Running the coupled sediment – water column model ERGOM SED 1.0

Hagen Radtke*
Leibniz Institute for Baltic Sea Research Warnemünde (IOW)

April 19, 2018

*hagen.radtke@io-warnemuende.de

About this document

This document practically describes the application of the model ERGOM SED 1.0, a combined model for water and sediment biogeochemistry. It describes step-by-step how to run the model in a one-dimensional setup, that is, for a vertical water column and the underlying sediment.

This document only covers the technical aspects of the model. The scientific concept behind the model, the model state variables and processes included, are described in detail in the following publication:

Radtke, H., Lipka, M., Bunke, D., Morys, C., Cahill, B., Böttcher, M.E., Forster, S., Leipe, T., and Neumann, T.: Ecological ReGional Ocean Model with vertically resolved sediments (ERGOM SED 1.0): Coupling benthic and pelagic biogeochemistry of the south-western Baltic Sea. Journal of Geoscientific Model Development, submitted.

This guide describes how to run the same experiments as described in the paper. The preceding calibration and model optimisation is not covered in this document. Also, the required calibration for applying this model to a new site is not described in detail here. We only give a coarse overview on how it is technically possible to optimise this model and leave the choice of the best calibration technique to the user.

Acknowledgements

This study is embedded in the KÜNO Project SECOS (03F0666A) funded by the German Federal Ministry for Education and Research (BMBF).

Table of Contents

1 Overview

1.1 The ERGOM model

The model ERGOM (Ecologial ReGional Ocean Model) is a biogeochemical model which describes the marine nitrogen, phosphorus and carbon cycle. It has been created at Leibniz Institute for Baltic Sea Research Warnemuende (IOW) (Neumann, 2000) and has been continuously developed since then. Today, it is used by several institutes around the Baltic Sea in different version and applied to several marine ecosystems.

The development philosophy at IOW is to keep the scientific content of the model separate from its numeric implementation. For this purpose, the model equations are described in a formal way as text files. These define the model constants, the model state variables and the processes transforming them. In this way, it is easy to modify or extend the model.

An additional advantage is that we can use the same biogeochemistry model in different physical host models. In this example, we take advantage of this functionality by the use of a rather simple one-dimensional model written in Pascal. This model setup aims at describing biogeochemical processes in the sediment at a single site as a reaction to water-column forcing.

A thorough discussion on the biogeochemical and physical processes included in the model is not given in this document, but rather in a complimentary scientific article, Radtke et al. (2018). We refer to this paper for the scientific discussion of the model and concentrate on the technical aspects of running the model in this user's guide.

1.2 Steps to take

The steps to run the model include

1. generating the model code from the text files,
2. compiling the model code to an executable,
3. providing the required input files and
4. running the model.

For the example sites in the south-western Baltic Sea, all of the steps 1-3 are already prepared for a Windows environment. So if you just wish to run the model itself, you can immediately start with Step 4, or begin with any of the other steps. These will be described in detail in the following sections.

1.3 Obtaining the model

Go to `https://github.com/hagenradtke/ERGOM-SED-1D` to download the model. Your download will consist of the following directories:

directory	contents	described in
`textfiles/`	formal description of the model state variables and processes	Section 2
`code_templates/`	a code template for the Pascal 1-d model	Section 2
`cgt/`	the "code generation tool" `cgt` required to create the Pascal code, as well as a specific editor `cgt_edit` for convenient editing of the text files, both as binaries and source code	Section 2
`finished_code/`	Pascal code files required to compile the model, including the one created from the two ingredients described above	Section 3
`stations/`	directories with all required input files to run the model for the seven stations described in the article, (Radtke et al., 2018)	Section 4

2 Creating the model code

To generate the model source code, you need the code generation tool `cgt`. If you want to skip this step and use the existing source code, please continue in Section 3.

2.1 Obtaining the Code Generation Tool (CGT)

A binary version of the code generation tool is supplied with this model. It can be found in the subfolder `cgt/` and runs on Windows and most recent Linux versions. The following files can be found there:

file	description
`cgt/cgt`	Linux 64-bit binary of the code generation tool
`cgt/cgt_edit`	Linux 64-bit binary of the editor for the text files describing the model equations
`cgt/cgt.exe`	Windows binary of the code generation tool
`cgt/cgt_edit.exe`	Windows binary of the editor for the text files describing the model equations

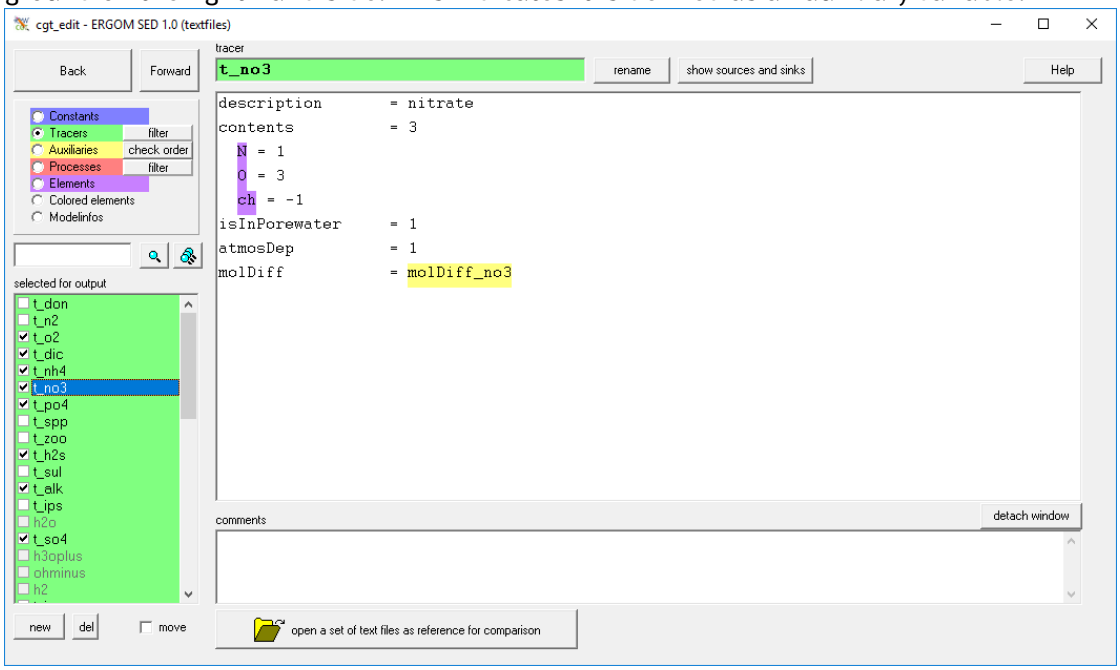
Source code and updated versions of `cgt` can be downloaded from `www.ergom.net`. Also, a detailed user's guide is available there.

2.2 Modifying process equations and model output – cgt_edit

The process equations are formulated in a formal way in text files. These text files can be found in the subdirectory `textfiles`. We will give a short example on how these can be explored and modified. So please make sure you have write access to the files.

1. Run `cgt/cgt_edit` (or `cgt/cgt_edit.exe` under Windows).
2. Click "load existing model".
3. Browse to subfolder `textfiles/` and open `modelinfos.txt`. You will see a list of the model constants in blue, and if you click them, they will be shown at the right-hand side.
4. Click at "tracers" on the top left. Now the model state variables will be shown.

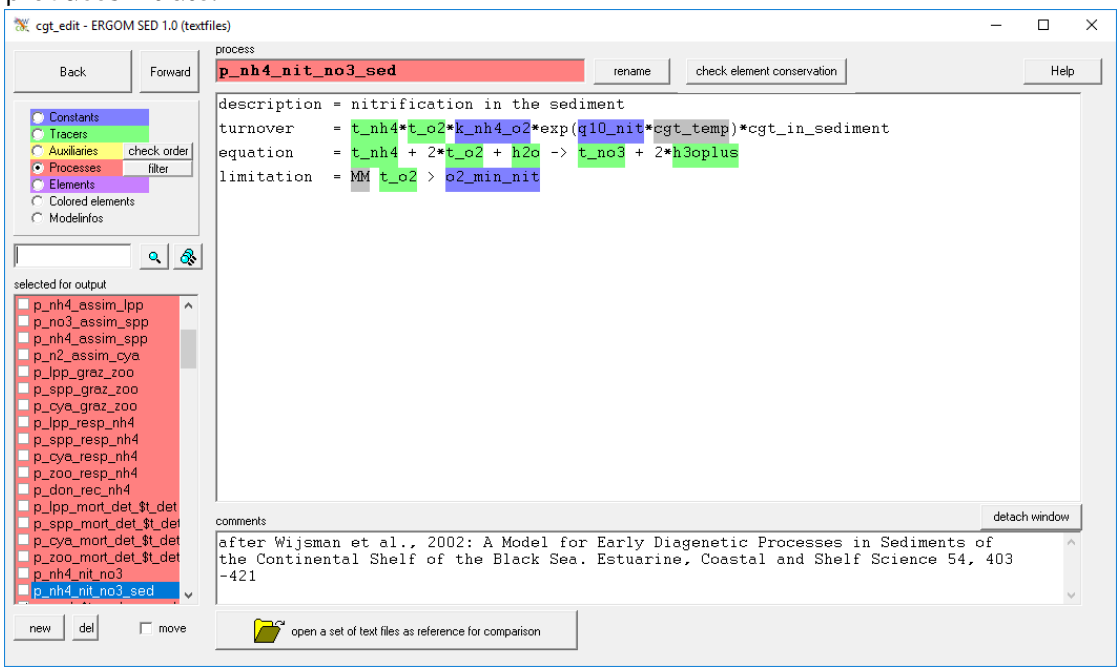
5. Click at tracer `t_no3` in the left column. Its details will be shown on the right.
6. Note the term `molDiff_no3` on the right-hand side is marked by a yellow background on the right-hand side. This indicates it is defined as an auxiliary variable.



7. Right-click `molDiff_no3` to see its definition. Then click “back” at the top left to return.
8. Click “show sources and sinks” in the top row to see which processes consume or produce dissolved nitrate. A window will open showing these processes.

9. Double-click on process `p_nh4_nit_no3_sed` (which is nitrification in the sediments). You will see two important entries on the right:
   - `turnover` – defines the speed of the process
   - `equation` – defines the sources and sinks of the process like in a chemical process equation

   Note that `t_no3` appears to the right of the arrow (`->`) which means this process produces nitrate.



10. Click at “help” at the top right. This will explain what the entries in the large white text box mean.
11. At the end of the line starting with `turnover=...`, add “`*2`”. Then click the “save” button at the top left. You have now speeded up the nitrification in the sediments by a factor of two.
12. You may wish to undo that by deleting the “`*2`” again and clicking “save” once more.

Very good! You have just learnt the basics on how our ERGOM model is structured and how to modify it. Please note that the “save” actions take place immediately and modify the loaded set of text files. So please make sure you have a backup before playing around.

Instead of using our editor `cgt_edit`, you can edit the textfiles with any editor of your choice.

2.3 Creating the model code – cgt

The model definition in the text files will need to be translated into model source code which a compiler (or interpreter) can understand. This is the task of the code generation tool `cgt`.

It will start from a "code template" which looks like this:

```
<constants>
  ! <description>:
  <name> = <value>
</constants>
...
```

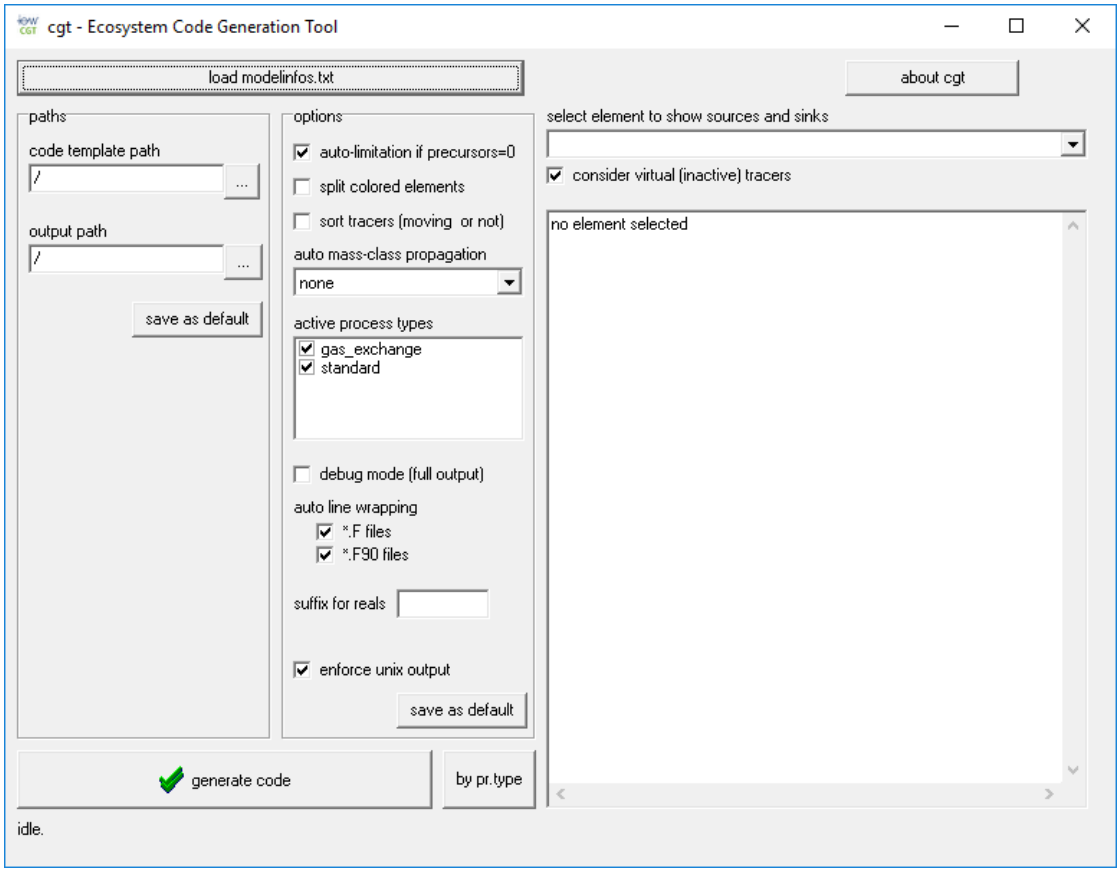
and will produce a code which looks like the following:

```
  ! critical shear stress for sediment erosion [N/m2]:
  critical_stress = 0.016
  ! seed concentration for diazotroph cyanobacteria [mol/kg]:
  cya0            = 0.0045e-6
  ! DIN half saturation constant for large-cell phytoplankton growth [mol/kg]:
  din_min_lpp     = 1.125e-6
...
```

To do so, it just fills in the information given in the text files.

To create the model code on your own, please follow these instructions:

1. Run `cgt/cgt` (or `cgt/cgt.exe` under Windows).
2. Click "load modelinfos.txt".
3. Browse to subfolder `textfiles/` and open `modelinfos.txt`.
4. When asked whether you want to load an add-on, click "no". You have now loaded the formal model description in the text files.



5. In the first text box (“code template path”), enter the full path of the subdirectory `code_templates/`. Alternatively, click at the dots next to the text box and select the file `cgt_1d_model.pas` in this subdirectory.
6. In the second text box, enter the full path of the subdirectory `finished_code/`. Alternatively, click at the dots next to the text box and select to save this file in this subdirectory.
7. Click “save as default” in the left column so `cgt` will remember these paths the next time you load this set of text files.
8. Click “generate code” at the bottom so the model source code will be produced. In our case, it is just one file, `cgt_1d_model.pas`. If there were more “code templates” in the code template directory, all of these would be filled in to become finished code files.

So, in this section, you have created the file `finished_code/cgt_1d_model.pas` which contains the important parts of the model source code. In the next section, you will compile it to create the running model.

3 Compiling the model

This section describes how to generate a model executable from the model source code. If you want to skip this step and use the existing binary, which will work under Windows but not typically under Linux, please continue in Section 4.

3.1 Obtaining FreePascal Lazarus

Since the model is written in Pascal, you may require the Free Pascal Compiler. It is available for both Windows and Linux and comes with a nice IDE (integrated development environment) called Lazarus which you should install.

If you are working on Linux, there is possibly already a Lazarus package in your distribution which you can install via your package manager. If you are working on Windows or on a Linux environment which accepts RPM or DEB packages, you can just download and install from this location:
`https://www.lazarus-ide.org/index.php?page=downloads`
If you install the linux packages, you can do this in the following order:

1. `fpc`
2. `fpc-src`
3. `lazarus`

Should you encounter problems or have an unsupported Linux distribution, detailed installation instructions can be found here:
`http://wiki.freepascal.org/Installing_Lazarus`

3.2 Installing the NetCDF library

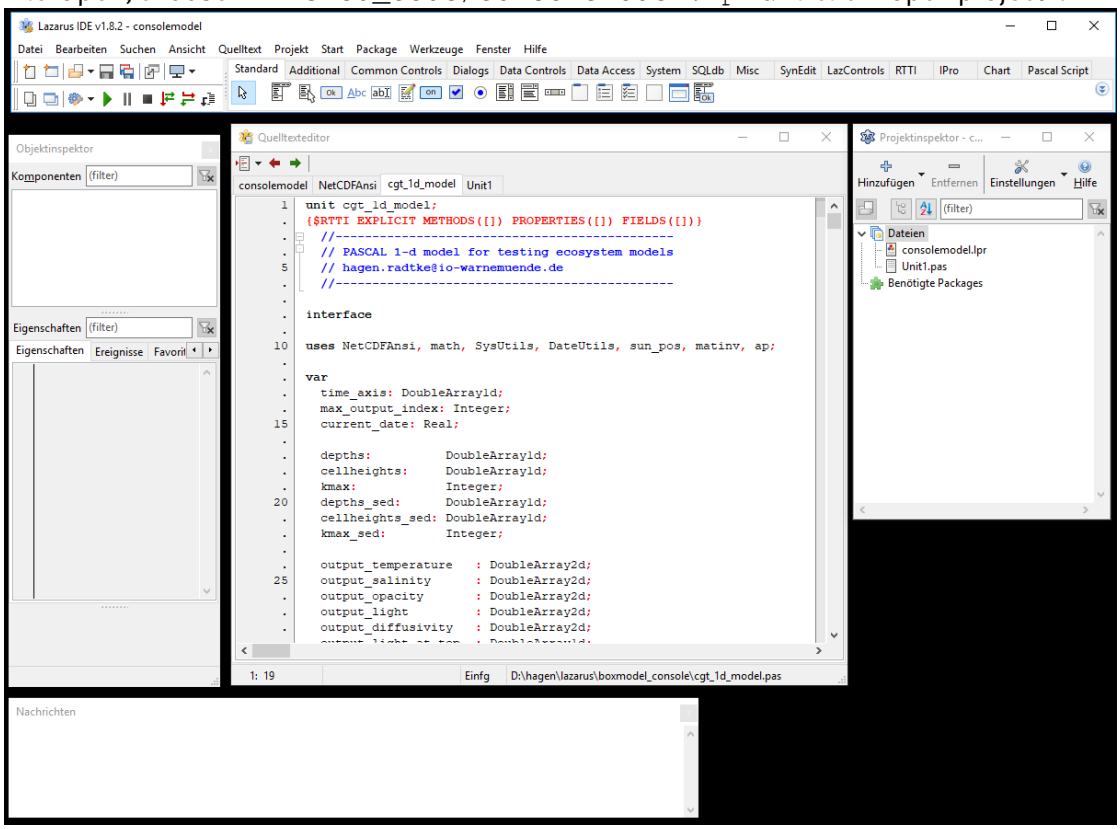
Model data output is stored in the NetCDF format, which requires the NetCDF library. Unless NetCDF4 already comes with your linux distribution, please download the 64-bit NetCDF4 version from
`https://www.unidata.ucar.edu/downloads/netcdf/index.jsp`
and install it on your system.

If you are working on Windows, copy the file `netcdf.dll` which can be found under `Programs/netCDF ?.?.?/bin`, to the subdirectory `finished_code`.

3.3 Compiling the model

1. Start the Lazarus IDE (under Linux by typing `startlazarus`).

2. File-Open, choose `finished_code/consolemodel.lpi` and click “open project”.



3. Now we need to make sure the compiler will find the NetCDF library. Under Windows, that’s already fine because we already copied it to the correct subdirectory. Under Linux, you need to find out where your netCDF library `libnetcdf.so` is stored. For example, this might be under `/usr/lib64`. Then you need to tell the compiler to include this search path:
   - Project - Project Settings - Compiler Settings - Paths
   - Under “Libraries (-Fl)”, enter the path to the netCDF library.
   - Click OK.
4. Click Run-Compile. Compilation should succeed and you should get a note about that.

Now you should have a freshly generated executable file `finished_code/consolemodel` (or `consolemodel.exe` under Windows). Copy this file to the subdirectory `stations/station_AB` which is the working directory to run the simulation for the station Arkona Basin.

4 Running the model

This section describes how to (a) prepare a run directory for an ERGOM-SED 1-d simulation and (b) run the model.

4.1 Run directory

In order to run the model, you need to put a few things together in a run directory.

file/subdirectory	details	described in
`consolemodel(.exe)`	model executable	
`physics/`	physical boundary values to drive the 1-d model	Section 4.2
`init/`	initial conditions for the model	Section 4.3
`options_run.txt`	settings file for e.g. time step and length of run	Section 4.4
`values_of_constants.txt`	optional, text file containing new values to override model constants	Section 4.4
`netcdf.dll`	the netCDF library (Windows only)	Section 4.5

If you want to run an existing setup, e.g. for one of the example stations, you can skip Sections 4.2 to 4.4 and immediately proceed with Section 4.5.

4.2 Preparing physical forcing

Text files defining physical boundary conditions for the 1-d biogeochemical model are stored in the subdirectory `physics/`. The following two tables describe their content:

Files for water column and fluff layer

file	contains	unit	format
`cellheights.txt`	water column cell heights top to bottom	m	Type 1
`diffusivity.txt`	water column turbulent diffusivity	$m^2\ s^{-1}$	Type 3
`light_at_top.txt`	downward flux of shortwave radiation directly below the water surface	$W\ m^{-2}$	Type 2
`opacity_water.txt`	light attenuation of the water and all its components not represented in the biogeochemical model	m^{-1}	Type 3
`salinity.txt`	salinity	g/kg	Type 3
`temperature.txt`	in-situ temperature	°C	Type 3
`bottom_stress.txt`	bottom shear stress (relevant for fluff layer erosion)	$N\ m^{-2}$	Type 2

Files for sediment and pore water

file	contains	unit	format
`sed_cellheights.txt`	sediment model cell heights top to bottom	m	Type 1
`sed_depth_bioturbation.txt`	depths of full bioturbation, depth scale of bioturbative diffusivity decay with depth, lower depth limit of bioturbation	m	Type 4
`sed_diffusivity_porewater.txt`	maximum effective diffusivity of bioirrigation	$m^2\ s^{-1}$	Type 2
`sed_diffusivity_solids.txt`	maximum effective diffusivity of bioturbation	$m^2\ s^{-1}$	Type 2
`sed_inert_deposition.txt`	volume flux of solid bioinert material	$m\ s^{-1}$	Type 2
`sed_inert_ratio.txt`	volume fraction of solids (=1-porosity)	1	Type 1

If none of these files are given, the model will run without the vertically resolved sediment model component.

The format types mean the following:

- **Type 1**
  Each row contains one real value (1.23e-45)
  No header line.
- **Type 2**
  Each row contains the following fields, separated by one or more spaces:
  `year month day decimalhour value`
  year, month and day = integer values (123) / decimalyear and value = real values (1.23e-45)
  No header line.
  The date specifies a start date from when on the value should be used, so a single line can be used to prescribe constant values over time.
- **Type 3**
  Each row contains the following fields, separated by one or more spaces:
  `year month day decimalhour value1 value2 ... valueN`
  year, month and day = integer values (123) / decimalyear and value1 ... valueN = real values (1.23e-45)
  value1 = value at the top, valueN = value at the bottom
  No header line.
  The date specifies a start date from when on the values should be used, so a single line can be used to prescribe constant values over time.
- **Type 4**
  Contains exactly 6 lines:

```
% depth of full bioturbation
 0.04
% depth scale for exponential decay of bioturbation
 0.07
% depth where bioturbation is set to zero
 0.11
```

  where lines 2, 4 and 6 may obviously be changed to the desired value.

4.3 Preparing initial files

Each of the tracers needs to have initial values specified. They may be given in the text files by specifying
`initValue = 123.45e-06`
and
`useInitValue = 1`
where obviously the value can be changed.

If this is not given, the model requires for each tracer

- one text file `???.txt` for the initial values in water column or fluff layer, and
- a second text file `sed_???.txt` for the initial values in sediment or pore water, unless the tracer has the properties
  `vertLoc=WAT`
  and
  `isInPorewater=0`
  which means it occurs in the water column, but not in the pore water.

Here, `???` stands for the name of the tracer.

The text files are formatted according to Type 2 (see previous section). The number of lines per file shall be …

- equal to the number of grid cells given in `physics/cellheights.txt` for the first file `???.txt`, unless
- the tracer has property `vertLoc=SED` meaning it occurs in fluff layer and sediment only, in this case give just one value for the first file `???.txt`, and
- equal to the number of grid cells given in `physics/sed_cellheights.txt` for the second file `sed_???.txt`.

Units given in the initial files depend on the units used for formulation of the model in the text files. The examples use the following units:

- mol kg^{-1} for water column and pore water concentrations of dissolved/suspended matter (`vertLoc=WAT`) and
- mol m^{-2}

  for fluff layer and sediment concentrations of particulate material (`vertLoc=SED`)

4.4 Settings for the run

Settings for the model can be specified in two files:

4.4.1 General settings

General settings for the model can be specified in the file `options_run.txt`. Please copy it from an example directory and just modify the values. The following settings can be changed:

option	accepted values
start date	format dd.mm.yyyy
end date	format dd.mm.yyyy
repeated runs	integer value indicating how many cycles from start date to end date shall be run, each with the given forcing
time step [days]	real value describing biogeochemical time step, use "." as decimal separator
output interval [days]	real value describing the step width on the time axis in the output files, use "." as decimal separator
Runge-Kutta depth	a value of 1 means the use of a positive Euler-Forward (pEf) or positive Euler-Forward Patankar (pEfP) scheme, a value of 2 means a positive Runge-Kutta (pRK) or positive Runge-Kutta Patankar (pRKP) scheme. For details on these numerical schemes, see Radtke and Burchard (2015). For a vertically resolved sediment model, only the value 1 is accepted.

4.4.2 Overriding of model constants

Optionally, you can provide a file `values_of_constants.txt` which overrides the predefined values of the constants.

In the text files where you define constants, you can add the lines
`minval=...`
and
`maxval=...`
which then means their value given by
`value=...`
can be overridden. To do so, supply a file `values_of_constants.txt` which gives the values for all of these constants in the order in which they were defined. In each line, you can add a comment after this value, the comment should start with "!".

4.5 Making sure the netCDF library is found

The model needs to load the netCDF library to write the results to the output files. So, make sure you have netCDF installed, see Section 3.2.

Under Windows, it is sufficient to copy the file `netcdf.dll` into the working directory, e.g. `stations/station_AB/`.

Under Linux, you will need to add the directory which contains `libnetcdf.so.7` to the environment variable `LD_LIBRARY_PATH`. For example, under openSUSE this can be done by the following command:
`export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/usr/lib64/`
if `/usr/lib64/` is the path to the netCDF library.

4.6 Executing the model

Now, just run the file `consolemodel` (or `consolemodel.exe` under Windows). It should write out the dates of the time steps and produce output files.

5 Model output

Model output consists of two parts, the model results and the restart files from which you could continue the present run.

5.1 Results files

There are two results files produced, both of which are in the netCDF format:

file	description
`results_1d_model.nc`	results in the water column and fluff layer
`results_1d_model_sed.nc`	results in the sediment

Each of these files gives the values of selected state variables, auxiliary variables and process rates on a z-t-grid. The file `results_1d_model.nc` contains water column and fluff layer results, while `results_1d_model_sed.nc` contains sediment and pore water results.

To select a tracer, auxiliary variable or process for output, you have to add
`isOutput=1`
to its definition in the text files. In the editor `cgt_edit`, this is done by clicking the check box next to its name in the list.

5.2 Restart files

After the run has finished, the model will create a subdirectory `final/` in its run directory. Therein, the values of the tracers at the end of the run are given. So, this directory can be used as an initial condidtion for a model run continuing from here. Please be aware, however, that state variables which have `initValue=...` and `useInitValue=1` will still be initialized these values then, so their values at the end of the run will be lost.

6 Model calibration

When applying the model to a different site, a recalibration of the model will become necessary. This calibration shall be done by the modeller. Here we give just two small hints.

6.1 Overriding model constants

The overriding of model constants described in Section 4.4.2 is a good way to tune the model without having to compile it again and again.

6.2 Automatic calibration

Instead of a manual calibration, an automatic optimisation of the model might be quicker. The fact that the model runs from the console eases the automation, which will typically consist of four steps per optimisation cycle:

1. writing a set of values to `values_of_constants.txt` (and possibly to other files in `init/` or `physics/`)
2. running the model
3. analysing the output, comparing to measurements and quantifying the model error
4. obtaining an optimized set of values

Running several instances of the model in different run directories in parallel can speed up the process. Each model writes a file `finished.txt` when its run has finished, and you can check for the existence of this file before you continue.

References

Neumann, T. (2000). Towards a 3d-ecosystem model of the Baltic Sea. *Journal of Marine Systems*, 25(3–4):405–419.

Radtke, H. and Burchard, H. (2015). A positive and multi-element conserving time stepping scheme for biogeochemical processes in marine ecosystem models. *Ocean Modelling*, 85:32–41.

Radtke, H., Lipka, M., Bunke, D., Morys, C., Cahill, B., Böttcher, M. E., Forster, S., Leipe, T., and Neumann, T. (2018). Ecological ReGional Ocean Model with vertically resolved sediments (ERGOM SED 1.0): Coupling benthic and pelagic biogeochemistry of the southwestern Baltic Sea. In preparation.