MEDUSA

—

The Basic Framework

Guy Munhoven
Université de Liège, Belgium
http://www.astro.ulg.ac.be/~munhoven

15th June 2020

Contents

1 Introduction

1.1 Early diagenesis equations

MEDUSA (Model of Early Diagenesis in the Upper Sediment with Adaptable complexity) solves the one-dimensional Advection-Diffusion-Reaction equations of early diagenesis, with fast chemical reactions eliminated by adopting the hypothesis of local chemical equilibrium

$$\frac{\partial \hat{C}_i}{\partial t} + \frac{\partial \hat{J}_i}{\partial z} - \hat{S}_i = 0 \tag{1}$$

where t is time and z depth below the sediment-water interface (positive downwards), $\hat{C}_i$ denotes the concentration of i per unit volume of total sediment (solids plus porewater), $\hat{J}_i$ is the local transport (advection and diffusion), per unit surface area of total sediment, and

$$\hat{S}_i = \hat{R}_i + \hat{r}_i + \hat{Q}_i \tag{2}$$

represents the net source-minus-sink balance for constituent i per unit volume of total sediment, $\hat{R}_i$ being the net reaction rate, $\hat{r}_i$ the net fast interconversion rate, that is going to be filtered out of the equations by an equilibrium consideration and $\hat{Q}_i$ the non-local transport (considered only for solutes). Solids are transported by advection throughout the sediment column and subject to bioturbation in the surface mixed layer. Bioturbation is represented as a diffusive process. Both inter- and intraphase biodiffusion variants (Boudreau, 1986; Mulsow et al., 1998) are taken into account and can be combined:

$$\hat{J}_i = -D_i^{\text{inter}} \frac{\partial \varphi^{\text{s}} C_i^{\text{s}}}{\partial z} - \varphi^{\text{s}} D_i^{\text{intra}} \frac{\partial C_i^{\text{s}}}{\partial z} + \varphi^{\text{s}} w C_i^{\text{s}}.$$

Biodiffusion coefficients are supposed to be the same for all solids within a given sediment column: $D_i^{\text{inter}}(z) \equiv D^{\text{inter}}(z)$ and $D_i^{\text{intra}}(z) \equiv D^{\text{intra}}(z)$. For convenience, we define $D^{\text{bt}}(z) = D^{\text{inter}}(z) + D^{\text{intra}}(z)$ and $D^{\text{inter}}(z) = \beta(z) D^{\text{bt}}(z)$. $\hat{J}_i$ can then be rewritten as

$$\hat{J}_i = -\varphi^{\text{s}} D^{\text{bt}} \frac{\partial C_i^{\text{s}}}{\partial z} + \left(\varphi^{\text{s}} w - \beta D^{\text{bt}} \frac{\partial \varphi^{\text{s}}}{\partial z} \right) C_i^{\text{s}}. \tag{3}$$

Solutes in surface sediments are transported by molecular and ionic diffusion in porewaters, by interphase bioturbation, porewater advection and by bioirrigation. In the absence of impressed flow, transport by porewater advection is, however, negligible compared to diffusion; biodiffusion coefficients

are furthermore an order of magnitude lower than molecular and ionic diffusion coefficients. The expression for the local transport term of a porewater solute i adopted in MEDUSA then reduces to

$$\hat{J}_i = -\varphi^{\mathrm{f}} \frac{D_i^{\mathrm{sw}}}{\theta^2} \frac{\partial C_i^{\mathrm{f}}}{\partial z}, \tag{4}$$

where D_i^{sw} is the free diffusion coefficient of the solute i in seawater, θ^2 is tortuosity, parametrized as function of porosity.

Bioirrigation provides a non-local transport mode for solutes. In MEDUSA, the source-sink approach (Boudreau, 1984) is used to quantify the effect of bioirrigation:

$$\hat{Q}_i(z) = \alpha(z)\varphi^{\mathrm{f}}(z)(C_i^{\mathrm{oc}} - C_i^{\mathrm{f}}(z))$$

where $\alpha(z)$ is the bioirrigation constant, which may be depth-dependent, and C_i^{oc} is the concentration of solute i in the irrigation channels, set equal to the solute's concentration in seawater overlying the sediment.

The equation system is amended by taking into account two constraints.

1. One of the solids' evolution equations is replaced by the static volume conservation equation:

$$\sum_{i \in Y^{\mathrm{s}}} \vartheta_i C_i^{\mathrm{s}} = 1 \tag{5}$$

   where Y^{s} denotes the inventory of solid components considered in the model configuration and ϑ_i the partial specific volume of solid i, supposed to be invariable.

2. For the solids' advection profile, the depth-integrated solid phase volume conservation equation is used

$$\varphi^{\mathrm{s}}(z)w(z) - \beta(z)D^{\mathrm{bt}}(z)\frac{\partial \varphi^{\mathrm{s}}}{\partial z} = \sum_{i \in \mathrm{Y^s}} \vartheta_i \hat{I}_i^{\mathrm{top}} + \int_{z_T}^{z} \sum_{i \in \mathcal{I}^{\mathrm{s}}} \vartheta_i \hat{R}_i(z')\, dz', \tag{6}$$

   where $\hat{I}_i^{\mathrm{top}}$ denotes the deposition rate of solid component i per unit surface of total sediment per unit time, entering the surface sediment through the sediment-water interface at the top.

The rapid interconversion reaction rates r_i are filtered out of the equation system on the assumption that the corresponding chemical reactions are in local thermodynamic equilibrium. Practically, subsets of equations are replaced by carefully selected linear combinations of themselves, and others replaced by the equilibrium relationships. The code generator takes care of these transformations, on the basis of auxiliary information about solutes that form systems and the stoichiometry of the chemical equilibria between the members of these systems.

1.2 System of units

The fundamental units adopted in MEDUSA are as follows:

Mass –
solutes: moles
solids: kg

Lengths – m

Time – yr
one may chose between three different calendars, with 360, 365 and 365.25 days/yr, resp., where 1 day = 24 hours, 1 hour = 3600 s (see section 4.3 below).

Temperature – °C

1.3 Sediment Column Partitioning and Vertical Discretisation

A complete sediment column in MEDUSA is subdivided into three different vertically stacked parts called realms (Fig. 1):

REACLAY — the top-most part extending downwards from the sediment top at the sediment-water interface and where chemical reactions are taken into consideration;

TRANLAY — the transition layer of changing thickness just underneath, acting as a temporary storage to connect REACLAY to the underlying TRANLAY;

CORELAY — a stack of sedimentary layers representing the deep sediment, i. e., the sediment core;

DBL [optional] — a Diffusive Boundary Layer acting as a diffusive barrier to the sediment-water exchange of solutes can be included on top of the REACLAY realm.

REACLAY encompasses the bioturbated sedimentary mixed-layer, where most of the reactions relevant for early diagenesis take place (organic matter remineralisation, carbonate dissolution etc.). The bottom of the bioturbation zone may coincide with the bottom of REACLAY or be situated within it.

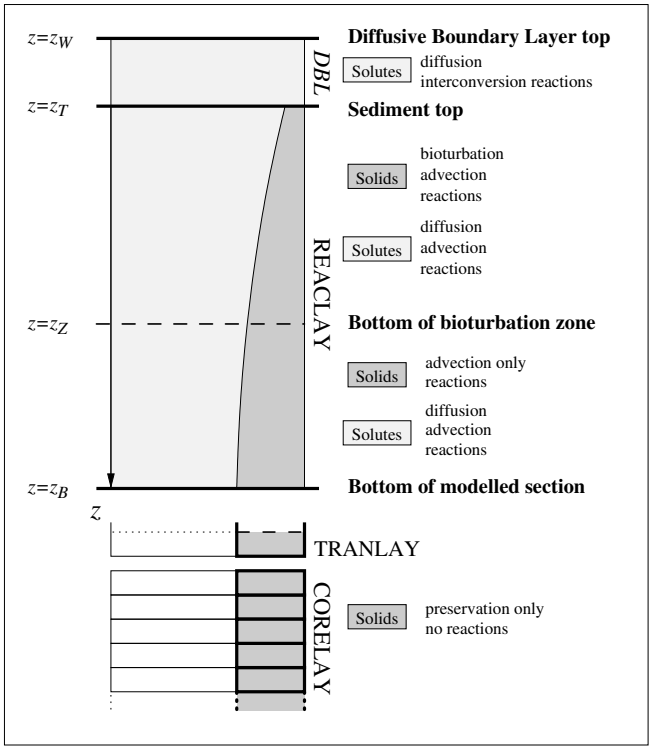

Figure 1: Partitioning of the sediment column in MEDUSA: an optional diffusive boundary layer (DBL) on top of the main part of the model sediment where diagenetic reactions and advective-diffusive transport take place (REACLAY), the transition layer (TRANLAY) and the core represented by the stack of layers (CORELAY). The bottom of the bioturbation zone may coincide with the bottom of REACLAY (see text for details).

The numerical solution is based upon a finite volume approach. The adopted discretisation grid is vertex-centred: the boundaries between the finite volume elements are located mid-way between the element grid-points. Fig. 2 illustrates the main characteristics of the grid? For the local fluxes (diffusive and advective), an upwind-weighted spatial discretisation is used with two schemes to choose: full upwind or exponential fitting.

1.4 Time stepping

MEDUSA may be used for transient or steady-state simulation experiments:

- for transient simulation experiments, time stepping is fully implicit (implicit Euler);
- for steady-state simulation experiments, the time-step lengths must be set to zero; the solver routines detect this special value and proceed to the direct solution for steady state in this case.

1.5 Source code tree

The source code tree is organised as follows:

`[trunk]/src-med`	– MEDUSA framework and build directory, including standard templates (in `templates`)
`/src-med/gen`	– destination sub-directory for the generated code parts and templates
`/src-mcg`	– MEDUSACOCOGEN (MEDUSA Configuration and Code GENeration tool, including the rate-law and law-of-mass-action library modules (in `lib`), XML description file collections and temporary work space for code generation (in the subdirectories `tmp` and `gen`)
`/apps`	– applications
`/docs`	– documentation

2 MEDUSA Code Framework

2.1 `solvsed_onestep.F` and `solvsedFVFullUpwind.F`: assembling and solving the equations

The subroutine `SOLVSED_ONESTEP` (source file `solvsed_onestep.F`) carries out one time-step for all of the columns registered in the central storage

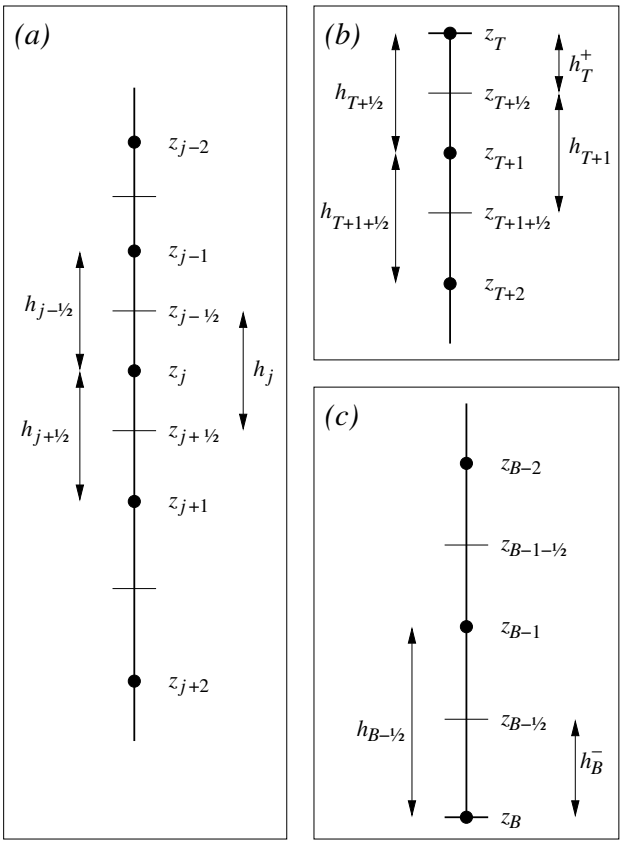

Figure 2: Grid definitions: (a) interior, (b) top and (c) bottom nodes of the REACLAY grid. Boundaries (vertices) between adjacent finite volumes are located at centres (denoted $T + \frac{1}{2}, \ldots, B - \frac{1}{2}$) between nodes; the upper boundary of the topmost volume is located at the top node (T) and the lower boundary of the bottom volume is located at the bottom node (B).

area (MOD_SEAFLOOR_CENTRAL – see below). For each column, it loads relevant data into the workspace modules (MOD_GRIDPARAM, MOD_MILIEUCHARAS, MOD_MATERIALCHARAS,...) and updates information held in others (such as MOD_TRANSPORT, ...). It then proceeds to determine the new column state at the end of the time step depending on the boundary conditions. The equation system used to describe the evolution of the model sediment and its pore waters is then assembled and solved by the subroutine ImplicitTimeStep from solvsedFVFullUpwind.F. The solution method is iterative (damped Newton scheme) and one critical stage is the initialization of that scheme. SOLVSED_ONESTEP offers a sequence of strategies to carry out this initialization and calls ImplicitTimeStep in turn until a satisfactory solution to the system is found. There are currently seven such strategies implemented:

BASIC — initialise the column concentrations (both solids and solutes) with the state of the initial instant of the time-step;

SOLUT — initialize the column porewater concentrations homogeneously with the boundary conditions and the solids' concentrations as in the initial instant of the time-step;

PSVOL — a continuation method, where the partial specific volumes of all the solids except for the mandatory clay are increased from zero to their actual value in 10 steps, and at each step, concentration profiles are calculated by starting the iterations with the results from the previous step;

RREAC — another continuation method, where the reaction rates are increased from zero to their full rate in 30 steps, and at each step, concentration profiles are calculated by starting the iterations with the results from the previous step;

WFFLX — another continuation method, where the mass fraction of non-lithogenic solids in the solid flux entering the sediment from above is increased from zero to its actually requested fraction in 10 steps, and at each step, concentration profiles are calculated by starting the iterations with the results from the previous step;

TIMST — another continuation method, only used for steady-state calculations, where the time step is increased from 10 to 10^6 years in 20 steps and then to ∞, and each step, concentration profiles are calculated by starting the iterations with the results from the previous step;

`COLLA` — yet another continuation method, only used in case the sediment column is subject to chemical erosion: in this case the volume fraction of non-lithogenic solids in the is replaced by its equivalent volume of lithogenic (inert) material is gradually increased from pure clay to its actual volume composition in 10 steps, and at each step, concentration profiles are calculated by starting the iterations with the results from the previous step.

For the three continuation methods, the very first iterative calculation of the sediment state is initialized by the initial state of the time step being carried out (as with `BASIC`).

2.2 MOD_EXECONTROL_MEDUSA: model execution control

The `MOD_EXECONTROL_MEDUSA` module is most relevant for MPI-enabled applications. For serial applications, only the `ABORT_MEDUSA` subroutine is used from this module. That subroutine correctly closes all the log, debug and error files before aborting, thus making sure that those file are complete.

With most compilers, the call to `ABORT()`, if supported, triggers the sequence of subroutine and function subprogram calls up to the abortion point to be printed out. this information may be useful for understanding the reasons for the program failure.

2.3 MOD_BASICDATA_MEDUSA: basic data and constants

`MOD_BASICDATA_MEDUSA` (source file `src-med/mod_basicdata_medusa.F`) includes a richly documented collection of basic physical data, such as elemental molar weights, physical constants (gas constant, unit conversion constants, ...), standard isotopic ratios ($^{13}C/^{12}C$ (PDB), $^{11}B/^{10}B$ (NIST SRM 951), ...) or radioactive decay constants. It also provides parameter constants for common unit conversions (e.g., cm^2/s in m^2/yr).

2.4 MOD_DEFINES_MEDUSA: global parameter, selector and flag values

`MOD_DEFINES_MEDUSA` (source file `src-med/mod_defines_medusa.F`) centralizes the logical unit numbers used by the log, error and debug files in MEDUSA.

2.5 MOD_SEAFLOOR_CENTRAL: multi-column management

`MOD_SEAFLOOR_CENTRAL` (source file `src-med/mod_seafloor_central.F`) is the central data turntable and provides important parts of the Application

Programming Interface (API) to the MEDUSA framework system.

The current state of the complete sediment (all columns modelled) is stored here. Dedicated subroutine and function subprograms are provided here to transfer the data between this central storage area and the single-column workspace where the solver takes them to solve the equations.

2.6 MOD_SEDCORE: the sediment core system

The MOD_SEDCORE module (in src-med/mod_sedcore.F) manages and controls the synthetic cores generated as solids get buried or chemically eroded, i.e., cross the bottom boundary of the sediments reactive part.

2.7 MOD_GRIDPARAM: grid related initialization and workspace

The MOD_GRIDPARAM module (src-med/mod_gridparam.F) holds the information about the model grid data. It also provides the workspace for the grid-point distribution of a column during its processing. Finally, it contains subroutines to initialize grid-point distributions.

The fundamental structure of the grid, i.e., the number of grid points in the various parts of the sediment column is set there by a series of PARAMETER constants:

ndn_w2t — the number of grid nodes in the DBL, i. e., in the interval $[z_W, z_S[$ (notice z_S excluded!) — may be zero;

ndn_t2z — the number of grid nodes between the sediment-water interface (or the sediment-DBL interface if a DBL is present) and the depth in the REACLAY realm where bioturbation stops, i. e. in the $[z_S, z_Z]$ interval;

ndn_z2b — the number of grid nodes between the depth in REACLAY where bioturbation stops and the bottom of REACLAY, i. e., in the $]z_Z, z_B]$ interval (notice z_Z excluded!) — may be zero.

Standard values are: ndn_w2t = 0 (no DBL), ndn_t2z = 21 and ndn_z2b = 0 (completely bioturbated REACLAY).

The dimensional characteristics of the grid are the same for all sediment columns in MOD_SEAFLOOR_CENTRAL: all the grids have the same number of grid-points in the different realms (DBL, REACLAY) and parts (e.g., the extent of the bioturbated layer, which is part of REACLAY).

Default parameter values for the various distributions can be overridden by using a configuration file `medusa_grid_config.nml`. Details about how this can be done in practice are provided in the comments of the template `medusa_grid_config.nml_template` in `src-med/templates`. Such a file must always start with a main namelist called `&nml_grid_options/`, with the following entries and default values:

```
&nml_grid_options
ctype_grid_variability       = 'static_global'
ctype_gridpoint_distribution = 'quad_lin'
/
```

Entries for which default values are to be adopted may simply be left out (as always with namelists). The namelist must always be present, even if the default settings should be used and it might be left empty.

2.7.1 Grid variability

Although the number of grid points is fixed, the grid-point distribution (spacing, extent, ...) may be different among columns and also change in time. The variability of the grid-point distribution can be set to

`static_global` — the same distribution is used for all columns and it does not change in time [***default***];

`static_local` — each column may use its own grid-point distribution, which does not change in time;

`dynamic` — grid-point-distributions may be different from column to column and change in time [not yet fully implemented].

This information is most relevant for the NetCDF files: for `static_global` grids, only one grid-point distribution (one 1D array) is written that is valid for all grids; for `static_local` grids, one array of grid-point distributions is written (organized as a 2D array); for `dynamic` grids, the grid-point distributions (2D arrays) are included with each time record.

The equations and Jacobians assembled in `solvsedFVFullUpwind` are currently only consistent with `static_global` and `static_local` grid-point distributions.

2.7.2 Grid-point distribution functions

There are currently six grid-point distribution functions implemented. In the following, it is assumed that the grid points are indexed from i_T – sediment-water interface – through i_Z – bottom of the bioturbated layer – to i_B at the

bottom of REACLAY). Their identifiers for usage in the configuration file are as follows:

`linear` — linear distributions, separately in the bioturbated and in the non-bioturbated parts of REACLAY:

$$z_i = L_\mathrm{B}\frac{i - i_\mathrm{T}}{i_\mathrm{Z} - i_\mathrm{T}}, \quad i \le i_\mathrm{Z}$$

$$z_i = L_\mathrm{B} + (L - L_\mathrm{B})\frac{i - i_\mathrm{Z}}{i_\mathrm{B} - i_\mathrm{Z}}, \quad i > i_\mathrm{Z}$$

Default values are: $L_\mathrm{B} = L = 10\,\mathrm{cm}$.

`quad_lin` — quadratic top and linear bottom (Boudreau, 1997, eq. (8.156), p. 333), separately in the bioturbated and in the non-bioturbated parts of REACLAY [***`default`***]:

$$z_i = L_\mathrm{B}\frac{\sqrt{(\frac{i-i_\mathrm{T}}{i_\mathrm{Z}-i_\mathrm{T}})^2 + \chi_\mathrm{T}^2} - \chi_\mathrm{T}}{\sqrt{1 + \chi_\mathrm{T}^2} - \chi_\mathrm{T}}, \quad i \le i_\mathrm{Z}$$

$$z_i = L_\mathrm{B} + (L - L_\mathrm{B})\frac{\sqrt{(\frac{i-i_\mathrm{Z}}{i_\mathrm{B}-i_\mathrm{Z}})^2 + \chi_\mathrm{B}^2} - \chi_\mathrm{B}}{\sqrt{1 + \chi_\mathrm{B}^2} - \chi_\mathrm{B}}, \quad i > i_\mathrm{Z}$$

Default values are: $L_\mathrm{B} = L = 10\,\mathrm{cm}$ and $\chi_\mathrm{T} = \chi_\mathrm{B} = 0.5$.

`quad_quad` — quadratic top and quadratic bottom, over the complete REACLAY, i.e., mirrored around $L/2$:

$$z_i = L\frac{\sqrt{(\frac{i-i_\mathrm{T}}{i_\mathrm{B}-i_\mathrm{T}})^2 + \chi^2} - \chi}{\sqrt{1 + \chi^2} - \chi}, \quad i < i_\mathrm{T} + (i_\mathrm{B} - i_\mathrm{T})/2$$

$$z_{i_\mathrm{T}+(i_\mathrm{B}-i_\mathrm{T})/2} = L/2, \quad \text{if } i_\mathrm{B} - i_\mathrm{T} \text{ even}$$

$$z_i = L - z_{i-i_\mathrm{B}+i_\mathrm{T}}, \quad i > i_\mathrm{T} + (i_\mathrm{B} - i_\mathrm{T})/2$$

Default values are: $L_\mathrm{B} = L = 10\,\mathrm{cm}$ and $\chi = 0.5$.

`geomprog_open` — grid points distributed such that the distances between consecutive grid-points form a geometric progression, where the scale factor δ and the ratio r are given:

$$\begin{aligned} z_{i_\mathrm{T}} &= 0 \\ z_i &= z_{i-1} + \delta r^{i-(i_\mathrm{T}+1)}, \quad i_\mathrm{T} < i \le i_\mathrm{B} \end{aligned}$$

Default values are: $\delta = 1\,\mathrm{mm}$ $r = 1.06$.

It is called *open* because the REACLAY thickness is given by the resulting z_{i_B} which is a function of n, δ and r.

geomprog_closed — grid points distributed such that the distances between consecutive grid-points form a geometric progression, where the scale factor δ is given and the ratio r is calculated such that $z_{i_\mathrm{B}} = L$:

$$\begin{aligned} z_{i_\mathrm{T}} &= 0 \\ z_i &= z_{i-1} + \delta r^{i-(i_\mathrm{T}+1)}, \quad i_\mathrm{T} < i \leq i_\mathrm{B} \end{aligned}$$

By default, $L = 10\,\mathrm{cm}$ and $\delta = 1\,\mathrm{mm}$.

It is called *closed* because the REACLAY thickness L is prescribed.

custom — use a custom grid-point distribution scheme. This requires that

- a source file named `gridef_custom.F` containing two subroutines, named `GRIDEF_CUSTOM` (to evaluate the custom distribution function) and `SETUP_GRIDEF_CUSTOM` (to initialize its parameters) is copied or linked into `src-med/gen/include` after the code generation stage completes and before the library is built (i.e., between `make codegen` and `make libmedusa.a`);
- the pre-processor switch `-DGRID_CUSTOM` is used to compile and build `libmedusa.a`, so that `gridef_custom.F` is embedded into `MOD_GRIDPARAM`.

A template that shows how to organize such a `gridef_custom.F` is provided in `gridef_custom.F_template` in `src-med/templates`. Please notice that `-DGRID_CUSTOM` does not activate the custom distribution function. It still needs to be activated with the configuration file.

If a DBL is included it is overlaid by a linear grid-point distribution prepended to the REACLAY grid and indexed from i_W – the top of the DBL – to $i_{\mathrm{T}-1}$. the two grids are not connected at a node in this case, but at a vertex (cell boundary).

2.8 MOD_MILIEUCHARAS: milieu properties initialization and workspace

`MOD_MILIEUCHARAS` (source file `src-med/mod_milieucharas.F`) holds data and subprograms that describe the milieu characteristics of the sediment. It provides the workspace for the following during the processing of a column:

- porosity profile ($\varphi(z_i)$, $\partial\varphi/\partial z|_{z_i}$)
- tortuosity parametrization ($\vartheta(z_i)$).

It also contains subroutines to initialize or update these profiles.

Default parameter values related to the porosity profiles and tortuosity relationships can be overridden by using an optional namelist configuration file `medusa_milieu_config.nml` that will be read in at run-time from the work directory if present. The extensively commented template `medusa_milieu_config.nml_template` in `src-med/templates` provides the details about the possibilities offered by this file. Such a file must always start with a main namelist called `&nml_milieu_options/`, with the following entries and default values:

```
&nml_milieu_options
ctype_porosity_variability = 'static_global'
ctype_phi_profile          = 'expdec'
ctype_tortuosity_rel       = 'modweissberg'
/
```

Entries for which default values are to be adopted may simply be left out (as always with namelists). The namelist must always be present, even if the default settings should be used and it might be left empty.

2.8.1 Porosity profile variability

The variability of the porosity profile can be managed similarly to the grid-point distribution:

`static_global` — the same profile is used for all columns and it does not change in time [***default***];

`static_local` — each column may use its own porosity profile, which does not change in time;

`dynamic` — porosity profiles may be different from column to column and change in time.

Similarly to the grid variability, this information is again most used while defining the variables for the NetCDF files and writing them.

The equations and Jacobians assembled in `solvsedFVFullUpwind` are currently only consistent with `static_global` and `static_local` porosity profiles.

2.8.2 Porosity profiles

There are currently three different porosity profiles available:

`const` — a constant porosity profile

$$\varphi(z) = \varphi_0$$

The default value is: $\varphi_0 = 0.9$.

`expdec` — an exponentially decreasing porosity profile [***default***]

$$\varphi(z) = \varphi_\infty + (\varphi_0 - \varphi_\infty) \exp(-z/\zeta_\varphi)$$

The default parameter values are: $\varphi_0 = 0.9$, $\varphi_\infty = 0.7$ and $\zeta_\varphi = 4\,\mathrm{cm}$.

`custom` — an custom profile. The pre-processor switch `-DPHI_CUSTOM` allows an custom porosity profile to be included in the code, which can then be activated by selecting it in the configuration file.

Each profile may offer additional adjustable parameters that can be adapted with additional namelists in the configuration file. For the standard profiles (i.e., non `custom` profiles), these are detailed in the template configuration file and in the `SETUP_TRANSPORT` subroutine.

2.8.3 Tortuosity parametrizations

There are currently three tortuosity parametrizations available:

`archie` — Archie's law

$$\theta^2(z) = \varphi(z)^{1-m}$$

The default parameter value is: $m = 2.14$ (Boudreau, 1997).

`burgerfrieke` — Burger-Frieke relationship

$$\theta^2(z) = a(1 - \varphi(z))$$

The default parameter value is: $a = 3.14$ (Boudreau, 1997).

`modweissberg` — modified Weissberg relationship [***default***]

$$\theta^2(z) = 1 - b \ln \varphi(z)$$

The default parameter value is: $b = 2.02$ (Boudreau, 1997).

It is currently not possible to use custom tortuosity parametrizations.

2.9 MOD_TRANSPORT: transport parameter initialization, processing and workspace

MOD_TRANSPORT (in src-med/mod_transport.F) is a "semi-finished" module, that needs to be completed by composition-dependent parts (e.g., diffusion coefficients for solutes, ...).

It holds the data relating to

- the biodiffusion profile: $D_B(z_i)$, $\eta(z_i)$ and their derivatives with respect to z)
- the bioirrigation profile: $\alpha(z_i)$

of the sediment column being currently processed. MOD_TRANSPORT furthermore contains subroutines to calculate the advection rate profiles (for solids, for solutes), to evaluate the different flux terms (local and the non-local), and utility subprograms for the application of the available upwinding schemes (full upwind and exponential fitting).

It must be finished by a subroutine to evaluate the molecular and ionic diffusion coefficients for the porewater solutes to be considered. This subroutine (MDIFFC) is generated by MEDUSACOCOGEN.

Default parameter values for the transport processes (biodiffusion and bioirrigation coefficient profiles, upwinding) can be overridden by using an optional namelist configuration file medusa_transport_config.nml that will be read in at run-time from MEDUSA's working directory if present. The comments in the template medusa_transport_config.nml_template which can be found in src-med/templates provide the details about the possibilities offered by this file. Such a file must always start with a main namelist called &nml_transport_options/, with the following entries and default values:

```
&nml_transport_options
ctype_biodiffusion  = 'const'
ctype_bioirrigation = 'none'
ctype_upwinding     = 'full'
/
```

Entries for which default values are to be adopted may simply be omitted (as always with namelists). The namelist must always be present, even if the default settings should be used and it might be left empty. Please notice that if an individual parameter of a profile must be changed, that profile must be explicitly appear in the above namelist, even if it relates to the default. Omitting an option above means that the complete defaults for that entry are accepted.

2.9.1 Biodiffusion coefficient profile

There are currently seven different biodiffusion coefficient profiles available:

`const` — constant, discontinuous across z_{Z} [***default***]

$$D^{\mathrm{bt}}(z) = D_0^{\mathrm{bt}}$$

Default value: $D_0^{\mathrm{bt}} = 0.15\,\mathrm{cm^2 yr^{-1}}$.

`lin0z` — linearly decreasing to 0 at z_{Z}, continuous across z_{Z}

$$D^{\mathrm{bt}}(z) = D_0^{\mathrm{bt}} \left(1 - \frac{z}{z_{\mathrm{Z}}}\right)$$

Default value: $D_0^{\mathrm{bt}} = 0.15\,\mathrm{cm^2 yr^{-1}}$.

`linxz` — linearly decreasing down to z_{Z}, discontinuous across z_{Z} unless slope is equal to 1

$$D^{\mathrm{bt}}(z) = D_0^{\mathrm{bt}} \left(1 - \frac{z}{z_{\mathrm{T}}\sigma}\right)$$

Default values: $D_0^{\mathrm{bt}} = 0.15\,\mathrm{cm^2 yr^{-1}}$ and $\sigma = 1$.

`quad0z` — quadratically decreasing to 0 at z_{Z}, continuous across $z_{i_{\mathrm{Z}}}$

$$D^{\mathrm{bt}}(z) = D_0^{\mathrm{bt}} \left(1 - \frac{z}{z_{\mathrm{Z}}}\right)^2$$

Default value: $D_0^{\mathrm{bt}} = 0.15\,\mathrm{cm^2 yr^{-1}}$.

`expdec` — exponentially decreasing, discontinuous across z_{Z}

$$D^{\mathrm{bt}}(z) = D_0^{\mathrm{bt}} \exp(-z/\sigma)$$

Default values: $D_0^{\mathrm{bt}} = 0.15\,\mathrm{cm^2 yr^{-1}}$ and $\sigma = 10\,\mathrm{cm}$.

`gaussn` — gaussian decrease, discontinuous across z_{Z}

$$D^{\mathrm{bt}}(z) = D_0^{\mathrm{bt}} \exp(-(z/\sigma)^2)$$

Default values: $D_0^{\mathrm{bt}} = 0.34\,\mathrm{cm^2 yr^{-1}}$ and $\sigma = 5\,\mathrm{cm}$.

`custom` — use a custom biodiffusion profile.This requires that

- a source file named `bdiffc_custom.F` that contains two subroutines, named `BDIFFC_CUSTOM` (to update the custom profile) and `SETUP_BDIFFC_CUSTOM` (to initialize its parameters) is copied or linked into `src-med/gen/include` after the code generation stage completes and before the library is built (i.e., after `make codegen`, but before `make libmedusa.a`);
- the pre-processor switch `-DBIODIFFUSION_CUSTOM` is used to compile and build `libmedusa.a`, so that `bdiffc_custom.F` is integrated into `MOD_TRANSPORT`.

A template that shows how such a `bdiffc_custom.F` must be set up is provided in `bdiffc_custom.F_template` in `src-med/templates`. Several usable examples (currently `bdiffc_muds.F`, `bdiffc_omexdia.F` and `bdiffc_dhabur.F`) can be found in `src-med/include/transport`. Similarly to `-DGRID_CUSTOM`, `-DBIODIFFUSION_CUSTOM` does not activate the custom profile function. It still needs to be activated with the configuration file.

Details about options and about how to override default parameter values for any of the available profiles can be found in the source code of the module (`mod_transport.F` in `src-med`) and its included subroutine `BDIFFC` (source code in `bdiffc.F` under `src-med/include/transport`).

2.9.2 Bioirrigation coefficient profile

The bioirrigation coefficient can currently be chosen to follow one of three options:

`none` — bioirrigation is neglected [***default***]

$$\alpha(z) = 0$$

`expdec` — exponentially decreasing

$$\alpha(z) = \alpha_0 \exp(-z/\sigma)$$

Default values are: $\alpha_0 = 200\,\mathrm{yr}^{-1}$ and $\sigma = 0.28\,\mathrm{cm}$ (van Cappellen and Wang, 1996), or, alternatively (currently commented out), $\alpha_0 = 240\,\mathrm{yr}^{-1}$ and $\sigma = 4\,\mathrm{cm}$ (Katsev et al., 2007);

`custom` — use a custom profile,similarly to the biodiffusion coefficient profile, to be provided here in a source file called `birric_custom.F`, with the subroutines `BIRRIC_CUSTOM` and `SETUP_BIRRIC_CUSTOM`, in conjunction with the `-DBIOIRRIGATION_CUSTOM` pre-processor switch.

To be consistent with the biodiffusion coefficient and the concept of bioirrigation (irrigation through channels and tubes produced by infaunal activity), the bioirrigation coefficient can only be non-zero for $z \leq z_{\mathrm{Z}}$, and is equal to zero for $z > z_{\mathrm{Z}}$.

Details about options and about how to override default parameter values for any of the available profiles can be found in the source code of the module `mod_transport.F` (in `src-med`) and its included subroutine `BIRRIC` (source code in `birric.F` under `src-med/include/transport`).

2.9.3 Upwind scheme

Select the upwinding scheme: full upwind or exponential fitting (similar to Fiadeiro-Veronis)

`full` — full upwind [***default***]

`expfit` — exponential fitting

There are currently no adaptable parameters for the two upwinding schemes. It is also not possible to use custom schemes.

3 Completing the Picture: Further Steps Towards a Usable Application

The common framework presented above needs to be completed to transform it into a concrete, useful and usable application. First of all, it must be determined which solids, solutes and solute systems have to be considered in the application; the chemical reaction network that describes the diagenetic processes to take into account has to be defined and the chemical equilibria within the solute systems need to be included. There are several additional details that need to be taken care of: for the solutes under consideration, diffusion coefficients are required to complete the transport terms in the solutes' equations; for solids, densities (from which partial specific volumes are derived) need to be specified to evaluate the static volume conservation equation; solubility products may be required for some solids, etc.

Once all the required information has been collected in standardized XML component, chemical reactions and equilibrium description files, the MEDUSA COnfiguration and COde GENeration tool — MEDUSACOCOGEN – then generates the code for the missing modules and subroutines. The most important of the generated source code files are

- `mod_indexparam.F` with the index definitions for the components and `mod_materialcharas.F` with the material characteristics (densities, molar compositions and masses, ...)
- `mod_rreac.F` with the subroutines to evaluate the process reaction rates and their derivatives with respect to the components, and the dependencies `mod_processcontrol.F` with subroutines to initialize and update process parameters, `mod_processdata.F` to hold the parameters and `mod_processsubr.F` to call the rate law library functions
- `mod_store_ncfiles.F` to write and `mod_reac_ncfiles.F` to read the NetCDF results files, together with `mod_netcdfparam.F` for the common parameters
- `mod_equilibcontrol.F` with subroutines to initialize and update equilibrium constants, `mod_equilibdata.F` to hold equilibrium related data and `mod_equilibsubr.F` to evaluate the equilibrium relationships for the requested equilibria
- `mod_chemicalconsts.F` with additional chemical constants and a subroutine to initialize and update them
- `flux2dae.F` to convert the partial differential equation system into a differential algebraic equation system, where the fast interconversion rate terms have been filtered out and the relevant equilibrium relationship introduced.

In addition, a series of templates are generated with optional code include files, host-model interface code blocks, setup routines, run-time configuration files and regular input files. Please refer to the companion *Reference Guide to the Configuration and Code Generation Tool* MEDUSACOCOGEN for details about this part of the model development process.

4 Global compilation options

There are several options that can be selected at the compilation stage (in the `Makefile`). Most of the available options are also listed in the `Makefile`s of common applications.

4.1 Debugging

Debugging can be switched on with the pre-processor switch `-DDEBUG` at compilation. Additional debugging switches can be selected in `debug.h` (in

`src-med`), allowing to restrict debugging to selected subroutines only, or even parts of subroutines. Default is not to generate debugging information.

4.2 Parallel execution with MPI

The Message-Passing Interface related parts of MEDUSA can be activated with the pre-processor switch `-DALLOW_MPI` during compilation. Default is serial execution only.

4.3 Calendar type (days per year)

As mentioned above, the fundamental unit of time in MEDUSA is the year. As MEDUSA is meant to be coupled to various biogeochemical models that use different year lengths, MEDUSA offers three different calendars: Three different calendars are made available (default: `-DCALENDAR_365DAYS`):

- `-DCALENDAR_360DAYS`, where $1\,\mathrm{yr} = 360\,\mathrm{d}$
- `-DCALENDAR_365DAYS`, where $1\,\mathrm{yr} = 365\,\mathrm{d}$
- `-DCALENDAR_365P25DAYS`, where $1\,\mathrm{yr} = 365.25\,\mathrm{d}$

In each case, we adopt $1\,\mathrm{d} = 24\,\mathrm{h}$ and $1\,\mathrm{h} = 3600\,\mathrm{s}$. This information is important for consistent mass balances in MEDUSA and a host model to which it is coupled.

The resulting adaptations for sub-year units (days, seconds, ...) are made in the module `MOD_BASICDATA_MEDUSA` (`mod_basicdata_medusa.F`) and the parameter values for the conversion of units ($\mathrm{cm^2 s^{-1}} \rightarrow \mathrm{m^2 yr^{-1}}$, $\mathrm{s^{-1}} \rightarrow \mathrm{yr^{-1}}$, ...) adapted accordingly.

4.4 Interface options: 1D, 2D or 2Dx2D

The default interface is for a simple sequence of sea-floor grid points in the host model, or equivalently a 1D-array of sediment columns (1D interface). For host model grids organised along more than one dimension, two extensions are offered that can be activated with pre-processor switches during compilation:

- `-DMEDUSA_BASE2D` for a host model grid that is mapped onto a 2D map (the most common situation)
- `-DMEDUSA_BASE2DT2D` for a host model grid that is mapped onto a 2D array of 2D tiles (e.g., MITGCM).

These two switches are mutually exclusive as they select different setup subroutines for `MOD_SEAFLOOR_CENTRAL` and also add specific API subroutines to this module.

4.5 Porewater advection

Porewater advection is neglected in the equations by default. It can be taken into account by using the pre-processor switch `-DALLOW_SOLUTE_ADVECTION` during compilation..

4.6 "Volumeless" solids

Except for solids' colours or production time concentrations, all solids have a finite non zero density, and thus a finite non zero partial specific volume. For compatibility reasons with other models, that commonly assume that the solid phase has a constant density and that do not enforce the static volume conservation equation, it is possible to override this behaviour. This is done by setting the partial specific volumes of all solids to zero, except for that of the main inert solid (typically clay). This latter then plays the role of the bulk solid, and, *de facto* ignoring the integral term in eqn. (6). This *volumeless solids* option can be activated by the pre-processor switch `-DSOLIDS_VOLUMELESS` during compilation..

5 Fine Tuning

MEDUSA offers a series of fine-tuning options, which do nevertheless usually not require any close inspection:

- number and order of solution strategies in `solvsed_onestep.F`: for time dependent simulation experiments, the order `BASIC-SOLUT-PSVOL-WFFLX-RREAC` has proven most efficient; for direct steady-state calculations, an `TIMST` is considered in addition and as last.
- Newton parameters in `solvsedFVFullUpwind.F`: for complex reaction networks, it may be useful to increase the maximum Newton damping exponent to try out (set by the parameter `jp_newton_test_max` in the subroutine `ImplicitTimeStep` in `solvsedFVFullUpwind.F`) and/or the threshold value above which further exponents are only tried out as long as the sum of the squares of the equation residuals is smaller than the previous minimum found (parameter `jp_newton_test_minmax`);

Table 1: Impact of Newton damping parameters on execution time.

$n_{\max}$	$n_{\text{Newt min}}$	$n_{\text{Newt minmax}}$	$n_{\text{Newt max}}$	CPU time	n_{SOLUT}	$n_{\text{globalmax}}$
120	0	4	4	466 s	0	91
120	0	2	4	414 s	0	91
120	0	1	4	369 s	0	83
120	0	0	4	307 s	0	79
120	0	4	5	532 s	0	??
120	0	3	5	473 s	0	??
120	0	2	6	409 s	17	??
240	0	2	6	411 s	0	227
120	1	4	4	1540 s	0	120
120	1	2	4	1424 s	0	120
120	-1	0	4	413 s	6	120+41

Obtained for the HBLOCH_FILE application (unpublished), with 3476 columns, 21-point grids and the following composition: solutes – CO_2, HCO_3^-, CO_3, O_2, H_4SiO_4; solids – clay, calcite, aragonite, opal and organic matter; equilibria: carbonate; processes – calcite dissolution, aragonite dissolution, opal dissolution, and oxic respiration of organic matter. Results for 10 time-steps of 500 yr each.

- with MPI: global or per-process NETCDF files via
  `LOGICAL, PARAMETER :: lp_exeproc_singleproc_nc`
  in `mod_execontrol_medusa.F`
- the range of logical unit numbers controlled by `MOD_LOGUNITS` (source file `src-med/mod_logunits.F`) can be changed to avoid conflicts between program units of the host program and of MEDUSA (e.g., increase the value of `min_valid` in `MOD_LOGUNITS`).

A Utilities

Several utility codes have been developed in the course of the MEDUSA development. There are two types of these:

- Fortran modules for general purpose tasks (they are independent on the rest of MEDUSA and can be used without pre-requirements);
- utility programs for post-processing MEDUSA output, such as remapping routines to convert between the "columns on a string" layout

adopted for storing results in the NETCDF files and a closer-to-real-world X-Y or longitude-latitude mapping, vertical integration of solid and solute concentrations, extractions of synthetic sediment cores, etc.

A.1 General purpose Fortran modules

A.1.1 Linear system solver

The module `MOD_GAUSS` (source file `mod_gauss.F` in `src-med`) provides a general solver for linear systems callable as `GM_DGESV`. Its argument list is entirely compatible with that of `LA_DGESV` from LAPACK95, which can thus be used instead. Please notice though that unlike `LA_DGESV`, `GM_DGESV` calls upon a factorization routine that uses a column pivot search with implicit row scaling, as advocated by (Engeln-Müllges and Uhlig, 1996, p.73).

A.1.2 Logical file unit management

The module `MOD_LOGUNITS` (source file `mod_logunits.F` in `src-med`) provides a means to automatically detect non-attached logical unit numbers for I/O operations. A free logical unit number can be searched for and reserved with the function `RESERVE_LOGUNIT(iunit)`; a file attached to the logical unit number `iunit` can be closed and released with the function `FREE_LOGUNIT(iunit)`.

`RESERVE_LOGUNIT` only scans above a set minimum logical unit number, set by `min_valid` in the module but is not limited above; `FREE_LOGUNIT` also only processes with logical unit numbers in that same range.

A.1.3 Miscellaneous

A series of general purpose function and subroutine programs have been developed for various purposes. They are collected in two modules:

1. `uti/mod_uticommon.F90`
    - EXPAND_LIST – expand a list of integer values defined by a list of ranges and individual values in a string (e.g., `'1-4,6,8-9'`) to an `INTEGER` array explicitly holding all the values included in the ranges;
    - CONDENSE_LIST – convert an `INTEGER` array of values to a condensed list of ranges and individual values;
    - DELIMIT_STRING_TOKENS – determine the starting positions and lengths of elements in a `CHARACTER` string delimited by a set of separation characters;

- TESTDIR – inquiry function to determine whether a given path is a directory

2. `src-mcg/mod_utilities.F90`

   - UPCASE – convert a `CHARACTER` string to all upper case;
   - LOWCASE – convert a `CHARACTER` string to all lower case;
   - SPC2UNDERSCORE – convert space characters in a `CHARACTER` string to underscore (_) characters;
   - EXPAND_TOKEN, EXPAND_2TOKEN, EXPAND_3TOKEN – expand given substrings (tokens) of a `CHARACTER` string by substituting them with requested character strings (resp. one, two or three at a time);
   - L_STR_EQ_DBL – inquiry function to test whether the contents of a `CHARACTER` string are equal to a given `DOUBLE PRECISION`;
   - INDEX_COPIES_IN_ARRAY – detect duplicates in a given array of `CHARACTER` strings and prepare an `INTEGER` index array of the same size as the `CHARACTER` array and that provides for each element of the `CHARACTER` array the position where it was first found in the array;
   - COUNT_COPIES_IN_ARRAY – for an array of `CHARACTER` strings, prepare an `INTEGER` array of the same size that provides the number of occurrences of the value of each element of the `CHARACTER` array;
   - DELIMIT_STRING_TOKENS – determine the starting positions and lengths of elements in a `CHARACTER` string delimited by a set of separation characters;
   - WRITE_CONDENSED_DBLE – write a condensed version (i.e., with trailing zeroes in the mantissa removed) either to a file (if a logical unit number – type `INTEGER` – is given as the first dummy argument) or a `CHARACTER` string (if the first argument is of type `CHARACTER`).

These utilities are completely independent of MEDUSA and can be used elsewhere.

A.2 Post-processing utilities

A collection of post-processing utilities is provided in the `uti` directory. Unlike the code configuration and generation utility executable, which must stay

in `src-mcg`, they can be moved to a central location, such as `/usr/local/bin`, as they do not depend on content in the MEDUSA source collection.

A.2.1 MEDINTEGRALEV

MEDINTEGRALEV integrates all vertical concentration profiles in a NETCDF results file produced by MEDUSA (i.e., concentration arrays that include a `lev` dimension) over the REACLAY realm. The source code of MEDINTEGRALEV is in `medintegralev.F90` and the executable can be built with

```
make medintegralev
```

The resulting NETCDF file does not include the dimension `lev` any longer and contains all the arrays of the input file (integrated along the vertical and with the dimension `lev` removed) and copies of all the variables of the input file that do not include `lev` among their dimensions. MEDINTEGRALEV can be used in interactive and non-interactive modes. The non-interactive usage requires the presence of a namelist file called `medintegralev.nml`. If such a file is present in the working directory, its contents are used without further questions; if not, the program switches to interactive processing. Upon successful completion, the finally adopted inputs are stored to be used as defaults for subsequent interactive executions, or as a template for controlling subsequent non-interactive executions. It is recommended to use it first in interactive mode and switch to non-interactive mode in a second stage if suitable.

A.2.2 MEDCOL2XY and MEDCOL2XY_2DT2D

MEDCOL2XY remaps results obtained with MEDUSA configured with a 2D interface from the "columns on a string" organisation to an X-Y distribution (typically longitude-latitude); MEDCOL2XY_2DT2D does the same for results obtained with the 2DT2D interface version. The source codes are provided in `medcol2xy.F90` and `medcol2xy_2DT2D.F90`, resp. and the executables can be built with

```
make medcol2xy
make medcol2xy_2DT2D
```

Both can be used in interactive and non-interactive modes. Non-interactive usage requires the presence of a namelist file called `medcol2xy.nml` (resp. `medcol2xy_2DT2D.nml`). If such a file is present in the working directory, its contents are used without further questions; if not, the program switches to interactive processing. Upon successful completion, the adopted inputs are stored to be used as defaults for subsequent interactive executions, or

as a template for controlling subsequent non-interactive executions. It is recommended to simply use the adequate version first in interactive mode.

A.2.3 COLUMN_EXTRACT2CSV

NETCDF was chosen as the file format to store the results produced by MEDUSA because of the portability it offers between different platforms and operating systems and because of the flexible possibilities it offers to include metadata. NETCDF files can be processed with a large diversity of data processing software, but popular spreadsheet programs generally lack support for it.[1]

We therefore developed COLUMN_EXTRACT2CSV which extracts results for a selection of sediment columns from any NETDCF results file produced by MEDUSA and stores these in CSV format, suitable for processing in any spreadsheet program. The source code is in `column_extract2csv.F90` and the executable can be built with

```
make column_extract2csv
```

COLUMN_EXTRACT2CSV can again be used in interactive and non-interactive modes. The non-interactive usage requires the presence of a namelist file called `column_extract2csv.nml`. If such a file is present in the working directory, its contents are used without further questions; if not, the program switches to interactive processing. Upon successful completion, the adopted inputs are stored to be used as defaults for subsequent interactive executions, or as a template for controlling subsequent non-interactive executions. It is recommended to use it first in interactive mode and switch to non-interactive mode in a second stage if suitable.

A.2.4 SEDCORE_EXTRACT

The format of the SEDCORE NETCDF file produced by MEDUSA is rather convoluted, as the information is stored as a bulk sequence of "events" (burial or unurial of layers of a given characteristics) registered in time for the various columns in the model setup.

SEDCORE_EXTRACT extracts such events for a selection of sediment cores and stores them in individual NETCDF files (one per core). The source code is in `sedcore_extract.F90` and the executable can be built with

[1] Users of 32-bit versions of MS EXCEL should check out the excellent add-in NETCDF4EXCEL. If only someone could port this fine piece of software to the now more common 64-bit version of MS EXCEL or even to the open-source alternative OPENOFFICE/LIBREOFFICE...

```
make sedcore_extract
```

SEDCORE_EXTRACT can again be used in interactive and non-interactive modes. The non-interactive usage requires the presence of a namelist file called `sedcore_extract.nml`. If such a file is present in the working directory, its contents are used without further questions; if not, the program switches to interactive processing. Upon successful completion, the adopted inputs are stored to be used as defaults for subsequent interactive executions, or as a template for controlling subsequent non-interactive executions. Please call SEDCORE_EXTRACT first in interactive mode and switch to non-interactive mode in a second stage if suitable.

References

B. P. Boudreau. *Diagenetic Models and Their Implementation*. Springer-Verlag, Berlin, 1997. doi: 10.1007/978-3-642-60421-8.

B. P. Boudreau. On the equivalence of non-local and radial-diffusion models for porewater irrigation. *J. Mar. Res.*, 42(3):731–735, 1984. doi: 10.1357/002224084788505924.

B. P. Boudreau. The mathematics of tracer mixing in sediments : I. Spatially-dependent, diffusive mixing. *Am. J. Sci.*, 286(3):161–198, 1986. doi: 10.2475/ajs.286.3.161.

G. Engeln-Müllges and F. Uhlig. *Numerical Algorithms with Fortran*. Springer-Verlag, Berlin, 1996. ISBN 3-540-60529-0.

S. Katsev, G. Chaillou, B. Sundby, and A. Mucci. Effects of progressive oxygen depletion on sediment diagenesis and fluxes: A model for the lower St. Lawrence River Estuary. *Limnol. Oceanogr.*, 52(6):2555–2568, 2007. doi: 10.4319/lo.2007.52.6.2555.

S. Mulsow, B. P. Boudreau, and J. N. Smith. Bioturbation and porosity gradients. *Limnol. Oceanogr.*, 43(1):1–9, 1998. doi: 10.4319/lo.1998.43.1.000I.

P. van Cappellen and Y. Wang. Cycling of iron and manganese in surface sediments : A general theory for the coupled transport and reaction of carbon, oxygen, nitrogen, sulfur, iron, and manganese. *Am. J. Sci.*, 296 (3):197–243, 1996. doi: 10.2475/ajs.296.3.197.

Index