deposition rate of FeP
## CaPflux 0.000000e+00 nmolP/cm2/d     deposition rate of CaP
## rFast   6.849315e-02          /d            decay rate FDET
## rSlow   1.369863e-04          /d            decay rate SDET
```

See the appendix of the main vignette for a complete list of the parameters.

Note: some parameters only apply if the bottom water concentration is modeled dynamically; they comprise the *dilution* of the bottom water (nudging to bottom water concentration), the height of the bottom water (*Hwater*), and the sinking rate of the solid constituents (C, FeP, CaP) (parameters *Cfall*, *CaPfall* and *FePfall*).

Budgets

Once the model is solved, it is possible to calculate budgets of the C, N, P and O2 cycle (*CNPDIAbudgetC*, *CNPDIAbudgetN*, *CNPDIAbudgetP*, *CNPDIAbudgetO2*).

```
DIA <- CNPDIAdyna(parms = list(Cflux = 1500))
std <- CNPDIAperturb(parms = list(Cflux = 1500))

print(CNPDIAbudgetC(std))
```

```
## $Fluxes
##                 FDET          SDET           DIC        CinCaP         Total
## surface  1.350000e+03  1.500000e+02 -1.477293e+03  0.000000e+00  2.270683e+01
## bottom  4.953499e-219 1.542884e-102  8.420476e-04 2.979111e-113  8.420476e-04
## perturb  9.967065e-15 -6.378922e-13 -1.796233e+01  0.000000e+00 -1.796233e+01
## netin    1.350000e+03  1.500000e+02 -1.495256e+03 -2.979111e-113  4.743664e+00
##
## $Rates
##              OxicMineralisation Denitrification AnoxicMineralisation TotalMineralisation CaPprecipita
## nmolC/cm2/d            973.0961        40.86862             486.0353                1500
##              MPBDICuptake MPBResp MPBCdeath EPSproduction EPSmineralisation FDETprodMPBdeath DICprodMI
## nmolC/cm2/d             0       0         0             0                 0                0
##
## $dC
##      DetritusC           DIC           CaP           sum
##   1.275784e-12  6.693083e+00 -1.038175e-112  6.693083e+00
##
## $Losses
## [1] 0.0008420476
##
## $Delta
## [1] 22.70599
##
## $Fluxmat
##                 Ext  DET  DIC CaP EPS MPB        Burial
## Ext     0.000000e+00 1500    0   0   0   0  0.000000e+00
## DET     0.000000e+00    0 1500   0   0   0 1.542884e-102
## DIC     1.477293e+03    0    0   0   0   0  8.420476e-04
## CaP    2.979111e-113    0    0   0   0   0  0.000000e+00
## EPS     0.000000e+00    0    0   0   0   0  0.000000e+00
## MPB     0.000000e+00    0    0   0   0   0  0.000000e+00
## Burial  0.000000e+00    0    0   0   0   0  0.000000e+00
```

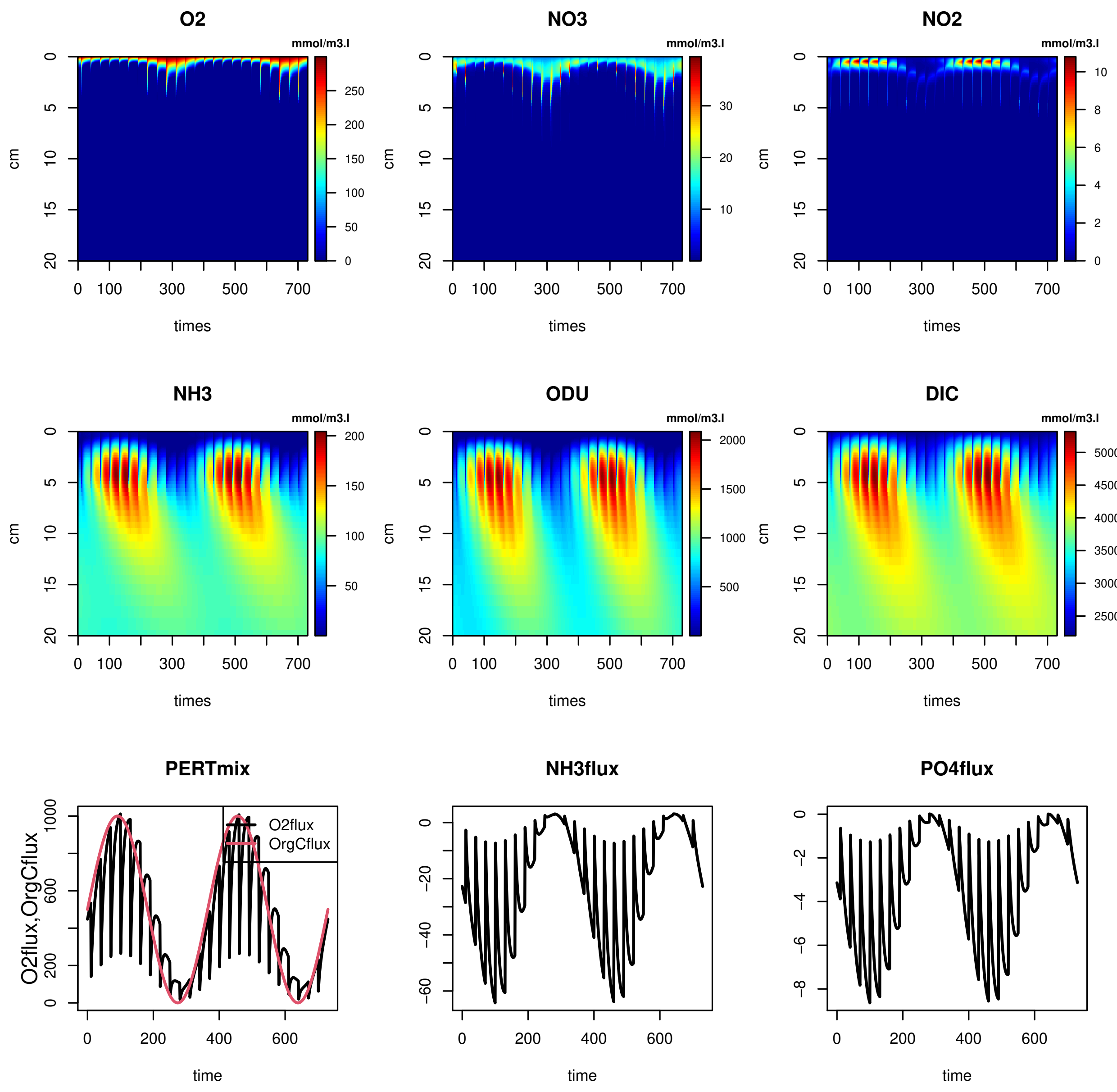
where all fluxes are in *nmol/cm2/day*.

Perturbation runs

Mixing events

We add a mixing event every 30 days, and run this for 2 years, with two years spinup (spinup not shown).

```
times <- 0:730
 PERTmix <- CNPDIAperturb(CfluxForc = list(amp = 1), spinup = 0:730,
       perturbTimes = seq(from = 10, to = 730, by = 30),
       times = times, verbose = FALSE)
image2D(PERTmix, ylim = c(20, 0), which = 3:8, mfrow = c(3,3))
matplot.0D(PERTmix, which = c("OrgCflux", "O2flux"), mfrow = NULL, lty = 1, lwd = 2)
plot(PERTmix, which = c("NH3flux", "PO4flux"), mfrow = NULL, lwd = 2)
```



The instantaneous fluxes

```r
PertFlux <- attributes(PERTmix)$perturbFlux
knitr:::kable(rbind(PertFlux))
```

	time	FDET	SDET	O2	NO3	NO2	NH3	ODU	DIC	PO4
Fluxes	10	0	0	728.3502	4.641279	-1.9662672	-79.23275	-635.3518	-2112.790	-30.47365
Fluxes.1	40	0	0	758.2580	15.601579	-3.5783857	-161.15982	-1362.1152	-3212.035	-45.48553
Fluxes.2	70	0	0	768.9303	20.700937	-4.7063923	-240.19740	-2038.9287	-4242.208	-57.34845
Fluxes.3	100	0	0	772.5427	22.012134	-5.0387971	-305.30855	-2601.1405	-5083.219	-66.23768
Fluxes.4	130	0	0	772.0142	21.998230	-4.9493487	-338.68165	-2901.5355	-5520.269	-70.36240
Fluxes.5	160	0	0	767.5137	20.899858	-4.5020786	-331.80811	-2862.9935	-5440.410	-68.71131
Fluxes.6	190	0	0	757.4003	18.051621	-3.6212876	-287.61035	-2499.8071	-4864.992	-61.56980
Fluxes.7	220	0	0	738.0694	14.007647	-2.4986609	-219.23903	-1913.6569	-3944.478	-50.50700

	time	FDET	SDET	O2	NO3	NO2	NH3	ODU	DIC	PO4
Fluxes.8	250	0	0	705.8355	11.411437	-1.5841850	-145.39525	-1266.3711	-2919.257	-37.95503
Fluxes.9	280	0	0	665.1133	9.181767	-1.0285896	-85.34269	-735.0403	-2057.019	-26.63006
Fluxes.10	310	0	0	640.8262	6.100907	-0.8834740	-52.47735	-445.8632	-1582.921	-19.54946
Fluxes.11	340	0	0	668.9136	4.608598	-1.1653742	-51.64020	-436.5760	-1620.865	-20.37899
Fluxes.12	370	0	0	723.4659	7.191191	-1.8715567	-85.73674	-727.6190	-2161.205	-29.72743
Fluxes.13	400	0	0	755.4395	14.273944	-3.3172920	-150.68632	-1284.0773	-3063.241	-42.51945
Fluxes.14	430	0	0	767.8336	20.255670	-4.5855648	-228.93638	-1951.0798	-4091.917	-54.74258
Fluxes.15	460	0	0	772.2840	21.919568	-5.0180346	-297.53503	-2540.1961	-4979.160	-64.37528
Fluxes.16	490	0	0	772.3794	22.073526	-4.9860269	-336.95667	-2889.9728	-5493.747	-69.50541
Fluxes.17	520	0	0	768.6120	21.182700	-4.6041262	-336.83562	-2908.2715	-5501.681	-68.92962
Fluxes.18	550	0	0	759.6421	18.689484	-3.7964802	-298.12594	-2593.3221	-5001.091	-62.70505
Fluxes.19	580	0	0	742.2067	14.643195	-2.6852750	-232.53485	-2033.4931	-4122.704	-52.21356
Fluxes.20	610	0	0	712.2957	11.763782	-1.7126571	-158.19129	-1382.9877	-3095.829	-39.78890
Fluxes.21	640	0	0	672.2489	9.606216	-1.0911752	-94.67797	-820.4268	-2188.465	-28.13840
Fluxes.22	670	0	0	642.6217	6.723268	-0.8881441	-56.72788	-485.2273	-1637.444	-20.17996
Fluxes.23	700	0	0	661.3367	4.660466	-1.0941433	-50.36673	-428.0560	-1586.645	-19.50877

… compared to mean fluxes

```
rbind(perturbflux = (colSums(PertFlux)/730)[2:7],
      ordinary = summary(PERTmix)[4,c("O2flux","NO3flux", "NH3flux", "ODUflux", "DICflux", "PO4flux")])
```

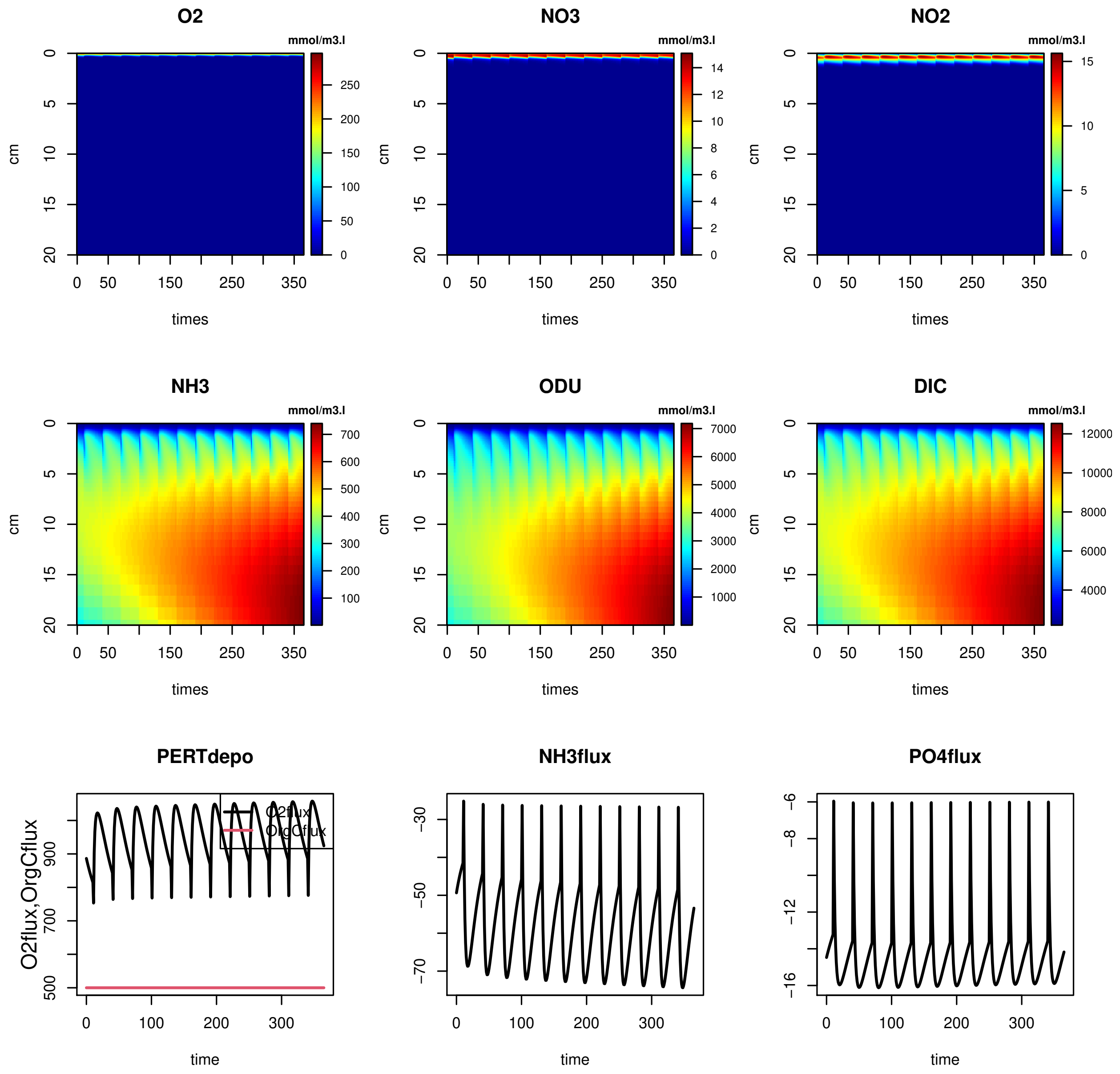
```
##                   O2flux       NO3flux   NH3flux     ODUflux        DICflux   PO4flux
## perturbflux 1.245883e-15  2.392096e-13  23.96457   0.4687658   -0.09749769 -6.336171
## ordinary    4.641356e+02 -1.197149e+01 -21.54392 -11.0171609 -388.09278952 -3.288138
```

deposition events - no loss in organics

We add a deposition event every month, and run this for 1 year, with one year spinup (spinup not shown). Here we keep the deposition flux constant.

In the first run, the deposited sediment has the same concentration of solids as that which is present (concfac = 1).

```
times <- 0:365
  PERTdepo <- CNPDIAperturb(spinup = 0:365, times = times,
         perturbType = "deposit", perturbDepth = 1,
        perturbTimes = seq(from = 10, to = max(times), by = 30), verbose = FALSE)
image2D(PERTdepo, ylim = c(20, 0), which = 3:8, mfrow = c(3,3))
matplot.0D(PERTdepo, which = c("OrgCflux", "O2flux"), mfrow = NULL, lty = 1, lwd = 2)
plot(PERTdepo, which = c("NH3flux", "PO4flux"), mfrow = NULL, lwd = 2)
```



The instantaneous fluxes, compared to the other fluxes

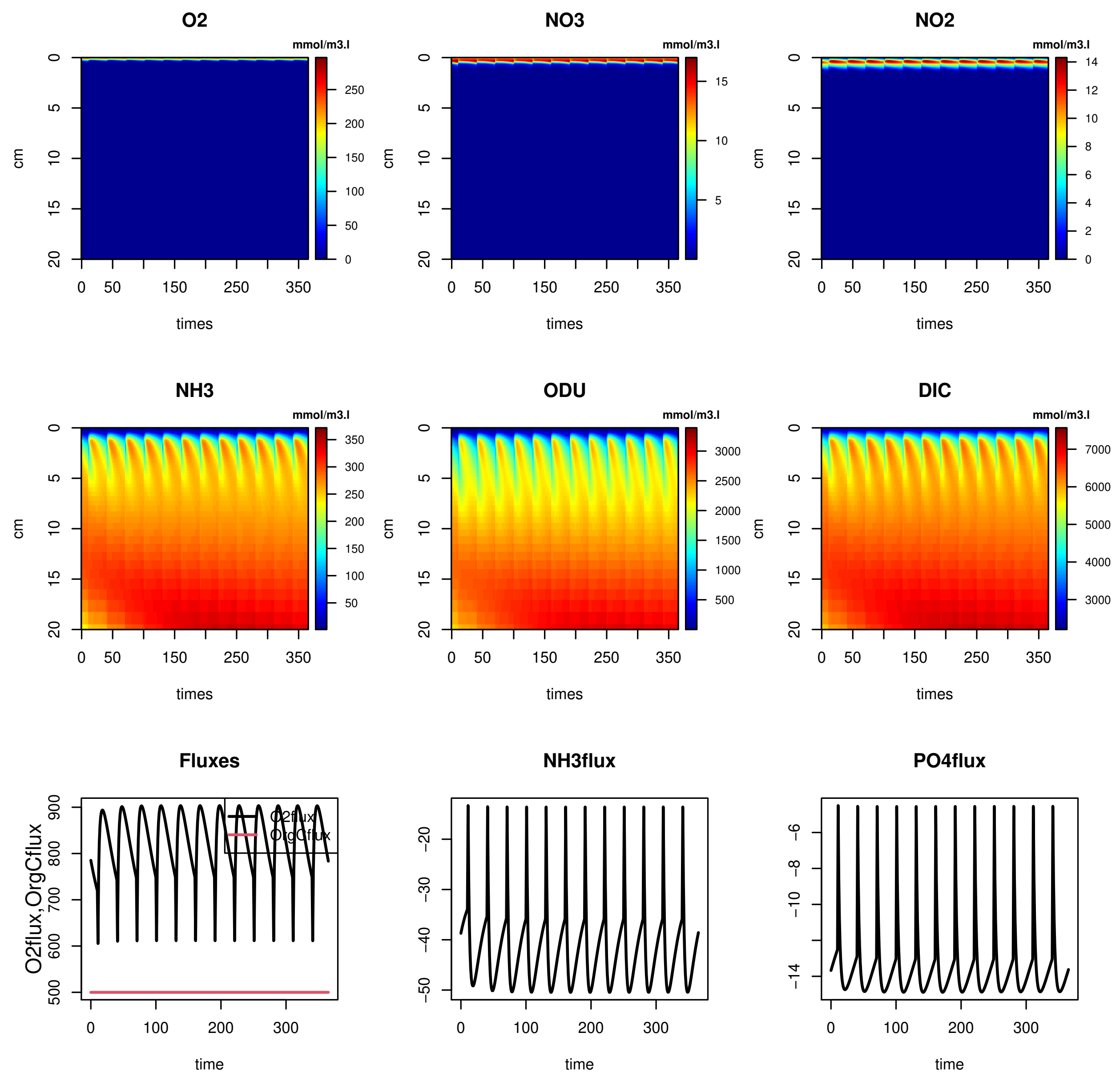
```r
PertFlux <- attributes(PERTdepo)$perturbFlux
rbind(perturbflux = (colSums(PertFlux)/365)[2:7],
      ordinary = summary(PERTdepo)[4,c("O2flux","NO3flux", "NH3flux", "ODUflux", "DICflux", "PO4flux")]
```

```
##                  O2flux   NO3flux    NH3flux      ODUflux        DICflux      PO4flux
## perturbflux 559.0210 4508.9784   6.016375    0.1758123      0.1861834  -0.9941927
## ordinary    960.0839   -16.0926 -58.927480 -252.9729929 -1048.0606571 -14.4544670
```

Deposition - loss in organics

The second run has half the concentration of solids as originally present (concfac = 0.5).

```r
times <- 0:365
PERTdepo2 <- CNPDIAperturb(spinup = 0:365, times = times,
        perturbType = "deposit", perturbDepth = 1, concfac = 0.5,
        perturbTimes = seq(from = 10, to = max(times), by = 30), verbose = FALSE)
image2D(PERTdepo2, ylim = c(20, 0), which = 3:8, mfrow = c(3,3))
matplot.0D(PERTdepo2, which = c("OrgCflux", "O2flux"), mfrow = NULL, lty = 1, lwd = 2, main = "Fluxes")
plot(PERTdepo2, which = c("NH3flux", "PO4flux"), mfrow = NULL, lwd = 2)
```



The instantaneous fluxes, compared to the other fluxes

```r
PertFlux <- attributes(PERTdepo2)$perturbFlux
rbind(perturbflux = (colSums(PertFlux)/365)[2:7],
      ordinary = summary(PERTdepo2)[4,c("O2flux","NO3flux", "NH3flux", "ODUflux", "DICflux", "PO4flux")
```

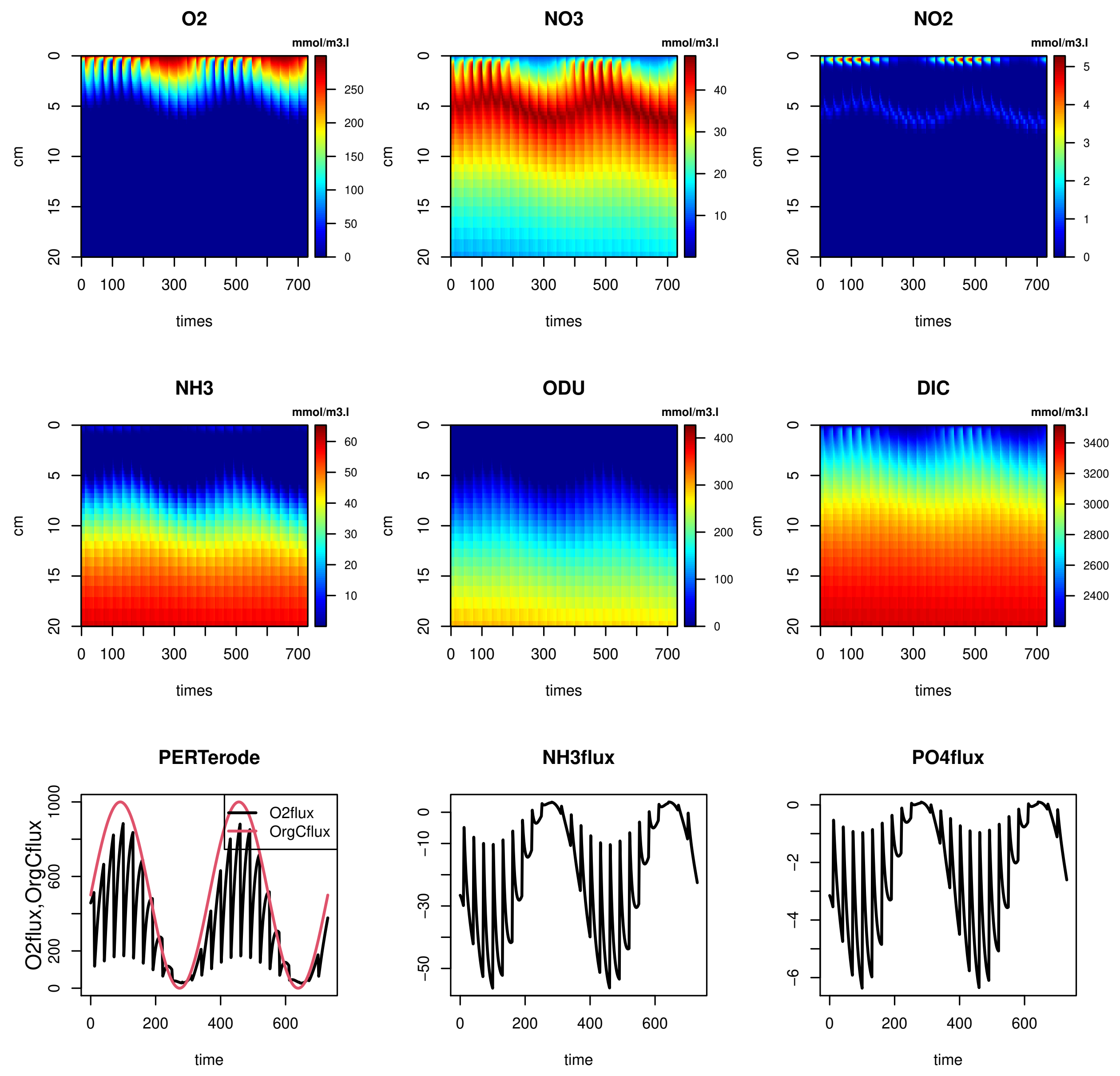
```
##                  O2flux   NO3flux    NH3flux      ODUflux       DICflux    PO4flux
## perturbflux 429.3203 221.79995   5.962672    0.169876    0.1909442  -1.066337
## ordinary    826.9576 -18.61434 -41.953282 -141.321039 -839.2579588 -13.445418
```

erosion

An erosion event every month removes a fraction of the sediment

```r
times <- 0:365*2
PERTerode <- CNPDIAperturb(CfluxForc = list(amp = 1), spinup = 0:365, times = times,
        perturbType = "erode", perturbDepth = 1,
        perturbTimes = seq(from = 10, to = max(times), by = 30), verbose = FALSE)

image2D(PERTerode, ylim = c(20, 0), which = 3:8, mfrow = c(3,3))
matplot.0D(PERTerode, which = c("OrgCflux", "O2flux"), mfrow = NULL, lty = 1, lwd = 2)
plot(PERTerode, which = c("NH3flux", "PO4flux"), mfrow = NULL, lwd = 2)
```



The instantaneous fluxes, compared to the other fluxes

```
PertFlux <- attributes(PERTerode)$perturbFlux
rbind(perturbflux = (colSums(PertFlux)/365)[2:7],
      ordinary = summary(PERTerode)[4,c("O2flux","NO3flux", "NH3flux", "ODUflux", "DICflux", "PO4flux")]
```

```
##                 O2flux    NO3flux    NH3flux     ODUflux       DICflux   PO4flux
## perturbflux -380.3208 -100.82892  -6.893692 -0.72261675   -0.06974625  2.199602
## ordinary     325.3288  -14.46624 -19.732831 -0.04364184 -289.89893744 -2.213270
```

```
CNPDIAbudgetO2(DIA, PERTerode)$Fluxes
```

```
##                     [,1]          [,2]
## O2surf      1.430507e+03  3.250761e+02
## O2deep      0.000000e+00 7.244770e-189
## O2perturb   0.000000e+00 -3.446846e+00
## O2net       1.430507e+03   3.216292e+02
## ODUsurf    -1.855823e+02  -4.362164e-02
## ODUdeep     2.705344e-04   5.847035e-05
## ODUperturb  0.000000e+00   7.490623e+00
## ODUnet     -1.855826e+02   7.446942e+00
```

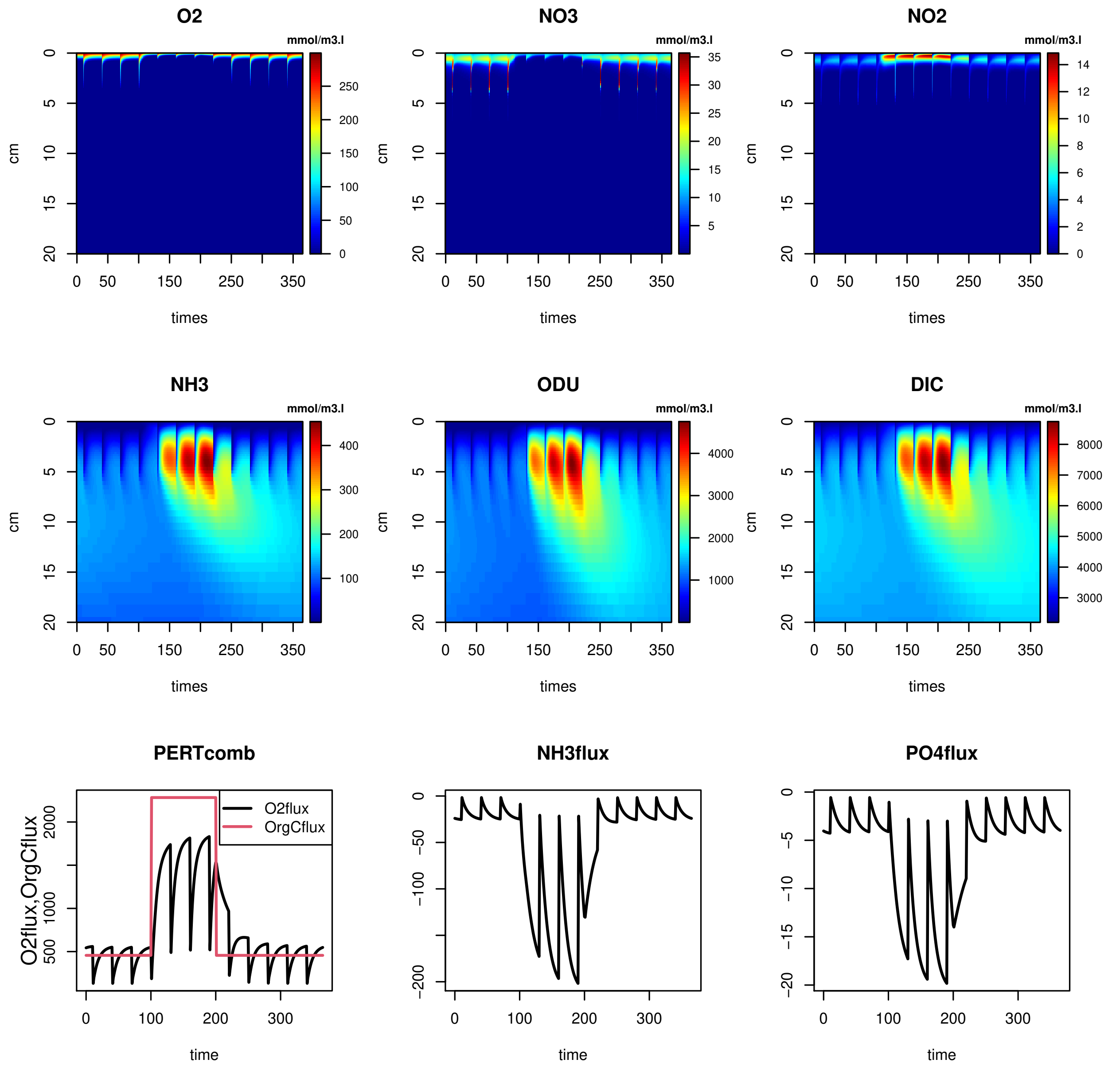
Mixing with variable forcings

We now combine a mixing event with a variable carbon deposition, imposed as a data series.

```
times <- 0:365
fluxforcdat <- data.frame(time = c(0, 100, 101, 200, 201, 365),
                          flux = c(20, 20, 100, 100, 20, 20)*1e5/12/365)
BASE <- CNPDIAdyna(spinup = 0:365, times = times,
        CfluxForc = list(data = fluxforcdat))

PERTcomb <- CNPDIAperturb(spinup = 0:365, times = times,
        CfluxForc = list(data = fluxforcdat),
        perturbTimes = seq(from = 10, to = max(times), by = 30))

image2D(PERTcomb, ylim = c(20, 0), which = 3:8, mfrow = c(3,3))
matplot.0D(PERTcomb, which = c("OrgCflux", "O2flux"), mfrow = NULL, lty = 1, lwd = 2)
plot(PERTcomb, which = c("NH3flux", "PO4flux"), mfrow = NULL, lwd = 2)
```



Mixing with explicitly modeled bottom water conditions

The simulation where bottomwater concentrations are also modeled is initiated with the steady-state conditions, while keeping the bottom water conditions constant.

```
std <- CNPDIAsolve(dynamicbottomwater = FALSE, parms = list(Cflux = 20*1e5/12/365))
CNPDIAbudgetO2(std, which = "Fluxes")
```

```
##                     O2           ODU
## surface  5.233103e+02 -1.192777e-01
## bottom  5.995640e-252  4.926509e-05
## perturb  0.000000e+00  0.000000e+00
## netin    5.233103e+02 -1.193269e-01
```

The initial conditions for the dynamic bottom water concentration run needs to have the bottom water concentrations in the first row.

The model is run for a couple of days.

```r
P <- CNPDIAparms(std, as.vector = TRUE)[c("O2bw","NO3bw","NO2bw", "NH3bw","ODUbw","DICbw","PO4bw")]
BW <- c(0, 0, P, 0, 0, 0)  # order of states, FDET,SDET, "P",FeP,CaP,Pads

dyn  <- CNPDIAdyna  (dynamicbottomwater = TRUE, yini = rbind(BW, std$y),
        parms = list(Cflux = 20*1e5/12/365, Hwater = 3), times = seq(0, 2, length.out = 100))
dyn1 <- CNPDIAperturb(dynamicbottomwater = TRUE, yini = rbind(BW, std$y),
                     perturbType = "mix", perturbTimes = 1, perturbDepth = 1,
        parms = list(Cflux = 20*1e5/12/365, Hwater = 3), times = seq(0, 2, length.out = 100))
dyn2 <- CNPDIAperturb(dynamicbottomwater = TRUE, yini = rbind(BW, std$y),
                     perturbType = "erode", perturbTimes = 1, perturbDepth = 1,
        parms = list(Cflux = 20*1e5/12/365, Hwater = 3), times = seq(0, 2, length.out = 100))
dyn3 <- CNPDIAperturb(dynamicbottomwater = TRUE, yini = rbind(BW, std$y),
                     perturbType = "deposit", perturbTimes = 1, perturbDepth = 1,
        parms = list(Cflux = 20*1e5/12/365, Hwater = 3), times = seq(0, 2, length.out = 100))
```

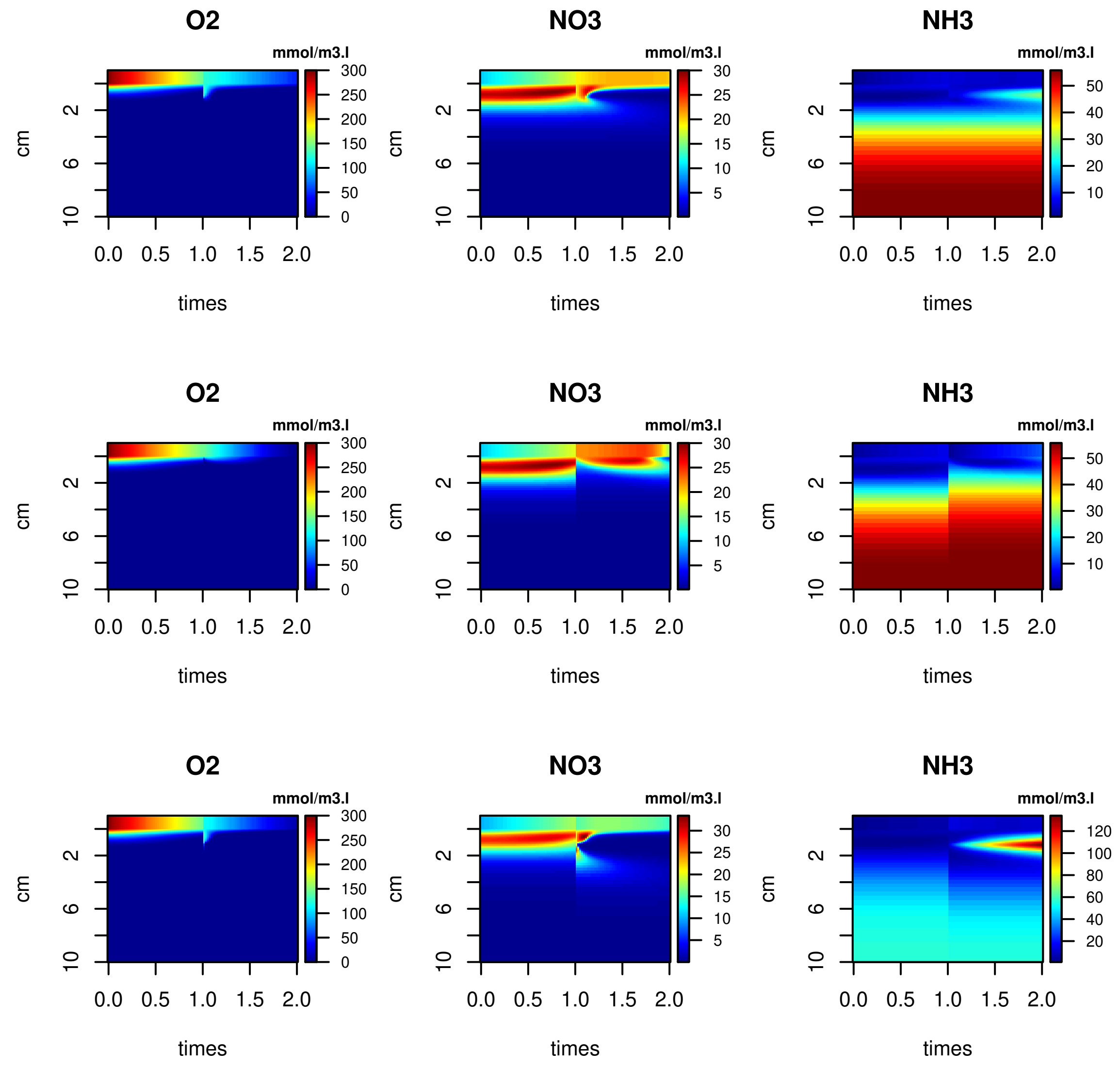
```r
#image(dyn1, which = c("O2", "NO3", "NH3"), grid = c(0,CNPDIAdepth(dyn1)), ylim = c(10,0), legend = TRU
#image(dyn2, which = c("O2", "NO3", "NH3"), grid = c(0,CNPDIAdepth(dyn1)), ylim = c(10,0), legend = TRU
#image(dyn3, which = c("O2", "NO3", "NH3"), grid = c(0,CNPDIAdepth(dyn1)), ylim = c(10,0), legend = TRU

image2D(dyn1, which = c("O2", "NO3", "NH3"), ylim = c(10,-1), mfrow = c(3,3))
image2D(dyn2, which = c("O2", "NO3", "NH3"), ylim = c(10,-1), mfrow = NULL)
```

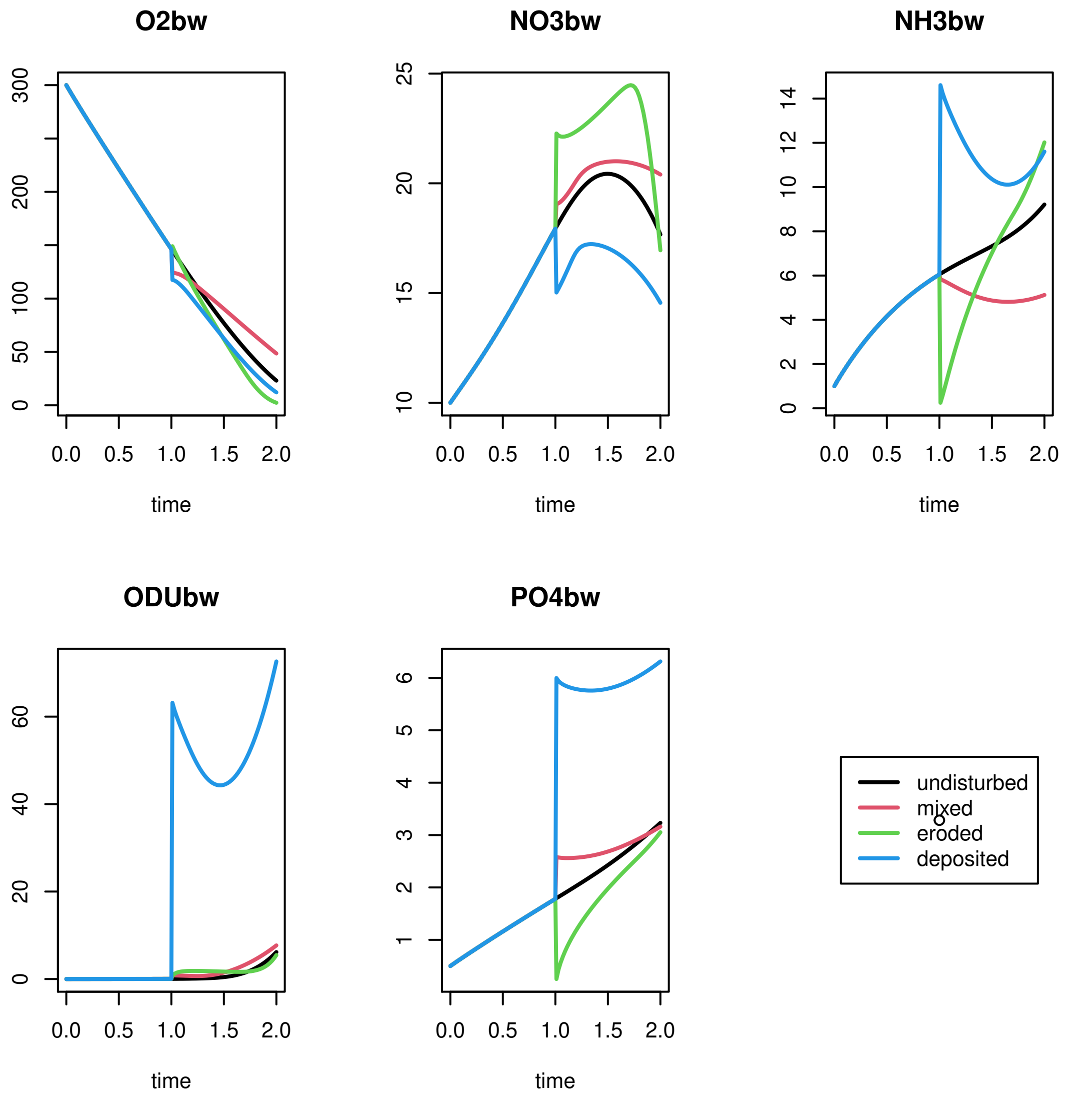
```
## Warning in rep(dots, length.out = n): 'x' is NULL so the result will be NULL
```

```r
image2D(dyn3, which = c("O2", "NO3", "NH3"), ylim = c(10,-1), mfrow = NULL)
```

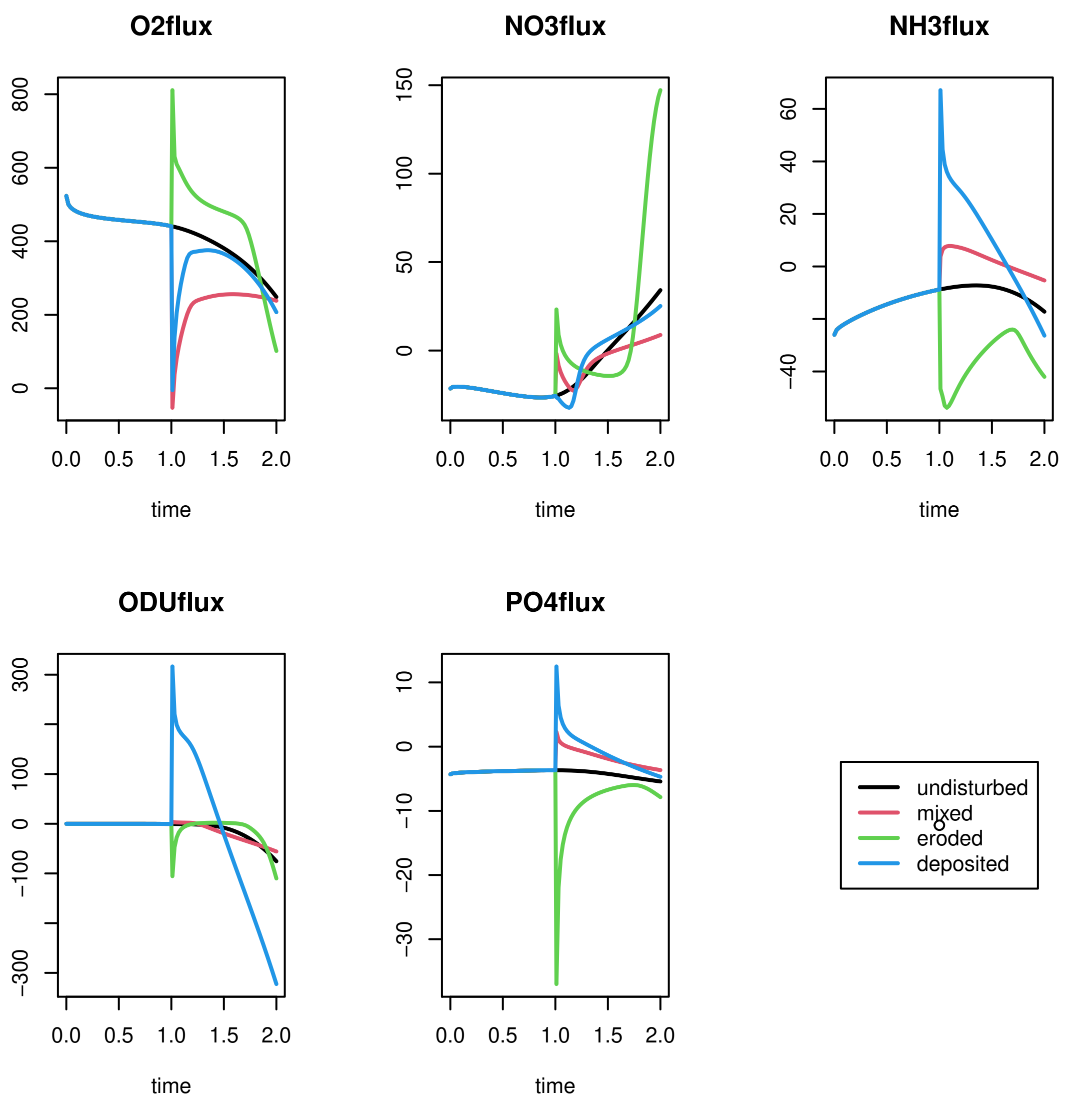
```
## Warning in rep(dots, length.out = n): 'x' is NULL so the result will be NULL
```



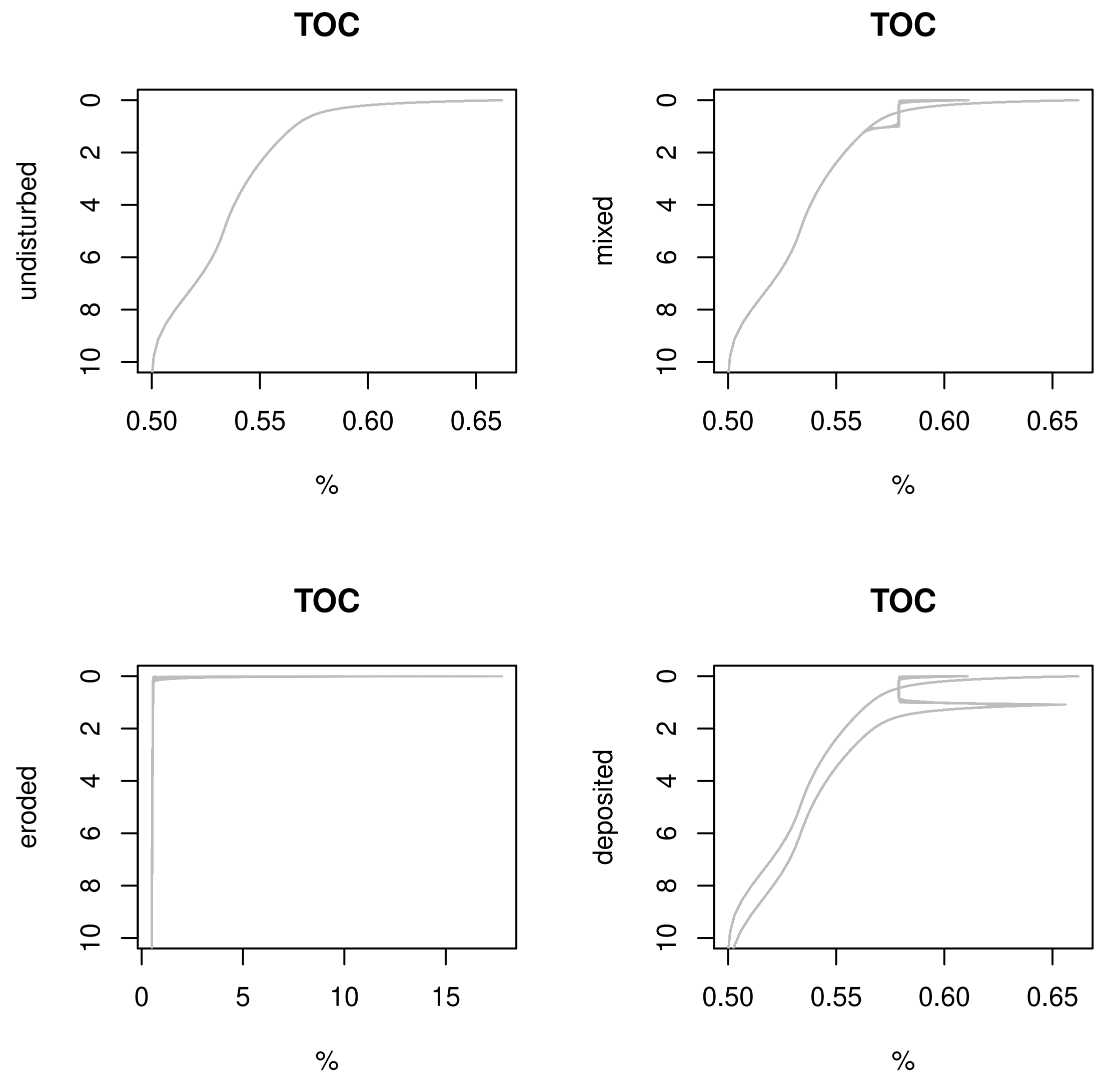
```r
plot(dyn, dyn1, dyn2, dyn3, which = c("O2bw","NO3bw","NH3bw","ODUbw","PO4bw"),
     type = "l", lwd = 2, lty = 1)
plot(0, axes = FALSE, xlab = "", ylab ="")
legend("center", col =1:4, lty = 1, lwd = 2, legend = c("undisturbed", "mixed", "eroded", "deposited"))
```



```r
plot(dyn, dyn1, dyn2, dyn3, which = c("O2flux","NO3flux","NH3flux","ODUflux","PO4flux"), type = "l", lw
plot(0, axes = FALSE, xlab = "", ylab ="")
legend("center", col =1:4, lty = 1, lwd = 2, legend = c("undisturbed", "mixed", "eroded", "deposited"))
```



```
par(mfrow = c(2,2))
matplot1D(dyn, which = "TOC", type = "l",
          col = "grey", ylim = c(10,0), mfrow = NULL, ylab = "undisturbed")
matplot1D(dyn1, which = "TOC", type = "l",
          col = "grey", ylim = c(10,0), mfrow = NULL, ylab = "mixed")
matplot1D(dyn2, which = "TOC", type = "l",
          col = "grey", ylim = c(10,0), mfrow = NULL, ylab = "eroded")
matplot1D(dyn3, which = "TOC", type = "l",
          col = "grey", ylim = c(10,0), mfrow = NULL, ylab = "deposited")
```



References

Soetaert K, PMJ Herman and JJ Middelburg, 1996a. A model of early diagenetic processes from the shelf to abyssal depths. Geochimica Cosmochimica Acta, 60(6):1019-1040.

Soetaert K, PMJ Herman and JJ Middelburg, 1996b. Dynamic response of deep-sea sediments to seasonal variation: a model. Limnol. Oceanogr. 41(8): 1651-1668.