SHYFEM

Finite Element Model for Coastal Seas

User Manual

The SHYFEM Group
Georg Umgiesser
Oceanography, ISMAR-CNR
Arsenale Tesa 104, Castello 2737/F
30122 Venezia, Italy

georg.umgiesser@ismar.cnr.it

Version 7.5.71

July 22, 2023

Contents

Disclaimer

Comments and additions should be sent to the author:

```
Georg Umgiesser
Oceanography, ISMAR-CNR
Arsenale Tesa 104, Castello 2737/F
30122 Venezia
Italy

Tel.   : ++39-041-2407943
Fax    : ++39-041-2407940
E-Mail : georg.umgiesser@ismar.cnr.it
```

Chapter 1

Overview

1.1 What is it

The finite element program SHYFEM is a program package that can be used to resolve the hydrodynamic equations in lagoons, seas, estuaries and lakes. The program uses finite elements for the resolution of the hydrodynamic equations. These finite elements, together with an effective semi-implicit time resolution algorithm, makes this program especially suitable for applications in areas with a complicated geometry and bathymetry.
The program SHYFEM resolves the depth integrated shallow water equations and can use both a two- and a three-dimensional formulation, depending on the user's needs.
Finite elements are well adapted to problems dealing with complex bathymetric situations and geometries. The finite element method has an advantage over other methods (e.g., finite differences) because it allows more flexibility with its subdivision of the system in triangles varying in form and size. This flexibility can be used also in situations where it is not desired to have uniform resolution of the whole basin, but where a focus in resolution is needed only in some parts of the area.
It is possible to simulate shallow water flats, i.e., tidal marshes that in a tidal cycle may be covered with water during high tide and then fall dry during ebb tide. This phenomenon is handled by the model in a mass conserving way.
Finite element methods have been introduced into hydrodynamics since 1973 and have been extensively applied to shallow water equations by numerous authors [9, 18, 11, 10, 12].
The model presented here [19, 20] uses the mathematical formulation of the semi-implicit algorithm that decouples the solution of the water levels and velocity components from each other leading to smaller systems to solve. Models of this type have been presented from 1971 on by many authors [13, 4, 1].

1.2 Where and how to get it

1.2.1 Where to get it

You can can the latest and older versions of SHYFEM both as files on Google Drive or on GitHub. From Google Drive you can download the latest and older version of the model. Please go to https://drive.google.com/open?id=0B742mznAzyDPbGF2em5NMjZYdHc and download the version you would like to install on your computer. Normally this will be the last available version.
You can also download the versions from GitHub. Please go to the GitHub website of the SHYFEM model https://github.com/SHYFEM-model/shyfem and navigate to releases.

From there you either can visualize the releases or the tags (versions) of the model and download them. Please see below for the difference between tags and releases.
If you are a developer then you should really install the `git` versioning system that will give you direct access to the latest versions of `SHYFEM`. Please see below how to do this.

1.2.2 Details

Development of `SHYFEM` is happening on `github`. Here is is some information on how the various releases are are managed and where to download the latest version.
In github you can always find the latest version of `SHYFEM`. There are various types of versions, which will be explained here below.

- commit: commits are the smallest change in the code base. Every time some changes have been carried out on the code this change will be committed to the repository. You can see all the commits of the code by typing `git-tags`. This shows you all the commits and also the tags, which are explained here below.
- version: the name version is just a shortcut for an existing tag or release. It has a version number that can be used to refer to a specific tag or release. A commit has no version number and can therefore not be identified in this way.
- tag: tags are like commits, but a version number is given to them. This means that these tags are more stable than simple commits. It is always advisable to download tags in order to be able to easily refer to the version number of `SHYFEM`.
- release: releases are nothing else than tags, but a name is also given to this tag. This means that releases should be even more stable than tags or commits. If you do not need a bleeding edge version than these are the versions that you should download.

You can find commits and tags with the command `gittags`. The output of this command will give you the latest commits and tags (if applied). In order to see releases you will have to go to the github web page (`https://github.com/SHYFEM-model/shyfem`). Click on `releases`, which takes you directly to page where you can see all the versions, both as releases or tags. From there you can directly download the latest version of `SHYFEM`.

1.2.3 Latest versions and developers

In order to get also access to the latest single commits (which cannot be found on the `github` web page) you will have to install `git` on your computer. Once installed, go to the web page (see address above) and click on `clone`. This will download the latest version with all the versioning information included.
Once you have cloned `SHYFEM` you can get easy access to the latest versions and commits. Unpack the archive in any directory, and enter the newly created directory. Then type `git fetch` and `git pull`. This will give you the newest commit of the code base.
If you are a developer you should have git installed on your computer. If you have worked on a new feature and want your changes being published, you will have to issue what is called a `pull request`. You can do this from the `github` website. Please be sure that you base your `pull request` on the latest commit (not tag or release). Therefore, once you are ready with your `pull request`, do a `git fetch` and `git pull`, check if your changes are compatible, and then do the `pull request`.
All `pull requests` have to be based on the `develop` branch.

Chapter 2

Installing SHYFEM

2.1 Downloading and unpacking

The source code of the model is provided in a file named `shyfem-7_5_71.tar.gz` or similar, depending on the version of the code. In this case the version is 7_5_71. The file can be downloaded from the SHYFEM GIT repository[1], where it is possible to choose the desired version and where the latest *develop* version is available. Otherwise the code is available also in the SHYFEM web-site[2].
Once you have downloaded the model distribution, move the file to the directory in which you want to install the model and unpack the distribution. In the following we will assume that the file is in your home directory and your home directory is called `/home/model`. However, any other directory works as well. To unpack the distribution in your home directory, move there and run the command:

```
cd /home/model
tar xzvf shyfem-7_5_71.tar.gz
```

At this point a new folder named `/home/model/shyfem-7_5_71` has been created. This directory is the root of the SHYFEM model. All other commands given in this chapter assume that you are in this directory. Therefore, before reading on, please move into this directory:

```
cd /home/model/shyfem-7_5_71
```

2.2 Needed software

The source code is composed mainly of Fortran 90 files, but files written in C, Fortran 77, Perl and Shell scripts are also present.
In order to use the model you have to compile it in a Linux Operating System. Several software products must be present in order to be able to compile the model. Please refer to the documentation of your Linux distribution for installing these programs.

- The package `make` is required for compilation.
- The `perl` interpreter and the `bash` shell are necessary for compiling.
- A Fortran 77 and 90 compiler. Supported compilers are the Gnu compiler `gfortran`, the Intel Fortran compiler `ifort` and the Portland group `pgf90` Fortran compiler.

[1] https://github.com/SHYFEM-model/shyfem
[2] http://www.ismar.cnr.it/shyfem/

- A C compiler. Supported compilers are the Gnu `gcc`, the Intel C compiler `icc` or the IBM `xlc` C compiler.

Please note that you might already have everything available in your Linux distribution, with the exception maybe of the Fortran compiler.
To find out what software is installed on your computer and what you still have to install you can run the following command:

```
make check_software
```

If you get something like `bash: make: command not found`, then you do not have make installed. Please first install the `make` command and then run the command again.
The output of the command will show you what software you will still have to install. The software is divided into different sections. The first section is needed software, which you will not be able to do without. The next section is recommended software, which you really should install, but for compilation and running you will not necessarily need it. The last section is software which is optional, but which makes life easier.
You can always run `make check_software` again to check if the software had been successfully installed. When you are satisfied with the output you can go to the next section.
Depending on the options that you choose for the compilation you may need some additional package or library. Usually, the error message gives you the name of the missing library. The name of the corresponding package to install can be found at the web-page[3] for Debian OS. Usually, Debian-based (e.g., Ubuntu) distributions have the same name.
Whereas most package names are easy to guess, probably the only problem could be the developer X11 libraries. In order to be abel to compile the program `grid` you will need to install some packages that may have different names depending on your distribution. The packages you will have to look for are `libx11-dev`, `x11proto-core-dev` and `libxt-dev`.
Please note that you have to carry out the steps in this section only the first time you install the model. If you install a new version of SHYFEM software you can skip these steps.

2.3 Installation

Before compiling it is advisable to install some files for a simpler usage of the model. As long as you only want to run a simulation, this step is not strictly necessary. But if you will run some scripts of the distribution, these scripts will not work properly if you do not install the model.
In order to install the model, you should run

```
make install
```

This command will do the following:

- It hardcodes the installation directory in all scripts of the model so only programs of the installed version will be executed.
- It inserts a symbolic link `syhfem` from the home directory to the root of the SHYFEM installation.
- It inserts a small snippet of code into the initialization files `.bashrc .bash_profile .profile` that are in your home directory. This will adjust your path to point to the SHYFEM directory and gives you access to some administrative commands.

[3] https://www.debian.org/distrib/packages

After this command you will find the original files that have been changed in your home directory saved with a trailing number (e.g., `.profile.35624` or similar). If you encounter problems, just substitute back these files.
In order that your new settings will take effect you will have to log out and log in again.
If you do not want to run the installation routine, you should at least manually insert a symbolic link to the root of the SHYFEM model and modify your PATH enviromental variable:

```
    cd
    ln -fs /home/model/shyfem-7_5_71 shyfem
    echo -e "
nexport PATH=$PATH:$HOME/shyfem/fembin" >> .bashrc
```

If you have more versions installed, you should use `make install_hard` in order to use the complete paths of the directories in the scripts. You can run `make install_hard_reset` to restore the previous situation.
If you ever want to uninstall the model, you can do it with the command `make uninstall`. This will delete the symbolic link, cancel the hard links in the model scripts and restore the systemfiles `.bashrc .bash_profile .profile` to their original content.
Please note that you will still have to delete manually the model directory. This can be done with the command `rm -rf /home/model/shyfem-7_5_71`). In this way, however, changes to the code you have made will be lost.
For other options refers to the `Rules.make` file.

2.4 Compilation

In order to compile the model you will first have to adjust some settings in the `Rules.make` file. Assuming that you are already in the SHYFEM root directory (in our case it would be `/home/model/shyfem-7_5_71`), open the file `Rules.make` with a text editor. In this file the following options can be set:

- `Compiler profile`. Set the level of verbosity of the messages. Use `SPEED` if you want the maximum performances. Use the other options, in case of errors, to have more informations.
- `Compiler`. Set the compiler you want to use. Please see also the section on needed software and the one on compatibility problems to learn more about this choice. It is advaisable to use the same type of compiler for C and Fortran.
- `Parallel compilation`. Some parts of the code are parallelized with OpenMP statements. Here you can set if you want to use it or not.
- `Solver for matrix solution`. There are three different solvers implemented. `SPARSKIT` is an iterative solver, quite fast, and is the default option. The `GAUSS` solver is a robust direct solver, but it is quite slow. `PARDISO` is set as direct solver but can be used as iterative solver as well. It can be fast, but it is not included in the code, since it is not provided with a compatible licence. In order to use it, you need an external library (dynamically linked) provided with the Intel MKL.
- `NetCDF library`. If you want output files in NetCDF format you need the NetCDF library.
- `GOTM library`. The GOTM turbulence model is already included in the code. However, a newer and better tested version is available as an external module. In order to use it please let this variable to true. This is the recommended choice. You will need a Fortran 90 compiler to enable this choice.

- `Ecological module`. This option allows for the inclusion of an ecological module into the code. Choices are between `EUTRO`, `ERSEM` and `AQUABC`. Please refer to information given somewhere else on how to run these programs.

- `Fluid mud`. This is an experimental feature. Don't use it if you are not a developer.

Once you have set all these options you can start compilation with

```
make clean
make fem
```

This should compile everything. In case of a compilation error you will find some messages during compilation and also at the bottom of the output, where a check is run to see, if the main routines have been compiled.
Please remember that you will always have to run the commands above when you change settings in the `Rules.make` file. If you only change something in the code, or if you only change dimension parameters, it might be enough to run only `make fem`, which only compiles the necessary files. However, if you are in doubt, it is always a good idea to run `make clean` or `make cleanall` before compiling, in order to start from a clean state.

2.5 Compatibility problems

The SHYFEM program is designed to work with most of the compilers that are available. Normally there should be no problems with compatibility. However you have to keep in mind some points that are listed below.

- With `ifort` it is possible to open the same file in read only mode more than once. This is useful, e.g., if you have two open boundaries, but you want to prescribe the same value on these two boundaries. With `gfortran` or `pgf90` you cannot do this. A file, even in read only mode, can be opened only once. In the above example you therefore have to copy the input file to a new name (duplicate it) and then prescribe the two different files as boundary conditions.

- With `gfortran` it is very difficult to decide if a file is formatted or unformatted. Some modules allow the use of either formatted or unformatted input files, where the check on the file type is made via software. In case of `gfortran` this may not work reliably. The only solution to this problem is to specify the file type directly in the code.

- Objects generated during compilation and libraries used in linking are normally not compatible between compilers. What this means is that, when you switch compiler, you will have to recompile everything with `make cleanall; make fem`. Otherwise you will encounter errors during the linkage process.

- Unformatted files are normally not portable between different compilers. You normally cannot use a basin file created with programs compiled with one compiler together with a program compiled with another compiler. The same is true for unformatted data files (initial conditions, wind and meteo forcing, etc.).

  If you have problems reading a basin file, try `shybas`. If this is not working chances are high that you have the problem described above. In case of unformatted data files the diagnosis is not so easy. In any case, you can solve the problem recompiling all programs with the commands `make cleanall; make fem` and then re-creating all unformatted files with the newly compiled programs. In case of the basin file, you will have to run the pre-processor on the grid again.

If you have obtained unformatted data files from others, then there is really no easy solution to this problem. Exchanging unformatted files between different computers and compilers is never a good idea.

- A similar problem exists if you switch files between different architectures (32 bit and 64 bits), even if created with the same compiler. These files are normally not portable.
- Nan values (Not a Number) are treated differently between different compilers. Nan values are created if a not well defined operation is executed (divide by 0 or square root of a negative number). All compilers above (except `pgf90`) treat Nans to be not comparable to any number. This means that a logical expression `a.eq.a` is always false if `a` is a Nan. However the `pgf90` compiler treats Nans to be comparable to any other number. So, an expression like `a.ne.a` will evaluate to true. SHYFEM includes code to handle these problems gracefully, but incompatibilities might still show up.
- In parallel execution you might get a segmentation fault during execution. This is normally due to limited stack size. You can change the behavior by increasing the stack size (`ulimit -s unlimited`) on the console before running the program. Compilers may behave differently. Please see also the section on parallel execution in the file `Rules.make`.

2.6 Compilation options

Below a summary of administrative commands is given that are available in SHYFEM.

`make version`	shows version of distribution
`make clean`	deletes objects and executables from a previous compilation
`make cleanall`	same as `make clean` but also deletes compiled libraries
`make fem`	compiles SHYFEM
`make doc`	makes this manual (`femdoc/shyfem.pdf`)
`make check_software`	checks the availability of installed software
`make check_compilation`	checks if all programs have been compiled
`make changed`	finds files that are changed with respect to the original distribution
`make changed_zip`	zips files that are changed with respect to the original distribution to the file `changed_zip.zip`
`make install`	installs SHYFEM
`make uninstall`	uninstalls SHYFEM

Finally, if you have installed the model with `make install`, the following utility commands are available

`shyfemdir`	shows information about actual SHYFEM settings
`shyfemdir fem_dir`	sets `fem_dir` to be the new default SHYFEM version
`shyfeminstall`	shows information about original SHYFEM installation
`shyfemcd`	moves into root of actual SHYFEM directory

Chapter 3

Preprocessing: The numerical grid

3.1 Overview

Before you can start using the model you have to create a numerical grid. This step is more difficult for models that work on unstructured grids (like finite element models) than for finite difference models, where often it is enough to have a regular gridded bathymetry to start running simulations.
This chapter describes the steps needed to create a numerical grid for SHYFEM.
The files containing the informations on the computational grid, used by SHYFEM, are two:

- `filename.grd` formatted
- `filename.bas` unformatted

You must create the first one, while the second can be obtained automatically by the first. The `bas` file is the one really used by the model.
The `grd` file can be composed of 3 different parts, describing different geometric objects, which are: Nodes, Elements, Lines. These parts are the following:

- Node section, containing the nodes information and coordinates
- Element section, containing the elements information and the their nodes
- Line section, containing the domain contour infomation and nodes

The presence of these structures depends on type of `grd` file, for example, in a boundary line the structures will be the line and its nodes. For more details on the format, please, refer to `grd` file appendix B.
The steps to create a `grd` file are the following:

1. obtain raw digital data of the coastline and the bathymetry and convert them into a `grd` format
2. smooth and reduce the coastline if needed
3. create the numerical grid with a mesh generator
4. regularize the grid
5. interpolate bathymetry onto the created grid

6. create the unformatted `bas` file

These steps are described in the following sections.

3.2 Coastline and bathymetry

In order to create the computational grid you need data of the coast line and of the bathymetry. First of all you must create a coast line in `grd` format. You can do this with your own tools, but you could find useful the script `coast.pl`, which converts a simple coast file (x,y) into a `grd` file. To use it run:

```
coast.pl mpcoast.dat > coast.grd
```

After this step you can check your cost line with the `grid` program:

```
grid coast.grd
```

Likewise, if you have a bathymetry in a simple (x,y,z) format, you can convert it with:

```
bathy.pl mpbathy.dat > depth.grd
```

You can check the file `depth.grd` with `grid` as well. However, this file will be used only after the creation of the mesh.
Please note that UTM coordinates are normally huge numbers, there might be an accuracy problem when you try to create the grid. If this happens, you should first shift your UTM coordinates so that the origin of your new coordinate system coincides with the central point of your grid. This translation can be done using the program `grd_transl.pl`.
Other transformation routines are:

- `dxf2grd.pl` Transforms a grid from `dxf` (Autocad) to `grd` format. This is still experimental.
- `kml2grd.pl` Transforms a grid from the Google Earth format `kml` to `grd` format.
- `xyz2grd.pl` Transforms a simple list of nodes to `grd` format. Every line contains 3 values (x,y,z) or two values (x,y), when the information on depth is missing.

Please note that for SHYFEM depth values have to be positive. If your files have depth values as negative numbers, you will have to invert them. You can use the command

```
grd_modify.pl -depth_invert grd-file
```

to achieve this task.

3.3 Boundary line: smoothing and reducing

Every `grd` file can be open with the `grid` program. Normally, the coastline needs some post-processing. It might either have resolution which is too high, island might show up as open lines etc..
It is important that there is one closed boundary line that defines the whole domain of the computation. If you have an open coastline, please close the line with the routine `grid` at the places where you want your open boundary to be.
Once this domain boundary line has been defined, care has to be taken that the lines inside this domain, which denote islands, are closed.
Finally, the resolution of the boundary lines (coast and islands) have to be adjusted if you use the meshing program here provided. If the coastline is left as it is you might have a

much too high resolution along the boundaries. This is due to the fact that the meshing algorithm does not discard any points given to it. This means that all boundary nodes are used for the meshing. Therefore, if you have a very high resolution boundary line, you will get many elements along the boundary and relatively little elements (depending on the number of internal points) in the inside of the basin.
Smoothing and reduction of the boundary lines can be done with the routine `reduce`. The command is

```
reduce -s sigma -r reduct coast.grd
```

Here `sigma` specifies the length scale for the smoothing operator and `reduct` is the length scale below which points may be deleted. Both values have to be given in the same units of the coordinates of the file `coast.grd`, so normally meters. The smoothed file can be found in `smooth.grd` and the subsequently reduced file in `reduct.grd`.
If there are some points in the boundary line that should not be smoothed they can be given a depth value of -1. This is a flag that indicates that the position of these points will not be touched.

3.4 Manipulating nodes, lines and elements: the grid program

The `grid` program allows one not only to visualize the `grd` files but provides also a graphical user interface to manipulate the different items of the grid (Nodes, Elements, Lines). The command line and the options available for this program are following reported.

```
grid [-options] [files]

Options :
  -o   do not outline elements      -f   fill elements with color
  -k   do extra checking            -u   check if nodes are used
  -T   show type instead of depth   -c#  use color table #
  -h   print this help screen       -a   ask for file names
  -d   display  (only X11)          -g   geometry (only X11)
  -M#  scale color to depth #       -S#  size of color table is #
  -N#  scale factor for nodes is #  -V#  scale factor for vectors is #
  -C   color nodes and lines        -On  use n as output file name
  -t#  use type # for new items
```

General GUI commands

`scroll`	zoom in and out
`right click`	select item
`left click`	confirm item
`up arrow`	increase the node size
`down arrow`	decrease the node size

The main GUI menu is composed of
`File`
`View`
`Show`
`Node`
`Element`
`Line`

`Change`	
File menu	
`Cancel`	Obsolete command
`Refresh`	Refresh the screen view (to be done to view the last change to the grid)
`Print`	Create a Black and White PostScript of the grid `plot.ps`
`Save`	Save changes in `save.grd`
`Exit`	Quit GRID program
View menu	
`Zoom Window`	Zoom in a delimited window defined by left clicking two points (left-bottom and right-top)
`Zoom in`	Obsolete command replaced by mouse scroll
`Zoom out`	Obsolete command replaced by mouse scroll
`Total View`	Go back to the total view of the grid
`Move`	Obsolete command
Show menu	
`Show Node`	All the items selected by right clicking will be nodes
`Show Element`	All the items selected by right clicking will be elements
`Show Line`	All the items selected by right clicking will be lines
Node menu	
`Make Node`	Create a new node by left clicking
`Del Node`	Delete a node by selecting it (right click) and confirming it (left click). `Refresh` to see the changes.
`Move Node`	Move a node in a new position. Select it (right click) and confirm it (left click), give the new position (left click). `Refresh` to see the changes.
`Unify Node`	Unify two different nodes. Select the first node you want to unify (right click) and confirm it (left click), select the second node (left click). `Refresh` to see the changes.
Element menu	
`Make Element`	Create a new element. Create new nodes (left click) or select and confirm (right-left click) each node of the new element, clicking twice on the last one to close the element. The element has to be created in anti-clockwise sense.
`Del Element`	Remove the element but not its nodes.
`Remove Element`	Remove the element and its nodes.
Line menu	
`Make Line`	Create a new line. Create new nodes (left click) or select and confirm (right-left click) each node of the new line, clicking twice on the last one. In case of a close line it has to be created in anti-clockwise sense.
`Del Line`	Remove the line but not its nodes.
`Remove Line`	Remove the line and its nodes.

Split Line	Split the line in two parts.
Join Line	Join two lines in one.
Del Node	Delete the node from line but not from the domain.
Remove Node	Remove the node from line.
Insert Node	Insert a new node on the line.

Change menu

Change Depth	Change the depth of the selected item.
Change Type	Change the type of the selected item.

3.5 Creating the mesh

In order to create a very good quality mesh, it is advisable to use a dedicated program. We suggest to use the open source gmsh program, normally available in most of the Linux distributions. The following routines convert the format of the files:

- grd2gmsh.pl Converts a grd coast line in a geo file, readable from gmsh. You have to create Plain Surface with gmsh gui. Place them before the Size Filed and add manually Physical Surface.
- gmsh2grd.pl Converts a msh mesh in a grd mesh. Check the generated mesh with the command grid -k gsmh_msh.grd for for clockwise elements and node connections.

If you want to use the meshing algorithm provided with the SHYFEM package, called mesh, see mesh -h for help of the command line options.
In order to use the mesh algorithm you will have to provide to the program a coastline in which the program will insert triangles. There are some things to remember:
There must be exactly one closed outer (external) line that will enclose all the other lines given in the coastline file. This means that it is not possible to mesh two independent domains at the same time. Clearly, you can divide the grid file into more files, each of which contains just one independent domain. These files can then be meshed independently.
The program normally is able to find out what is the external line. It will simply be the line that encloses all the other lines. If no such line is found, then this will lead to an error. The program will distinguish between the external line, islands and fault lines. Fault lines are lines that will constrain triangles to not cross these lines. For example, putting a fault line along the edge of a channel will ensure that the triangles will not cross the channel edge but will be placed along this edge.
In order to decide what line is of which type, the program considers the largest closest line as the external line, all other lines as islands, and any open line as a fault line. Normally this is the expected behavior. The program will classify these lines only if the line type is 0.
If you want to overwrite this behavior you can give explicit line types to the lines in the coast line file. A type of 1 signals an external line, a type of 2 an island, and a type of 4 a fault line. Clearly there can be only one line with type 1. If you have more than one line with type 1 the program will exit with an error. You can however have a fault line which is closed, a behavior that will not be possible with the automatic determination of line types, because a closed line with type 0 is always an island. Clearly an open line with type 2 (island) is also an error.
The mesh routine is able to create a grid with uniform or different resolution mesh, depending on user interest.

```
MESH - Automatic Grid Generation Routine - Version 1.75
       1995-2009 (c) Georg Umgiesser - ISMAR-CNR

Options :
 -b   do not refine boundaries      -n   do not insert internal nodes
 -s#  passes for smoothing          -o#  relax. par. for smoothing
 -I#  number of internal nodes      -B#  number of boundary nodes
 -R#  overall resolution            -a#  obtain this aspect ratio
 -g#  element type for background   -h   show this help screen
```

If you want a grid with a uniform solution all over, then you are already in a position to run the meshing algorithm. You just say: `mesh -I2000 coast-new.grd` and then the constructed mesh will be in final.grd. The number 2000 means that you want aprox. 2000 internal points in the domain. You may adjust this number to your needs.
However, you will normally want to have different resolution in the domain (high at the inlets of lagoons, at interesting sites like harbours etc..). Then you have to construct a background grid that gives an indication to the meshing algorithm what kind of resolution is need in what area.
You open the coastline with grid and construct elements that cover the parts or all of the domain. The areas where no background grid exists will use the (constant) resolution of the domain computed by the routine mesh using the total number of internal nodes (2000 in this example).
Where a background grid exists the model uses the depth values at the element vertices (nodes) to compute a new value for this resolution. The depth value acts like a factor that multiplies the constant overall resolution to obtain a local resolution. So, for example, constructing a background grid and setting all depth values to 1 would not change the resolution at all from a situation without background grid. A factor higher than 1 increases the resolution and one smaller than 1 decreses it. Therefore, in areas where resolution should be higher than average you can set it to 2 or higher, and in other areas, where you want lower resolution, you can set it to 0.5 or lower. All nodes of the background grid need to have a depth (resolution) value. In side each background element the resolution is interpolated between the three nodes (vertices).
In order to distinguish the background grid from the elements that are constructed by the meshing routine, they must become a unique element type. You can set it to a value that is not used for other elements (99 is a good choice). All elements of the background grid must have this element type.
withse extract the background grid from the grid file you just have constructed by running exgrd: ”exgrd -LS coast-new.grd”. The file ”new.grd” contains only the background grid. Rename it to something more useful (mv new.grd bck1.grd). The following is a recapitulation of steps for the background grid creation:

- manually construct background grid using coast-new.grd
- delete coastline (`exgrd -LS coast-new.grd`). This leaves only background elements in file.
- set depth at nodes for resolution
- set type in elements to 99
- rename to bck1.grd
- mv new.grd bck1.grd

Now you are ready to start the meshing algorithm.

3.5.1 Meshing of the basin

The main options of mesh routine are:

```
-I2000          use aprox. 2000 internal nodes for the domain
-g99            element type of background grid is 99
```

With this parameters the call to mesh would be:

```
mesh -I2000 -g99 coast-new bck1
```

The created mesh can be found in final.grd.
Please note that you can specify mor than one file for the coast line, so you could keep the domain line and the island lines in seperate files. You can also have different background grid for different areas in different files. So a call like this is also possible:

```
mesh -I2000 -g99 coast island1 island2 bck1 bck2 bck3
```

After the meshing please have a look at the result (final.grd). If you need more overall resolution, increase the number of internal points (here 2000). If you need more resolution in the background grid, open the background file and increase the factor (depth) value where needed. You might also need other areas with a background grid. Once you are satisfied with the result please save it to a more meaningful name.

```
mesh -I2000 -g99 coast-new bck1
mv final.grd mesh1.grd
```

3.5.2 Adjust elements for regularity

After the creation of the mesh, the grid is still not good enough for usage in a finite element model. This is due to the fact that the grid is too irregular. Therefore a program has to be applied that regularizes the grid.
The program is called `regularize`. It must be given the input grid file (irregular) and creates a new one with much more regular characteristics. The program has to be called as:

```
regularize mesh1.grd mesh2.grd
```

In this case the new regular grid is in `mesh2.grd`. Note that you can use this program even if you have made your grid using a program different from `mesh`.

3.6 Interpolating the bathymetry into the grd file

After the grid creation, with `mesh` or other programs, you must interpolate the bathymetry. The bathymetry must be contained in a `grd` file, previously created. This file, together with the basin onto which the bathymetry has to be interpolated, has to be specified for the program `shybas`. The simplest call is:

```
shybas -bfile bathy.grd mesh2.grd
```

where `bathy.grd` is the `grd` file with the bathymetry values and `mesh2.grd` is the basin for which to interpolate the bathymetry. Different types of interpolation can be used. Please run `shybas -h` for more options.

3.7 Creating the bas file

The pre-processing routine `shypre` is used to generate the `bas` unformatted file from the `grd` formatted file. You can use it with the command:

```
shypre final_mesh.grd
```

The main task of routine `shypre` is the optimization of the internal numbering of the nodes and elements. Re-numbering the elements is just a mere convenience. When assembling the system matrix the contribution of one element after the other has to be added to the system matrix. If the elements are numbered in terms of lowest node numbers, then the access of the nodal pointers is more regular in computer memory and paging is more likely to be inhibited.
However, re-numbering the nodes is absolutely necessary. The system matrix to be solved is of band-matrix type. I.e., non-zero entries are all concentrated along the main diagonal in a more or less narrow band. The larger this band is, the larger the amount of cpu time spent to solve the system. The time to solve a band matrix is of order $n \cdot m^2$, where n is the size of the matrix and m is the bandwidth. Note that m is normally much smaller than n.
If the nodes are left with the original numbering, it is very likely that the bandwidth is very high, unless the nodes in the file GRD are by chance already optimized. Since the bandwidth m is entering the above formula quadratically, the amount of time spent solving the matrix will be prohibitive. E.g., halving the bandwidth will speed up computations by a factor of 4.
The bandwidth is equal to the maximum difference of node numbers in one element. It is therefore important to re-number the nodes in order to minimize this number. However, there exist only heuristic algorithms for the minimization of this number.
The two main algorithms used in the routine `shypre` are the Cuthill McGee algorithm and the algorithm of Rosen. The first one, starting from one node, tries to number all neighbors in a greedy way, optimizing only this step. From the points numbered in this step, the next neighbors are numbered.
This procedure is tried from more than one node, possibly from all boundary nodes. The numbering resulting from this algorithm is normally very good and needs only slight enhancements to be optimum.
Once all nodes are numbered, the Rosen algorithm tries to exchange these node numbers, where the highest difference can be found. This normally gives only a slight improvement of the bandwidth. It has been seen, however, that, if the node numbers coming out from the Cuthill McGee algorithm are reversed, before feeding them into the Rosen algorithm, the results tend to be slightly better. This step is also performed by the program.
All these steps are performed by the program without intervention by the operator, if the automatic optimization of bandwidth is chosen in the program `shypre`. The choices are to not perform the bandwidth optimization at all (`grd` file has already optimized node numbering), perform it automatically or perform it manually. It is suggested to always perform automatic optimization of the bandwidth. This choice will lead to a nearly optimum numbering of the nodes and will be by all means good results.
If, however, you decide to do a manual optimization, please follow the online instructions in the program.

3.7.1 Internal and external node numbering

As explained above, the elements and nodes of the basin are re-numbered in order to optimize the bandwidth of the system matrix and so the execution speed of the program.
However, this re-numbering of the node and elements is transparent to the user. The program keeps pointers from the original numbering (external numbers) to the optimized numbering (internal numbers). The user has to deal only with external numbers, even if the program uses internally the other number system.

Moreover, the internal numbers are generated consecutively. Therefore, if there are a total of 4000 nodes in the system, the internal nodes run from 1 to 4000. The external node numbers, on the other side, can be anything the user likes. They just must be unique. This allows for insertion and deletion of nodes without having to re-number over and over again the basin.
The nodes that have to be specified in the input parameter file use again external numbers. In this way, changing the structure of the basin does not at all change the node and element numbers in the input parameter file. Except in the case, where modifications actually touch nodes and elements that are specified in the parameter file.

Chapter 4

Running SHYFEM

In the following an overview is given on running the model SHYFEM. The model needs a parameter input file in ascii format, with extension str, that is read on standard input. Moreover, it needs some external files that are specified in this parameter input file. The model produces several external files with the results of the simulation. Again, the name of this files can be influenced by the parameter input file

4.1 The parameter input file (str)

Once the str file (e.g., param.str) is made, the following line starts the simulation

```
shyfem param.str
```

4.1.1 Structure

The input parameter file is the file that guides program performance. It contains all the necessary informations for the main routine to execute the model. Nearly all parameters that can be given have a default value which is used when the parameter is not listed in the file. Only some time parameters are compulsory and must be present in the file.
The format of the file looks very like a namelist format, but is not dependent on the compiler used. Values of parameters are given in the form : name = value or name = 'text'. If name is an array the following format is used :

```
name = value1 , value2, ... valueN
```

The list can continue on the following lines. Blanks before and after the equal sign are ignored. More then one parameter can be present on one line. As separator blank, tab and comma can be used.
Parameters, arrays and data must be given in between certain sections. A section starts with the character $ followed by a keyword and ends with $end. The $keyword and $end must not contain any blank characters and must be the first non blank characters in the line. Other characters following the keyword on the same line separated by a valid separator are ignored.
Several sections of data may be present in the input parameter file. Further ahead all sections are presented and the possible parameters that can be specified are explained. The sequence in which the sections appear is of no importance. However, the first section must always be section \$title, the section that determines the name of simulation and the basin file to use and gives a one line description of the simulation.
Lines outside of the sections are ignored. This gives the possibility to comment the parameter input file.

Figure 4.1 shows an example of a typical input parameter file and the use of the sections and definition of parameters.

4.2 Basic usage

This section explains typical usage of the model. It will show how the model can be run doing basic 2D hydrodynamic simulations, simulate a passive tracer, compute T/S, use the Coriolis force and apply wind forcing. More advanced usages of the model, like 3D simulations and the use of the turbulence module will be presented later. This section is conceived as a simple HOWTO document. For the exact meaning and usage of the single parameters, please see the section on input parameters.
To run a simulation, two things are needed. The first is the description of the basin and the numerical grid, which must be prepared beforehand and then must be compiled in a form that the model can use. How this is been done has already been described in the chapter dealing with preprocessing.
The second thing that is needed is a description of the simulation and the forcings that have to be applied. This is done through a parameter input file. Here we call it `str` file, because historically these files always ended with an extension of `.str`. However, any extension can be used.

4.2.1 Minimal simulation

A basic version of an `str` file can be found in 4.2. In fact, it is so basic, it really does not do anything. Here only the compulsory parameters have been inserted. These are:

- An introductory section `$title` where on three lines the following information is given:

  1. A description of the run. This can be any text that fits on one line.
  2. The name of the simulation. This name is used for all files that the simulation produces. These files differ from each other only by their extension.
  3. The name of the basin. This is the basin file without the extension `.bas`. The file must exist in the current directory.

- A section `$para` that contains all necessary parameters for the simulation to be run. The only compulsory parameters are the ones that specify the start of the simulation `itanf`, its end `itend`, its time step `idt` and a reference date (`date = yyyymmdd`). This is the reference for all the time parameters used in the `str` file and in all the files provided as input to the simulation, as well as the output files.

All the time parameters can be specified in seconds from the reference date or using a date label `'yyyy-mm-dd::HH:MM:SS'`. The parameters that specify time steps can be prescribed both in seconds or using the following labels: `'Ns'`, `'Nm'`, `'Nh'`, `'Nd'`. Where `N` is a number, `s` means seconds, `m` minutes, `h` hours and `d` days.
In order to be more helpful, some more information must be added to the `str` file. As an example let's have a look on 4.1. Here we have added two parameters that deal with the type of friction to be used. `ireib` specifies the bottom friction formulation, here through a simple quadratic bulk formula. (For the exact meaning of the parameters, please refer to the appendix where all parameters are listed.) The parameter `czdef` specifies the value to use for the bottom drag coefficient.

```
#--------------------------------------------------------------------
#
#    Copyright (C) 1985-2020  Georg Umgiesser
#
#    This file is part of SHYFEM.
#
#--------------------------------------------------------------------

$title
        benchmark test for test lagoon
        bench
        venlag
$end

$para
        itanf = 0  itend = 86400  idt = 300
        ireib = 5  czdef = 2.5E-3
        dragco = 2.5E-3
$end

$bound1
        kbound = 73 74 76
        boundn = 'levels1.dat'
$end

$bound2
        kbound = 150 157 97 101
        boundn = 'levels1.dat'
$end

$name
        wind='win18sep.win'
$end

   next are tide gauges used in calibration

$extra ------- tide gauges for calibration ---------
13,133,99,259,328,772,419,1141,1195,1070,1064,942,468,1154
73,74,76,353,350,349,1374,1154,1160,1161,408,409,786,795
$end
```

Figure 4.1: Example of a parameter input file (`STR` file)

```
#-------------------------------------------------------------------
#
#    Copyright (C) 1985-2020  Georg Umgiesser
#
#    This file is part of SHYFEM.
#
#-------------------------------------------------------------------

$title
        benchmark test for test lagoon
        bench
        venlag
$end

$para
        itanf = 0  itend = 86400  idt = 300
$end
```

Figure 4.2: Example of a basic parameter input file (`str` file)

4.2.2 Boundary conditions

In order to have a more meaningfull simulation, we need to specify boundary conditions. In this section we will deal with the open boundary conditions, e.g., the conditions at the place where the basin comunicates with other water bodies (e.g., for lagoons it could be the inlets).

For every boundary condition one section `$bound` must be specified. Since you can have more than one open boundary you must specify also the number of your boundary, e.g., `$bound1`, `$bound2` etc. Inside every section you can then specify the various parameters that characterize your boundary.

Basically there two types of open boundary conditions. Either the water level or the discharges (fluxes) can be specified. The parameter that decides the type of boundary is `ibtyp`. A value of one indicates water levels, instead a value of 2 or 3 indicates fluxes. If you specify discharges entering at the border of the domain, `ibtyp` = 2 should be specified. Otherwise, if there are internal sources in the basin then `ibtyp` = 3 must be used. If you do not define this parameter, a value of 1 will be used and water levels will be specified.

The only compulsary parameter in this section is the list of boundary nodes. You do this with the parameter `kbound`. In the case of `ibtype` 1 or 2 at least two nodes must be specified, in order to give an extension of the boundary. The numeration of the boundary nodes must be consecutive and with the basin on its left side when going along the boundary nodes. In the case of `ibtyp` = 3 even a single point can be given.

The boundary values you want to give are normally specified through a a file with a time series. You give the name of the file that contains the time series with the parameter `boundn`. An example with two boundaries can again be found in Fig. 4.1. Here water levels are prescribed and the values for the water levels are read from a file `levels1.dat`.

If the values on the boundary you want to impose can be described through a simple sinus function, you can also give the bounadry values specifying the parameters for the sinus function. An example of a water level boundary with a tide of $\pm 70cm$ and a period of 12 hours (semi-diurnal) is given in Fig. 4.3. Note thet `zref` gives the average water level of the boundary. If you specify `ampli`=0 you get a constant boundary value of `zref`.

```
$bound1
      ibtyp = 1   kbound = 23 25 28
      ampli = 0.70  period = 43200  phase = 10800  zref = 0.
$end
```

Figure 4.3: Example of a boundary with regular sinusoidal water levels. The pahse of 10800 (3 hours) makes sure that the simulation starts at slack tide when the basin is completely full.

4.2.3 Wind forcing

The wind and the mean sea level pressure can be prescribed by means of an external file, with extension `fem`, which can be both formatted and unformatted. Please see the section on file formats to write such a file correctly. The name of the file must be specified in the `str` file in the section `name`, using the flag `wind`. For example:

```
$name
...
wind = 'mywind.fem'
...
$end
```

Other important parameters to set in the `str` file, in the section `para`, are `iwtype` and `itdrag`. The former specifies the wind stress formulation, while the latter is used to prescribe a constant value of the wind drag coefficient.
For more information see the appendix.

4.3 Advanced usage

4.3.1 Variable time step

Generally SHYFEM is run with a fixed time step given by the parameter `idt`. This choice is acceptable when the model runs in unconditionally stable conditions (ie. linear simulation, no horizontal viscosity).
The non-linear terms of the momentum advection (`ilin=0`) or the horizontal viscosity (`ahpar` greater 0) can introduce computational instabilities. To be sure that the model runs in stable conditions, it must be assured that the Courant Number is smaller than 1. Please note that only in the case of advection we should call this number the Courant number. However, we will continue to use the term Courant number for all stability related issues.
In the case of advection the Courant number is defined as

$$Cou = \frac{v\Delta t}{\Delta x} \tag{4.1}$$

where v is the current speed, Δt the time step and Δx the element size. For finite elements, due to the triangular grid, this expression is slightly more complicated. As can be seen, lowering the time step will bring the Courant number below the limit of 1.
To keep the Courant Number under the limit it is necessary to adapt the time step at every computation. The variable timestep is computed introducing in the `str` file in the `$para` section the parameters `itsplt`, `coumax` and `idtsyn`.
`coumax` gives the limit of the Courant number. This is normally 1, but since no exact stability limit can be derived for the non-linear advective terms, another value can be specified. If instabilities arise, a slightly lower value than 1 (0.9) can be tried.
`itsplt` decides about the time step splitting. If this value is 0, the time step will be kept constant at its initial value. A value of 1 devides the initial time step into (possibly) equal

parts, but makes sure that at the end of the micro time steps one complete macro time step has been executed. The last mode `itsplt` = 2 does not care about the macro time step, but always uses the biggest time step possible. In this case it is not assured that after some micro time steps a macro time step will be recovered. Please note that the initial macro time step `idt` will never be exceeded.
Finally, the parameter `idtsyn` is only used in case of `itsplt` = 2. This parameter makes sure that after a time of `idtsyn` the time step will be syncronized to this time. Therefore, setting `idtsyn` = 3600 means that there will be a time stamp every hour, even if the model has to take one very small time step in order to reach that time.
An example of how to set the variable time stepping scheme is shown in Fig. 4.4. Here the Courant number is lowered to 0.9 and the variable time step is syncronized every 3600 seconds (1 hour).

```
$para
        coumax = 0.9   itsplt = 2   idtsyn = 3600
$end
```

Figure 4.4: Example of variable time step settings. The time step is syncronized at every hour, and the Courant number is lowered to 0.9.

4.3.2 3D computations

General information

The basic way to run the model is in 2D, computing for each element of the grid one value for the whole water column. All the variables are computed in the center of the layer, halfway down the total depth. Deeper basins or highly variable bathymetry can require for the correct reproduction of the velocities, temperature and salinity the need for 3D computation.
The 3D computation is performed with z layers, sigma layers or with a hybrid formulation. In the default z layer formulation, each layer horizontally has constant depth over the whole basin, but vertically the layer thickness may vary between different layers. However, the first layer (surface layer) is of varying thickness because of the water level variation, and the last layer of an element might be only partially present due to the bathymetry.
Layers are counted from the the surface layer (layer 1) down to the maximum layer, depending again on the local depth. Therefore, elements (and nodes) normally have a different total number of layers from one to each other. This is opposed to sigma layers where the number of total layers is constant all over the basin, but the thickness of each layer varies between different elements.

Z layers

In order to use z layers for 3D computations a new section `$layers` has to be introduced into the `STR` file, where the sequence of depth values of the bottom of the layers has to be declared. Layer depths must be declared in increasing order. An example of a `$layer` section is given in figure 4.5. Please note that the maximum depth of the basin in the example must not exceed 20 m.
A specific treatment for the bottom layer has to be carried out. In fact, if the model runs on basins with variable bathymetry, for each element there will be a different total number of layers. The bathymetric value normally does not coincide with one of the layer depths, and therefore the last layer must be treated separately.
To declare how to treat the last layer two parameters have to be inserted in the `$para` section. The first is `hlvmin`, the minimum depth, expressed as a percentage with respect to

```
$layers
2 4 6 8 10 13 16 20
$end
```

Figure 4.5: Example of section $layers for z layers. The maximum depth of the basin is 20 meters. The first 5 layers have constant thickness of 2 m, while the last three vary between 3 and 4 m.

the full layer depth, ranging between 0 and 1, This is the fraction that the last layer must have in order to be maintained as a separate layer. The second parameter is ilytyp and it defines the kind of adjustment done on the last layer. If it is set to 0 no adjustment is done, if it is set to 1 the depth of the last layer is adjusted to the one declared in the STR file (full layer change). If it is 2 the adjustment to the previous layer is done only if the fraction of the last layer is smaller than hlvmin (change of depth). If it is 3 (default) the bathymetric depth is kept and added to the last but one layer. Therefore with a value of 0 or 3 the total depth will never be changed, whereas with the other levels the total depth might be adjusted.
As an example, take the layer definition of Fig. 4.5. Let hlvmin be set to 0.5, and let an element have a depth of 6.5 m. The total number of layers is 4, where the first 3 have each a thickness of 2 m and the last layer of this element (layer 4) is 0.5 m. However, the nominal thickness of layer 4 is 2 m and therefore its relative thickness is 0.25 which is smaller than hlvmin. With ilytyp=0 no adjustment will be done and the total number of layers in this element will be 4 and the last layer will have a thickness of 0.5 m. With ilytyp=1 the total number of layers will be changed to 3 (all of them with 2 m thickness) and the total depth will be adjusted to 6 m. The same will happen with ilytyp=2, because the relative thickness in layer 4 is smaller than hlvmin. Finally, with ilytyp=3 the total number of layers will be changed to 3 but the remaining depth of 0.5 m will be added to layer 3 that will become 2.5 m.
In the case the element has a depth of 7.5 m, the relative thickness is now 0.75 and greater than hlvmin. In this case, with ilytyp=0, 2 and 3 no adjustment will be done and the total number of layers in this element will be 4 and the last layer will have a thickness of 1.5 m. With ilytyp=1 the total number of layers will be kept as 4 but the total depth will be adjusted to 8 m. This will make all layers equal to 2 m thickness.

Sigma layers

Sigma layers use a constant number of layers all over the basin. They are easier to use than z layers, because only one parameter has to be specified. In the $para section of the STR file the parameter nsigma has to be set to the number of desired sigma layers. The layers are then equally spaced between each other.
Sigma layers can be also specified in the $layers section. In this case the negative percentage of the layers have to be given. An example is given in figure 4.6. This is only useful if the layers are not equally spaced, because for equally spaced sigma layers the parameter nsigma in the STR can be used.
In the bathymetry file depth values have to be given on nodes and not on elements. in case the utility shybas can be used to convert between elemental and nodal depth values.

Hybrid layers

Hybrid layers are also called ”sigma over zeta” layers. They are a mixture between sigma layers close to the surface and zeta layers in the deeper parts of the basin. This is useful if strong bathymetry gradients are present. This avoids possible instabilities due to the sigma layers in the deeper parts.

```
$layers
-0.2 -0.4 -0.6 -0.8 -1.0
$end
```

Figure 4.6: Example of section `$layers` for sigma layers. The depth is divided in 5 equally spaced layers. Please note that this division could have also been achieved setting `nsigma` to the value of 5.

For the hybrid layers a depth of closure has to be defined. This is the depth value above which sigma layers are used and below which zeta levels are applied. Please note that the basin has to be prepared in order that depth values are given on nodes and in an elements the three depth values on the vertices are either higher equal or lower equal than the depth of closure. The routine `shybas` can be used in order to create a grid compatible with hybrid coordinates. An example of how to specify hybrid layers is given in figure 4.7.

```
$layers
-0.2 -0.4 -0.6 -0.8 10. 20. 30. 40. 50
$end
```

Figure 4.7: Example of section `$layers` for hybrid layers. The depth is divided in 5 equally sigma layers on the surface above 10 meters (which is the depth of closure) and zeta layers below until a depth of 50 meters.

Please note that hybrid layers are still experimental. So use with care.

Vertical viscosity

The introduction of layers requires also to define the values of vertical eddy viscosity and eddy diffusivity. In any case a value of these two parameters has to be set if the 3D run is performed. This could be done by setting a constant value of the parameters `vistur` (vertical viscosity) and `diftur` (vertical diffusivity). In this case possible values are between $1 \cdot 10^{-2}$ and $1 \cdot 10^{-5}$, depending on the stability of the water column. Higher values ($1 \cdot 10^{-2}$) indicate higher stability and a stronger barotropic behavior.
The other possibility is to compute the vertical eddy coefficients through a turbulence closure scheme. This usage will be described in the section on turbulence.

4.3.3 Baroclinic terms

The baroclinic term permits to compute the variation of the water density due to the horizontal gradients of temperature and/or salinity. These variations causes an additional motion of the water.
In order to use the baroclinic term, the parameter `ibarcl` (in section `$para`) must be set different from 0.
Setting `ibarcl` to a value different from 0 will simulate the transport and diffusion of temperature and salinity in the basin. A value of 1 will compute the full baroclinic pressure terms, due to density gradients, and the advection and diffution of temperature and salinity. A value of 2 will do diagnostic simulations. This means that baroclinic pressure terms are still included in the hydrodynamic equations, but temperature and salinity values will be read from a file. Finally for `ibarcl`=3 temperature and salinity will be computed but no baroclinic pressure term will be used. In this case the hydrodynamic equations and the equations for temperature and salinity are decoupled and there is no feed back from the water density field to the currents.

It is advisable to use a 3D computation with the non-linear terms and a variable time step. In any case, if temperature and salinity are computed, first they must be initialized either with constant values or with variable 3D matrices. In the first case the reference values have to be imposed in `temref` and `salref`. An example of this type of simulation is given in Fig. 4.8.
If the temperature and salinity are given as 3D matrices files, they must be provided in the `$name` section, giving the file names in `tempin` and `saltin`. In case of diagnostic simulations the matrices of temperature and salinity have to be provided in the files named `tempd` and `saltd` and data must be available for the whole period of simulation.
If the `ibarcl`=1 option is used, the following additional forcing files must the provided:

- A file with short wave solar radiation, relative humidity, air temperature and cloudiness
- A file containing percipitation data

```
$para
        ibarcl = 1   temref = 18.    salref = 35.
$end
```

Figure 4.8: Example of baroclinic simulation. The initial values for temperature and salinity are set to 18 C and 35.

4.3.4 Restart

The model solves the shallow water equations, forwarding in time an initial state of the hydrodynamical system. This state is composed by all the model independent variables. The restart routines allow to load the initial values for these variables, which, otherwise, would be unknown and set to zero. The restart allows also to write the state in a file one time or at different time intervals.
To load a restart file the parameters `itrst` and `restrt` must be set. The first must be included in the section `para` and is the time (in seconds from the initial date or in date label) relative to the restart record to read. The second must be specified in the section `name` and is the name of the restart file, which must have extension `rst`. For example:

```
$para
...
itrst = '2016-01-25::06:00'
...
$end

...

$name
...
restrt = 'myrestart.rst'
...
$end
```

A new restart file can be created by using `itmrst`, the time to write the first restart record, and `idtrst`, the time step between different records. Both the parameters must be specified in the `para` section. For example:

```
$para
...
itmrst = '2016-01-26::06:00'
idtrst = '6h'
...
$end
```

Finally if you want to check the records contained in a restart file, you can use `rstinf`:

```
rstinf myrestart.rst
```

For more information see the description of the parameters in the appendix.

4.3.5 Turbulence

In the Reynolds equations turbulent eddy diffusivities and viscosities are introduced into the equations that must be parameterized and given some value. Moreover SHYFEM assumes the hydrostatic approximation. Therefore, there is the need to parameterize the non-hydrostatic effects. These are considered sub-scale processes which are mainly of convective nature.
Vertical eddy viscosities and diffusivities have to be defined if there is the intent to model the turbulence effects. These vertical eddy viscosities and diffusivities can be set to constant values, defining `vistur` and `diftur` in the `$para` section. There is also the opportunity to compute, at each timestep, variable values of them, using the turbulence closure module.
The parameter that has to be set in order too choose the turbulence scheme is `iturb` in the `$para` section.. If `iturb`=0 the vertical eddy viscosity and eddy diffusivity are set constant (default 0) and must be defined in `vistur` and `diftur`.
If `iturb`=1 the GOTM turbulence closure module is used. If `iturb`=2 the turbulence closure scheme applied is the $k-\varepsilon$ model. Finally, if `iturb`=3, the Munk-Anderson model is used.
With `iturb`=1, the file `gotmturb.nml` must be provided that sets all necessary parameters. This file must be declared in the section `$name` for the item `gotmpa`.
A default `gotmturb.nml` file is provided and it allows the computation of the vertical eddy viscosity and eddy diffusivity by means of the GOTM $k-\varepsilon$ model. More information on the GOTM turbulence closure module can be found in the GOTM Manual [1].
If the turbulence module should be used, a value of `iturb`=1 is recommended. An example of the settings for the turbulence closure scheme is given in Fig. 4.9.

```
$para
        iturb = 1
$end
$name
        gotmpa = 'gotmturb.nml'
$end
```

Figure 4.9: Example of turbulence settings. The GOTM module for the turbulence closure is used. The parameters are contained in file gotmturb.nml.

4.3.6 Sediment transport

The sediment transport module calculates sediment transport for currents only or combined waves and currents over either cohesive and non-cohesive sediments. The core of this

[1] http://www.gotm.net/index.php?go=documentation

module is derived by the SEDTRANS05 sediment transport model, which is here coupled with SHYFEM. The sediment transport module computes the erosion and deposition rates at every mesh node and determines the sediment volume that is injected into the water column. After this step the sediments are advected with the transport and diffusion module described above. The module update every time step the characteristcs of the bottom in terms of grainsize composition and sediment density.
The sediment transport module is activated by setting `isedi` = 1 and `sedgrs` in the section `sedtr`. For more details about the parameters see the Appendix C.
For more information about the sediment transport module please refer to Neumeier et al. [14] and Ferrarin et al. [6].

Sediment transport formulation

For the non-cohesive sediment there are two transport mechanisms, the bedload and the suspended transport, while for the cohesive sediment is assumed to exist only the suspended transport. Then we use the sediment continuity equation for the non-cohesive bedload transport and the transport and diffusion equation to describe the transport of the suspended sediment, both cohesive and non-cohesive. For the bedload component a direct advection scheme is used. Five methods are proposed to predict sediment transport for non-cohesive sediments. The methods of Brown [3], Yalin [25] and Van Rijn [22] predict the bedload transport. The methods of Engelund and Hansen [5] and Bagnold [2] predict the total load transport.
Different approaches have been used to compute the sediment flux between water column and bottom for cohesive and non-cohesive sediment. For cohesive sediments the model computes erosion and deposition rates, while for non-cohesive the net sediment flux between bottom and water column s computed as the difference between the equilibrium concentration and the existing concentration in the lower level. Here the source (erosion occurs, flux from the bottom to the water column) term has been taken as explicit, whereas the sink term as semi-implicit (deposition occurs, flux from the water column to the bottom). This approach permits to avoid negative concentration due to deposition higher then the sediment mass present in the water column.
The vertical mixing coefficient has been calculated using analytical expressions given by [22] for both the case of current related and wave related turbulent mixing.

Bed representation

The bed module is designed to have spatially different characteristics, such as grainsize composition, sediment density and critical stress for erosion. The bed is subdivided in several layers and levels. Each layer is considered homogeneous, well mixed and characterized by its own grainsize distribution (fraction of each class of sediment considered). On the level are defined the dry bulk density ρ_{dry} and the critical stress for erosion τ_{ce}; it is assumed that these variables vary linearly between two levels. These characteristics could vary spatially in the domain.
At each location the uppermost layer has to be always greater or equal to the surficial active, or mixed, layer that is available for suspension. Sediment below the active layer is unavailable resuspension until the active layer moves downward either because erosion has occurred, or it has thickened due to increase shear stress. As active layer is considered the bottom roughness height considered as the sum of the grain roughness, the bedload roughness and the bedform (ripple) roughness.
Multiple sand grain size classes are considered to behave independently. At each location the average grain size (based on the sediment fractions) is used to compute the bed roughness and critical shear velocities. Modification of the bed elevation and to the grain size distribution are updated at each time step based on the net erosion and deposition. For

each size class, the volume of sediment removed from the bed during any time step is limited by the amount available in the active layer. In this way the model takes into account time-dependent and spatial sediment distribution and bed armouring.

Sand-mud mixture

The morphological behaviour of estuaries and lagoons often depend on non-cohesive as well as cohesive sediments. Prediction the distribution of sediments in these environments, characterized by zonation of sand, mud and mixed deposits, is of crucial importance for sustainable management and development of such systems.
Based on laboratory and field experiments several researchers identified a transition from non-cohesive to cohesive behaviour at increasing mud content in a sand bed. A sand bed with small amount of mud already shows increased resistance against erosion. Above a critical mud content ($\% < 0.063$ mm), the bed behaved cohesively. The critical mud content depend on the history of the bed and on the geochemical properties of the sand and mud and could varies from 3-20 %. Such a wide range clearly demonstrate that the parameters governing the erosion behaviour of sand-mud mixture, are not fully understood yet. For this reason is crucial to estimate the critical mud or clay content from laboratory and field experiments.
Below the critical mud content the sediments particles are eroded as non-cohesive sediments. It is assumed that the sediments in suspension always behave independently, either if flocculation processes of the thinner particles could trap sand grains into the floc.
Moreover sand increases the binding between the clay particles and results in a more compact and dense matrix which is more resistant to erosion. For non-consolidated cohesive bed adding sand to mud increase the erosion resistance because of the increased density and consolidation rate. The improved critical stress for erosion due to sand particles is taken into consideration increasing the dry density with the sand fraction.

Sediment model output

The sediment transport model writes the following output:

- erosion/deposition [m], 2D variable 80 in file SED
- average grainsize of first bottom layer [m], 2D variable 81 in file SED
- bed shear stress [Pa] of first bottom layer, 2D variable 82 in file SED
- updated node depth [m], 2D variable 83 in file SED
- total bedload transport [kg/ms], 2D variable 84 in file SED
- total suspended concentration [kg/m3], 3D variable 85 in file SCO

The time step and start time for writing to file SED and SCO are defined by the parameters `idtcon` and `itmcon` in the `para` section. These parameter are the same used for writing tracer concentration, salinity and water temperature. If `idtcon` is not defined, then the sediment model does not write any results.

4.3.7 Wind waves

SHYFEM could be coupled with the spectral wind wave unstructured model WWMIII or computes wave characteristics using empirical prediction equations.
This empirical wave module is used to calculate the wave height and period from wind speed, fetch and depth using the EMPIRICAL PREDICTION EQUATIONS FOR SHALLOW WATER [21].

WWMIII is not provided in the SHYFEM distribution. The coupling of SHYFEM with WWMIII is done through the FIFO PIPE mechanism. The numerical mesh need to be converted to the GR3 format using the bas2wwm program. WWMIII needs its own input parameter file (wwminput.nml). The use of the coupled SHYFEM-WWMIII model require additional software which is not described here.
The wave module writes in the WAV file the following output:

- significant wave height [m], variable 231
- mean wave period [s], variable 232
- mean wave direction [deg], variable 233

The time step and start time for writing to file WAV are defined by the parameters `idtwav` and `itmwav` in the `waves` section. These parameter are the same used for writting tracer concentration, salinity and water temperature. If `idtwav` is not defined, then the wave module does not write any results. The wave results can be plotted using `plots -wav`.
In case of SHYFEM-WWMIII coupling several variables are exchanged between the two models:

- SHYFEM sends to WWMIII:
  - surface velocities
  - water level
  - bathymetry and number of vertical layers
  - 3D layer depths
  - wind components**
- SHYFEM reads from WWMIII:
  - gradient of the radiation stresses
  - significant wave height
  - mean period
  - significant wave direction
  - wave supported stress
  - peak period
  - wave length
  - orbital velocity
  - stokes velocities
  - wind drag coefficient
  - wave pressure
  - wave dissipation
  - wind components**

**Wind could be either read from SHYFEM or WWMIII, see parameter `iwave` in Appendix C. For more information about WWMIII and its couling with SHYFEM please refer to Roland et al. [15] and Ferrarin et al. [6].

4.3.8 Tidal potential

SHYFEM includes an astronomical tidal model which can be activated by setting the parameter `rtide` equal 1 in the `para` section.
The model calculates equilibrium tidal potential (η) and load tides (β) and uses these to force the free surface. The term η in the momentum equations is calculated as a sum of the tidal potential of each tidal constituents multiplied by the frequency-dependent elasticity factor. The factor β accounts for the effect of the load tides, assuming that loading tides are in-phase with the oceanic tide. β is function of the water depth as $\beta = ltidec * H$ with `ltidec` a calibration factor to be set in the str `para` section.
The model cosiders the following tidal costituents:

- Semidiurnal species:
  - M2 semi-diurnal principal lunar
  - S2 semi-diurnal principal solar
  - N2 large elliptical tide of first-order to M2
  - K2 semi-diurnal declination to M2
  - NU2 large evection tide to M2
  - MU2 large variation tide to M2
  - L2 small elliptical tide of first-order to M2
  - T2 large elliptical tide of first-order to S2
- Diurnal species:
  - K1 declination luni-solar
  - O1 principal lunar
  - P1 principal solar
  - Q1 elliptical lunar
  - J1 elliptical tide of first-order to K1
  - OO1 evection tide to O1
  - S1 radiational tide
- Long-Period species:
  - MF fortnightly lunar
  - MM monthly lunar
  - MSM S0-semiannual solar
  - MSF Evection tide to M0
  - SSA semiannual solar
  - SA elliptical tide of first-order to S0

SHYFEM also allows to perform the tidal analysis of water levels during the model runtime. The tidal analysis is actived by setting `itmtid` and `idttid`. `idttid` should be long enough to perform a reliable analysis. The parameter `itmtid` can be used to start the analysis after the simulation spin-up. The tidal analysis module write an output file .tide.shy containing amplitudes and phases of all tidal constituents over the computational domain (on the nodes).

4.3.9 Meteo forcing

The description of this module is under work
The list of parameters related to the meteorological forcing is reported in the appendix.

4.3.10 Lagrangian particle module

Lagrangian analysis provides a powerful tool to evaluate the output of ocean circulation models. SHYFEM is equipped with a 3-D particle-tracking module, which simulates the trajectory of particles as a function of the hydrodynamics.
The vertical components of the turbulent diffusion velocity is computed using the Milstein scheme reported by Grawe [8]. The horizontal diffusion was computed using a random walk technique based on Fisher [7], with the turbulent diffusion coefficients obtained by means of the Smagorinsky [17] formulation. The wind drag and Stokes drift contribution to the total transport is parametrized by `stkpar` factor. An additional calibration parameter to account for the drifter inertia could be set (`dripar`). The model allows particle to beach on the shore (`lbeach`).
The particle-tracking model can be also used off-line (parameter `idtoff`). In this case it uses the Eulerian hydrodynamic fields generated by the forecast system. The main advantage of the off-line approach is that the trajectory calculation typically takes much less computational effort than the driving hydrodynamic model.
The lagrangian particles can be released:

- inside the given areas (filename `lgrlin`). If this file is not specified they are released over the whole domain. The amount of particles released and the time step is specified by `nbdy` and `idtl`.
- at selected times and location, e.g. along a drifter track (filename `lgrtrj`). `nbdy` particles are released at the times and location specified in the file.
- as initial particle distribution (filename `lgrini`) at time `itlgin`. This file has the same format as the lagrangian output.
- at the open boundaries, either as particles per second or per volume flux (parameter `lgrpps`).

The particle-tracking model is activated by setting `ilagr` ¿ 0. The lagrangian module runs between the times `itlanf` and `itlend`. See more details in the list of parameters and they description reported in the appendix.
The lagrangian model can be used in other sub-modules to specifically simulate sediments (`ised=1`), oil (`ioil=1`) and larvae (`ilarv=1`).

Appendix A

Hydrodynamic equations and resolution techniques

A.1 Equations and Boundary Conditions

The equations used in the model are the well known vertically integrated shallow water equations in their formulation with water levels and transports.

$$\frac{\partial U}{\partial t} + gH\frac{\partial \zeta}{\partial x} + RU + X = 0 \tag{A.1}$$

$$\frac{\partial V}{\partial t} + gH\frac{\partial \zeta}{\partial y} + RV + Y = 0 \tag{A.2}$$

$$\frac{\partial \zeta}{\partial t} + \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} = 0 \tag{A.3}$$

where ζ is the water level, u, v the velocities in x and y direction, U, V the vertical integrated velocities (total or barotropic transports)

$$U = \int_{-h}^{\zeta} u\,dz \qquad V = \int_{-h}^{\zeta} v\,dz$$

g the gravitational acceleration, $H = h + \zeta$ the total water depth, h the undisturbed water depth, t the time and R the friction coefficient. The terms X, Y contain all other terms that may be added to the equations like the wind stress or the nonlinear terms and that need not be treated implicitly in the time discretization. following treatment.
The friction coefficient has been expressed as

$$R = \frac{g\sqrt{u^2 + v^2}}{C^2 H} \tag{A.4}$$

with C the Chezy coefficient. The Chezy term is itself not retained constant but varies with the water depth as

$$C = k_s H^{1/6} \tag{A.5}$$

where k_s is the Strickler coefficient.
In this version of the model the Coriolis term, the turbulent friction term and the nonlinear advective terms have not been implemented.
At open boundaries the water levels are prescribed. At closed boundaries the normal velocity component is set to zero whereas the tangential velocity is a free parameter. This corresponds to a full slip condition.

A.2 The Model

The model uses the semi-implicit time discretization to accomplish the time integration. In the space the finite element method has been used, not in its standard formulation, but using staggered finite elements. In the following a description of the method is given.

A.2.1 Discretization in Time - The Semi-Implicit Method

Looking for an efficient time integration method a semi-implicit scheme has been chosen. The semi-implicit scheme combines the advantages of the explicit and the implicit scheme. It is unconditionally stable for any time step Δt chosen and allows the two momentum equations to be solved explicitly without solving a linear system.
The only equation that has to be solved implicitly is the continuity equation. Compared to a fully implicit solution of the shallow water equations the dimensions of the matrix are reduced to one third. Since the solution of a linear system is roughly proportional to the cube of the dimension of the system the saving in computing time is approximately a factor of 30.
It has to be pointed out that it is important not to be limited with the time step by the CFL criterion for the speed of the external gravity waves

$$\Delta t < \frac{\Delta x}{\sqrt{gH}}$$

where Δx is the minimum distance between the nodes in an element. With the discretization described below in most parts of the lagoon we have $\Delta x \approx 500$m and $H \approx 1$m, so $\Delta t \approx 200$ sec. But the limitation of the time step is determined by the worst case. For example, for $\Delta x = 100$ m and $H = 40$ m the time step criterion would be $\Delta t < 5$ sec, a prohibitive small value.
The equations (1)-(3) are discretized as follows

$$\frac{\zeta^{n+1} - \zeta^n}{\Delta t} + \tfrac{1}{2}\frac{\partial(U^{n+1} + U^n)}{\partial x} + \tfrac{1}{2}\frac{\partial(V^{n+1} + V^n)}{\partial y} = 0 \tag{A.6}$$

$$\frac{U^{n+1} - U^n}{\Delta t} + gH\tfrac{1}{2}\frac{\partial(\zeta^{n+1} + \zeta^n)}{\partial x} + RU^{n+1} + X = 0 \tag{A.7}$$

$$\frac{V^{n+1} - V^n}{\Delta t} + gH\tfrac{1}{2}\frac{\partial(\zeta^{n+1} + \zeta^n)}{\partial y} + RV^{n+1} + Y = 0 \tag{A.8}$$

With this time discretization the friction term has been formulated fully implicit, X, Y fully explicit and all the other terms have been centered in time. The reason for the implicit treatment of the friction term is to avoid a sign inversion in the term when the friction parameter gets too high. An example of this behavior is given in Backhaus [1].
If the two momentum equations are solved for the unknowns U^{n+1} and V^{n+1} we have

$$U^{n+1} = \frac{1}{1 + \Delta t R}\left(U^n - \Delta t g H \tfrac{1}{2}\frac{\partial(\zeta^{n+1} + \zeta^n)}{\partial x} - \Delta t X\right) \tag{A.9}$$

$$V^{n+1} = \frac{1}{1 + \Delta t R}\left(V^n - \Delta t g H \tfrac{1}{2}\frac{\partial(\zeta^{n+1} + \zeta^n)}{\partial y} - \Delta t Y\right) \tag{A.10}$$

If ζ^{n+1} were known, the solution for U^{n+1} and V^{n+1} could directly be given. To find ζ^{n+1} we insert (A.9) and (A.10) in (A.6). After some transformations (A.6) reads

$$\zeta^{n+1} \quad - \quad (\Delta t/2)^2 \frac{g}{1 + \Delta t R}\left(\frac{\partial}{\partial x}(H\frac{\partial \zeta^{n+1}}{\partial x}) + \frac{\partial}{\partial y}(H\frac{\partial \zeta^{n+1}}{\partial y})\right)$$

$$
\begin{aligned}
&= \zeta^n + (\Delta t/2)^2 \frac{g}{1+\Delta t R}\left(\frac{\partial}{\partial x}(H\frac{\partial \zeta^n}{\partial x}) + \frac{\partial}{\partial y}(H\frac{\partial \zeta^n}{\partial y})\right) \qquad \text{(A.11)}\\
&- (\Delta t/2)\left(\frac{2+\Delta t R}{1+\Delta t R}\right)\left(\frac{\partial U^n}{\partial x} + \frac{\partial V^n}{\partial y}\right)\\
&+ \frac{\Delta t^2}{2(1+\Delta t R)}\left(\frac{\partial X}{\partial x} + \frac{\partial Y}{\partial y}\right)
\end{aligned}
$$

The terms on the left hand side contain the unknown ζ^{n+1}, the right hand contains only known values of the old time level. If the spatial derivatives are now expressed by the finite element method a linear system with the unknown ζ^{n+1} is obtained and can be solved by standard methods. Once the solution for ζ^{n+1} is obtained it can be substituted into (A.9) and (A.10) and these two equations can be solved explicitly. In this way all unknowns of the new time step have been found.
Note that the variable H also contains the water level through $H = h + \zeta$. In order to avoid the equations to become nonlinear ζ is evaluated at the old time level so $H = h + \zeta^n$ and H is a known quantity.

A.2.2 Discretization in Space - The Finite Element Method

While the time discretization has been explained above, the discretization in space has still to be carried out. This is done using staggered finite elements. With the semi-implicit method described above it is shown below that using linear triangular elements for all unknowns will not be mass conserving. Furthermore the resulting model will have propagation properties that introduce high numeric damping in the solution of the equations.
For these reasons a quite new approach has been adopted here. The water levels and the velocities (transports) are described by using form functions of different order, being the standard linear form functions for the water levels but stepwise constant form functions for the transports. This will result in a grid that resembles more a staggered grid in finite difference discretizations.

Formalism

Let u be an approximate solution of a linear differential equation L. We expand u with the help of basis functions ϕ_m as

$$u = \phi_m u_m \qquad m = 1, K \qquad \text{(A.12)}$$

where u_m is the coefficient of the function ϕ_m and K is the order of the approximation. In case of linear finite elements it will just be the number of nodes of the grid used to discretize the domain.
To find the values u_m we try to minimize the residual that arises when u is introduced into L multiplying the equation L by some weighting functions Ψ_n and integrating over the whole domain leading to

$$\int_\Omega \psi_n L(u)\, d\Omega = \int_\Omega \psi_n L(\phi_m u_m)\, d\Omega = u_m \int_\Omega \psi_n L(\phi_m)\, d\Omega \qquad \text{(A.13)}$$

If the integral is identified with the elements of a matrix a_{nm} we can write (A.13) also as a linear system

$$a_{nm} u_m = 0 \qquad n = 1, K \qquad m = 1, K \qquad \text{(A.14)}$$

Once the basis and weighting functions have been specified the system may be set up and (A.14) may be solved for the unknowns u_m.

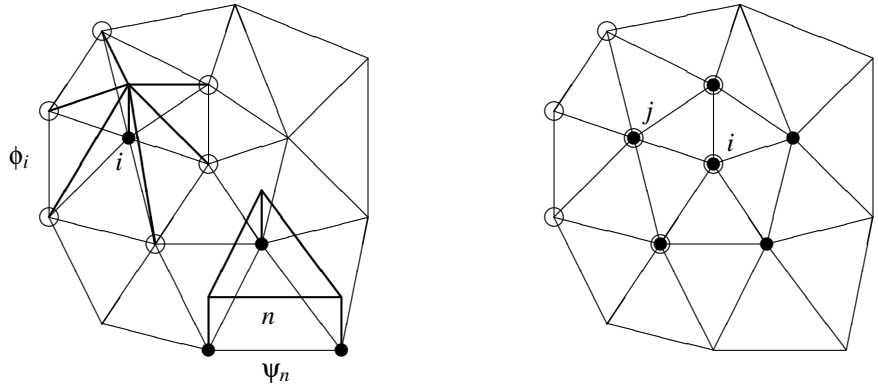

Figure A.1: a) form functions in domain b) domain of influence of node i

Staggered Finite Elements

For decades finite elements have been used in fluid mechanics in a standardized manner. The form functions ϕ_m were chosen as continuous piecewise linear functions allowing a subdivision of the whole area of interest into small triangular elements specifying the coefficients u_m at the vertices (called nodes) of the triangles. The functions ϕ_m are 1 at node m and 0 at all other nodes and thus different from 0 only in the triangles containing the node m. An example is given in the upper left part of Fig. 1a where the form function for node i is shown. The full circle indicates the node where the function ϕ_i take the value 1 and the hollow circles where they are 0.
The contributions a_{nm} to the system matrix are therefore different from 0 only in elements containing node m and the evaluation of the matrix elements can be performed on an element basis where all coefficients and unknowns are linear functions of x and y.
This approach is straightforward but not very satisfying with the semi-implicit time stepping scheme for reasons explained below. Therefore an other way has been followed in the present formulation. The fluid domain is still divided in triangles and the water levels are still defined at the nodes of the grid and represented by piecewise linear interpolating functions in the internal of each element, i.e.

$$\zeta = \zeta_m \phi_m \qquad m = 1, K$$

However, the transports are now expanded, over each triangle, with piecewise constant (non continuous) form functions ψ_n over the whole domain. We therefore write

$$U = U_n \psi_n \qquad n = 1, J$$

where n is now running over all triangles and J is the total number of triangles. An example of ψ_n is given in the lower right part of Fig. 1a. Note that the form function is constant 1 over the whole element, but outside the element identically 0. Thus it is discontinuous at the element borders.
Since we may identify the center of gravity of the triangle with the point where the transports U_n are defined (contrary to the water levels ζ_m which are defined on the vertices of the triangles), the resulting grid may be seen as a staggered grid where the unknowns are defined on different locations. This kind of grid is usually used with the finite difference method. With the form functions used here the grid of the finite element model resembles very much an Arakawa B-grid that defines the water levels on the center and the velocities on the four vertices of a square.
Staggered finite elements have been first introduced into fluid mechanics by Schoenstadt [16]. He showed that the un-staggered finite element formulation of the shallow water

equations has very poor geostrophic adjustment properties. Williams [23, 24] proposed a similar algorithm, the one actually used in this paper, introducing constant form functions for the velocities. He showed the excellent propagation and geostrophic adjustment properties of this scheme.

The Practical Realization

The integration of the partial differential equation is now performed by using the subdivision of the domain in elements (triangles). The water levels ζ are expanded in piecewise linear functions ϕ_m, $m = 1, K$ and the transports are expanded in piecewise constant functions ψ_n, $n = 1, J$ where K and J are the total number of nodes and elements respectively. As weighting functions we use ψ_n for the momentum equations and ϕ_m for the continuity equation. In this way there will be K equations for the unknowns ζ (one for each node) and J equations for the transports (one for each element).
In all cases the consistent mass matrix has been substituted with the their lumped equivalent. This was mainly done to avoid solving a linear system in the case of the momentum equations. But it was of use also in the solution of the continuity equation because the amount of mass relative to one node does not depend on the surrounding nodes. This was important especially for the flood and dry mechanism in order to conserve mass.

Finite Element Equations

If equations (A.9,A.10,A.11) are multiplied with their weighting functions and integrated over an element we can write down the finite element equations. But the solution of the water levels does actually not use the continuity equation in the form (A.11), but a slightly different formulation. Starting from equation (A.6), multiplied by the weighting function Φ_M and integrated over one element yields

$$\int_\Omega \Phi_N(\zeta^{n+1} - \zeta^n)\, d\Omega + (\tfrac{\Delta t}{2}) \int_\Omega \left(\Phi_N \frac{\partial (U^{n+1} + U^n)}{\partial x} + \Phi_N \frac{\partial (V^{n+1} + V^n)}{\partial y} \right) d\Omega = 0$$

If we integrate by parts the last two integrals we obtain

$$\int_\Omega \Phi_N(\zeta^{n+1} - \zeta^n)\, d\Omega - (\tfrac{\Delta t}{2}) \int_\Omega \left(\frac{\partial \Phi_N}{\partial x}(U^{n+1} + U^n) + \frac{\partial \Phi_N}{\partial y}(V^{n+1} + V^n) \right) d\Omega = 0$$

plus two line integrals, not shown, over the boundary of each element that specify the normal flux over the three element sides. In the interior of the domain, once all contributions of all elements have been summed, these terms cancel at every node, leaving only the contribution of the line integral on the boundary of the domain. There, however, the boundary condition to impose is exactly no normal flux over material boundaries. Thus, the contribution of these line integrals is zero.
If now the expressions for U^{n+1}, V^{n+1} are introduced, we obtain a system with again only the water levels as unknowns

$$\begin{aligned} \int_\Omega \Phi_N \zeta^{n+1}\, d\Omega \quad &+ \quad (\Delta t/2)^2 \alpha g \int_\Omega H \left(\frac{\partial \Phi_N}{\partial x} \frac{\partial \zeta^{n+1}}{\partial x} + \frac{\partial \Phi_N}{\partial y} \frac{\partial \zeta^{n+1}}{\partial y} \right) d\Omega \\ &= \quad \int_\Omega \Phi_N \zeta^n\, d\Omega + (\Delta t/2)^2 \alpha g \int_\Omega H \left(\frac{\partial \Phi_N}{\partial x} \frac{\partial \zeta^n}{\partial x} + \frac{\partial \Phi_N}{\partial y} \frac{\partial \zeta^n}{\partial y} \right) d\Omega \\ &+ \quad (\Delta t/2)(1+\alpha) \int_\Omega \left(\frac{\partial \Phi_N}{\partial x} U^n + \frac{\partial \Phi_N}{\partial y} V^n \right) d\Omega \\ &- \quad (\Delta t^2/2)\alpha \int_\Omega \left(\frac{\partial \Phi_N}{\partial x} X + \frac{\partial \Phi_N}{\partial y} Y \right) d\Omega \end{aligned} \tag{A.15}$$

Here we have introduced the symbol α as a shortcut for

$$\alpha = \frac{1}{1 + \Delta t R}$$

The variables and unknowns may now be expanded with their basis functions and the complete system may be set up.

A.2.3 Mass Conservation

It should be pointed out that only through the use of this staggered grid the semi-implicit time discretization may be implemented in a feasible manner. If the Galerkin method is applied in a naive way to the resulting equation (A.11) (introducing the linear form functions for transports and water levels and setting up the system matrix), the model is not mass conserving. This may be seen in the following way (see Fig. 1b for reference). In the computation of the water level at node i, only ζ and transport values belonging to triangles that contain node i enter the computation (full circles in Fig. 1b). But when, in a second step, the barotropic transports of node j are computed, water levels of nodes that lie further apart from the original node i are used (hollow circles in Fig. 1b). These water levels have not been included in the computation of ζ_i, the water level at node i. So the computed transports are actually different from the transports inserted formally in the continuity equation. The continuity equation is therefore not satisfied.
These contributions of nodes lying further apart could in principle be accounted for. In this case not only the triangles Ω_i around node i but also all the triangles that have nodes in common with the triangles Ω_i would give contributions to node i, namely all nodes and elements shown in Fig. 1b. The result would be an increase of the bandwidth of the matrix for the ζ computation disadvantageous in terms of memory and time requirements.
Using instead the approach of the staggered finite elements, actually only the water levels of elements around node i are needed for the computation of the transports in the triangles Ω_i. In this case the model satisfies the continuity equation and is perfectly mass conserving.

A.2.4 Inter-tidal Flats

Part of a basin may consist of areas that are flooded during high tides and emerge as islands at ebb tide. These inter-tidal flats are quite difficult to handle numerically because the elements that represent these areas are neither islands nor water elements. The boundary line defining their contours is wandering during the evolution of time and a mathematical model must reproduce this features.
For reasons of computer time savings a simplified algorithm has been chosen to represent the inter-tidal flats. When the water level in at least one of the three nodes of an element falls below a minimum value (5 cm) the element is considered an island and is taken out of the system. It will be reintroduced only when in all three nodes the water level is again higher then the minimum value. Because in dry nodes no water level is computed anymore, an estimate of the water level has to be given with some sort of extrapolation mechanism using the water nodes nearby.
This algorithm has the advantage that it is very easy to implement and very fast. The dynamical features close to the inter-tidal flats are of course not well reproduced but the behavior of the method for the rest of the lagoon gave satisfactory results.
In any case, since the method stores the water levels of the last time step, before the element is switched off, introducing the element in a later moment with the same water levels conserves the mass balance. This method showed a much better performance than the one where the new elements were introduced with the water levels taken from the extrapolation of the surrounding nodes.

Appendix B

File formats

B.1 GRD file

```
format of input file for grid utility

============================================================

legend :

n item number (node, element, line)
t type
d depth
x,y coordinates
ntot number of following nodes
n1,n2 node numbers

============================================================

format of lines in input file :

comment:

0 [anything]

node:

1 n t x y [d]

element:

2 n t ntot n1 n2 n3 ...  [d]

line:

3 n t ntot n1 n2 ... [d]

============================================================

comment :
```

```
lines may be split at any point, except befor optional argument
d must not be on seperate line
if line is split, the continuation line(s) must start with a blank
blank lines can be inserted as needed
if d is not specified -999. will be used (flag)
use t=0 if you do not know what to use
n must be unique for every item type
item numbers need not be consecutive
the sequence of items is not important,
nodes can be mixed with elements/lines
the minimum number of nodes for element items is 3
the minimum number of nodes for line items is 2
element items should have all nodes unique
line items with the same first and last node are considered closed

============================================================

example 1 :

0 example of one line

1 11 0 10. 10.
1 12 0 20. 20.

3 7 0 2 11 12

#----------------

example 2 :

0 example of one element with continuation line

1 11 0 10. 10.
1 12 0 20. 20.
1 15 0 10. 20.

2 7 0 3
   11 12 15

============================================================
```

Appendix C

Parameter list

C.1 Parameter list for the SHYFEM model

C.1.1 Section `$title`

This section must be always the first section in the parameter input file. It contains only three lines. An example is given in figure C.1.

```
$title
        free one line description of simulation
        name_of_simulation
        name_of_basin
$end
```

Figure C.1: Example of section `$title`

The first line of this section is a free one line description of the simulation that is to be carried out. The next line contains the name of the simulation. All created files will use this name in the main part of the file name with different extensions. Therefore the hydrodynamic output file (extension `out`) will be named `name_of_simulation.out`. The last line gives the name of the basin file to be used. This is the pre-processed file of the basin with extension `bas`. In our example the basin file `name_of_basin.bas` is used.
The directory where this files are read from or written to depends on the settings in section `$name`. Using the default the program will read from and write to the current directory.

C.1.2 Section `$para`

This section `$para` defines the general behavior of the simulation, gives various constants of parameters and determines what output files are written. In the following the meaning of all possible parameters is given.
Note that the only compulsory parameters in this section are the ones that chose the duration of the simulation and the integration time step. All other parameters are optional.

Compulsory time parameters These parameters are compulsory parameters that define the period of the simulation. They must be present in all cases.

`itanf` Start time of simulation. (Default 0)

`itend` End time of simulation.

`idt` Time step of integration.

Output parameters The following parameters deal with the output frequency and start time to external files. The content of the various output files should be looked up in the appropriate section.
The default for the time step of output of the files is 0 which means that no output file is written. If the time step of the output files is equal to the time step of the simulation then at every time step the output file is written. The default start time of the output is `itanf`, the start of the simulation.

`idtout` Time step and start time for writing to file OUT, the file
`itmout` containing the general hydrodynamic results.

`idtext` Time step and start time for writing to file EXT, the file
`itmext` containing hydrodynamic data of extra points. The extra points for which the data is written to this file are given in section `extra` of the parameter file.

`idtrst` Time step for writing the restart file (extension RST). No restart file is written with `idtrst` equal to 0. A negative value is also possible for the time step. In this case the time step used is `-idtrst`, but the file is overwritten every time. It therefore contains always only the last written restart record. The special value of `idtrst = -1` will write only the last time step of the simulation in the restart file. This is useful if you want to start another simulation from the last output. (Default 0)

`itmrst` Start time for writing the restart file. If not given it is the beginning of the simulation.

`itrst` Time to use for the restart. If a restart is performed, then the file name containing the restart data has to be specified in `restrt` and the time record corresponding to `itrst` is used in this file. A value of -1 is also possible. In this case the last record in the restart file is used for the restart and the simulation starts from this time. Be aware that a value of -1 changes the parameter `itanf` to the time of the last record found in `restrt`.

`ityrst` Type of restart. If 0 and the restart file is not found the program will exit with an error. Otherwise the program will simply continue with a cold start. If `ityrst` is 1 and the given time record is not found in the file it will exit with error. If it is 2 it will initialize all values from the first time record after `itrst`. Therefore, the value of 2 will guarantee that the program will not abort and continue running, but it might not be doing what you intended. (Default 0)

`flgrst` This variable indicates which variables are read from the restart file. By default all available variables are read and used. If some variables are not wanted (because, e.g., you want to restart from a different T/S field), this fact can be indicated in `flgrst`. 1 indicates restart of hydro values, 10 the depth values, 100 T/S values, 1000 the tracer concentration, 10000 vertical velocities and 100000 the ecological variables. Therefore, a value of 10111 indicates a restart of everything except the tracer and the ecological values. The default value for `flgrst` is -1, which means 111111.

`idtres` `itmres` Time step and start time for writing to file RES, the file containing residual hydrodynamic data.

`idtrms` `itmrms` Time step and start time for writing to file RMS, the file containing hydrodynamic data of root mean square velocities.

`idtflx` `itmflx` Time step and start time for writing to file FLX, the file containing discharge data through defined sections. The transects for which the discharges are computed are given in section `flux` of the parameter file.

`idtstb` `itmstb` Time step and start time for writing a file with the stability index for debug reasons.

`idtdbg` `itmdbg` Time step and start time for writing a debug file with values of various variables. In order to use this feature you have to run `shyfem` with the command line option `-debout`. Please be aware that this feature will create extremely big files. Use at you own risk.

`idtvol` `itmvol` Time step and start time for writing to file VOL, the file containing volume information of areas defined by transects. The transects that are used to compute the volumes are given in section `volume` of the parameter file.

`netcdf` This parameter chooses output in NetCDF format if `netcdf` is 1, else the format is unformatted FORTRAN files. (Default 0)

idtoff handles offline mode (default 0):

0 do nothing (no offline routines called)

>0 write offline data file (.off) with time step idtoff.

<0 reads offline data from file offlin defined in section name. Usage:

-1 uses offline hydro results

-2 uses offline T/S results

-4 uses offline turbulence results

Combinations are possible. A value of -3 would read hydro and T/S results.

itmoff Start time for writing to file OFF, the file containing data for offline runs.

General time and date parameters A time and date can be assigned to the simulation. These values refer to the time 0 of the FEM model. The format for the date is YYYYMMDD and for the time HHMMSS. You can also give a time zone if your time is not referring to GMT but to another time zone such as MET.
date The real date corresponding to time 0. (Default 0) time The real time corresponding to time 0. (Default 0) tz The time zone you are in. This is 0 for GMT, 1 for MET and 2 for MEST (MET summer time). (Default 0)

Model parameters The next parameters define the inclusion or exclusion of certain terms of the primitive equations.

ilin Linearization of the momentum equations. If ilin is different from 0 the advective terms are not included in the computation. (Default 1)

itlin This parameter decides how the advective (non-linear) terms are computed. The value of 0 (default) uses the usual finite element discretization over a single element. The value of 1 chooses a semi-lagrangian approach that is theoretically stable also for Courant numbers higher than 1. It is however recommended that the time step is limited using itsplt and coumax described below. (Default 0)

iclin Linearization of the continuity equation. If iclin is different from 0 the depth term in the continuity equation is taken to be constant. (Default 0)

The next parameters allow for a variable time step in the hydrodynamic computations. This is especially important for the non-linear model (ilin=0) because in this case the criterion for stability cannot be determined a priori and in any case the time integration will not be unconditionally stable.
The variable time steps allows for longer basic time steps (here called macro time steps) which have to be set in idt. It then computes the optimal time step (here micro time step) in order to not exceed the given Courant number. However, the value for the macro time step will never be exceeded.

Normally time steps are always given in full seconds. This is still true when specifying the macro time step idt. In older versions also the computed micro time steps also had to be full integer values. Starting from version 7.1 also fractional time steps are allowed. This gives the possibility to have time steps smaller than 1 s.

itsplt — Type of variable time step computation. If this value is 0, the time step will be kept constant at its initial value. A value of 1 divides the initial time step into (possibly) equal parts, but makes sure that at the end of the micro time steps one complete macro time step has been executed. The mode itsplt = 2 does not care about the macro time step, but always uses the biggest time step possible. In this case it is not assured that after some micro time steps a macro time step will be recovered. Please note that the initial macro time step will never be exceeded. In any case, the time step will always be rounded to the next lower integer value. This is not the case with itsplt = 3 where the highest possible fractional time step will be used. (Default 0)

coumax — Normally the time step is computed in order to not exceed the Courant number of 1. However, in some cases the non-linear terms are stable even for a value higher than 1 or there is a need to achieve a lower Courant number. Setting coumax to the desired Courant number achieves exactly this effect. (Default 1)

idtsyn — In case of itsplt = 2 this parameter makes sure that after a time of idtsyn the time step will be synchronized to this time. Therefore, setting idtsyn = 3600 means that there will be a time stamp every hour, even if the model has to take one very small time step in order to reach that time. This parameter is useful only for itsplt = 2 and its default value of 0 does not make any synchronization.

idtmin — This variable defines the smallest time step possible when time step splitting is enabled. Normally the smallest time step is 1 second. Please set idtmin to values smaller than 1 in order to allow for fractional time steps. A value of 0.001 allows for time steps of down to 1 millisecond. (Default 1)

These parameters define the weighting of time level in the semi-implicit algorithm. With these parameters the damping of gravity or Rossby waves can be controlled. Only modify them if you know what you are doing.

azpar — Weighting of the new time level of the transport terms in the continuity equation. (Default 0.5)

ampar — Weighting of the new time level of the pressure term in the momentum equations. (Default 0.5)

afpar — Weighting of the new time level of the Coriolis term in the momentum equations. (Default 0.5)

avpar — Weighting of the new time level of the non-linear advective terms in the momentum equations. (Default 0.0)

The next parameters define the weighting of time level for the vertical stress and advection terms. They guarantee the stability of the vertical system. For this reason they are normally set to 1 which corresponds to a fully implicit discretization. Only modify them if you know what you are doing.

`atpar` Weighting of the new time level of the vertical viscosity in the momentum equation. (Default 1.0)

`adpar` Weighting of the new time level of the vertical diffusion in the scalar equations. (Default 1.0)

`aapar` Weighting of the new time level of the vertical advection in the scalar equations. (Default 1.0)

Coriolis parameters The next parameters define the parameters to be used with the Coriolis terms.

`icor` If this parameter is 0, the Coriolis terms are not included in the computation. A value of 1 uses a beta-plane approximation with a variable Coriolis parameter f, whereas a value of 2 uses an f-plane approximation where the Coriolis parameter f is kept constant over the whole domain. (Default 0)

`dlat` Average latitude of the basin. This is used to compute the Coriolis parameter f. This parameter is not used if spherical coordinates are used (`isphe`=1) or if a coordinate projection is set (`iproj` $>$0). (Default 0)

`isphe` If 0 a cartesian coordinate system is used, if 1 the coordinates are in the spherical system (lat/lon). Please note that in case of spherical coordinates the Coriolis term is always included in the computation, even with `icor` = 0. If you really do not want to use the Coriolis term, then please set `icor` = -1. The default is -1, which means that the type of coordinate system will be determined automatically.

Depth parameters The next parameters deal with handling depth values of the basin.

`href` Reference depth. If the depth values of the basin and the water levels are referred to mean sea level, `href` should be 0 (default value). Else this value is subtracted from the given depth values. For example, if `href = 0.20` then a depth value in the basin of 1 meter will be reduced to 80 centimeters.

`hzmin` Minimum total water depth that will remain in a node if the element becomes dry. (Default 0.01 m)

`hzoff` Total water depth at which an element will be taken out of the computation because it becomes dry. (Default 0.05 m)

`hzon` Total water depth at which a dry element will be re-inserted into the computation. (Default 0.10 m)

`hmin` Minimum water depth (most shallow) for the whole basin. All depth values of the basin will be adjusted so that no water depth is shallower than `hmin`. (Default is no adjustment)

`hmax` Maximum water depth (deepest) for the whole basin. All depth values of the basin will be adjusted so that no water depth is deeper than `hmax`. (Default is no adjustment)

Bottom friction The friction term in the momentum equations can be written as Ru and Rv where R is the variable friction coefficient and u, v are the velocities in x, y direction respectively. The form of R can be specified in various ways. The value of `ireib` is choosing between the formulations. In the parameter input file a value λ is specified that is used in the formulas below. In a 2D simulation the Strickler (2) or the Chezy (3) formulation is the preferred option, while for a 3D simulation is is recommended to use the drag coefficient (5) or the roughness length formulation (6).

`ireib` Type of friction used (default 0):

0 No friction used

1 $R = \lambda$ is constant

2 λ is the Strickler coefficient. In this formulation R is written as $R = \frac{g}{C^2}\frac{|u|}{H}$ with $C = k_s H^{1/6}$ and $\lambda = k_s$ is the Strickler coefficient. In the above formula g is the gravitational acceleration, $|u|$ the modulus of the current velocity and H the total water depth.

3 λ is the Chezy coefficient. In this formulation R is written as $R = \frac{g}{C^2}\frac{|u|}{H}$ and $\lambda = C$ is the Chezy coefficient.

4 $R = \lambda/H$ with H the total water depth. This corresponds to a linear bottom friction.

5 λ is a constant drag coefficient and R is computed as $R = \lambda\frac{|u|}{H}$. This corresponds to a quadratic bottom friction.

6 λ is the bottom roughness length and R is computed through the formula $R = C\frac{|u|}{H}$ with $C = \left(\frac{0.4}{log(\frac{\lambda+0.5H}{\lambda})}\right)^2$

7 If $\lambda \geq 1$ it specifies the Strickler coefficient (`ireib=2`), otherwise it specifies a constant drag coefficient (`ireib=5`).

8 The bottom roughness length computed with sedtrans (sediment transport module) is used to compute the friction (similar to 6).

9 Experimental for fluid mud (no documentation).

10 Hybrid formulation switching between quadratic (5) and linear (4) bottom friction. The velocity below which linear friction is used has to be given in `uvmin`.

`czdef` The default value for the friction parameter λ. Depending on the value of `ireib` the coefficient λ is representing linear friction, a constant drag coefficient, the Chezy or Strickler parameter, or the roughness length. (default 0)

`iczv` Normally the bottom friction coefficient (such as Strickler, Chezy, etc.) is evaluated at every time step (`iczv` = 1). If for some reason this behavior is not desirable, `iczv` = 0 evaluates this value only before the first time step, keeping it constant for the rest of the simulation. Please note that this is only relevant if you have given more than one bottom friction value (inflow/outflow) for an area. The final value of R is computed at every time step anyway. (default 1)

`uvmin` Critical velocity for `ireib`=10 below which bottom friction will be used as linear. (Default 0.2)

The value of λ may be specified for the whole basin through the value of `czdef`. For more control over the friction parameter it can be also specified in section `area` where the friction parameter depending on the type of the element may be varied. Please see the paragraph on section `area` for more information.

Physical parameters The next parameters describe physical values that can be adjusted if needed.

`rowass` Average density of sea water. (Default 1025 $\mathrm{kg\,m^{-3}}$)

`roluft` Average density of air. (Default 1.225 $\mathrm{kg\,m^{-3}}$)

`grav` Gravitational acceleration. (Default 9.81 $\mathrm{m\,s^{-2}}$)

Wind parameters The next two parameters deal with the wind stress to be prescribed at the surface of the basin.
The wind data can either be specified in an external file (ASCII or binary) or directly in the parameter file in section `wind`. The ASCII file or the wind section contain three columns, the first giving the time in seconds, and the others the components of the wind speed. Please see below how the last two columns are interpreted depending on the value of `iwtype`. For the format of the binary file please see the relative section. If both a wind file and section `wind` are given, data from the file is used.
The wind stress is normally computed with the following formula

$$\tau^x = \rho_a c_D |u| u^x \quad \tau^y = \rho_a c_D |u| u^y \tag{C.1}$$

where ρ_a, ρ_0 is the density of air and water respectively, u the modulus of wind speed and u^x, u^y the components of wind speed in x, y direction. In this formulation c_D is a dimensionless drag coefficient that varies between $1.5 \cdot 10^{-3}$ and $3.2 \cdot 10^{-3}$. The wind speed is normally the wind speed measured at a height of 10 m.
`iwtype` The type of wind data given (default 1):

0 No wind data is processed

1 The components of the wind is given in [m/s]

2 The stress (τ^x, τ^y) is directly specified

3 The wind is given in speed [m/s] and direction [degrees]. A direction of 0^o specifies a wind from the north, 90^o a wind from the east etc.

4 As in 3 but the speed is given in knots

`itdrag` Formula to compute the drag coefficient.

0 constant value given in `dragco`.

1 Smith and Banke (1975) formula

2 Large and Pond (1981) formula

3 Spatial/temporal varying in function of wave. Need the coupling with WWMIII.

4 Spatial/temporal varying in function of heat flux. Only for `iheat` = 6.

5 Hersbach (2011) formula. This is a fit of kinematic viscosity for light wind and a Charnock coefficient for strong wind. Neutral wind should be used (small differences). See the paper and/or the ECMWF report.

(Default 0)

`dragco` Drag coefficient used in the above formula. (Default 2.5E-3). If `itdrag` = 5 this is the Charnock parameter and you should use values from 0.01 (swell) to 0.04 (steep young waves). (Default 0.025). Please note that in case of `iwtype` = 2 this parameter is of no interest, since the stress is specified directly.

`wsmax` Maximum wind speed allowed in [m/s]. This is in order to avoid errors if the wind data is given in a different format from the one specified by `iwtype`. (Default 50)

`wslim` Limit maximum wind speed to this value [m/s]. This provides an easy way to exclude strong wind gusts that might blow up the simulation. Use with caution. (Default -1, no limitation)

Meteo and heat flux parameters The next parameters deal with the heat and meteo forcing.

`iheat` The type of heat flux algorithm (Default 1):

1 As in the AREG model

2 As in the POM model

3 Following A. Gill

4 Following Dejak

5 As in the GOTM model

6 Using the COARE3.0 module

7 Read heat fluxes directly from file. The columns in the data file must be "time srad qsens qlat qlong".

8 MFS heat fluxes as Pettenuzzo et al., 2010

Except when `iheat` is 7, the time series file has the columns "time srad airt rhum cc".

`ihtype` Different ways of how to specify water vapor content are possible. Normally relative humidity has to be given (`ihtype`=1). However, also wet bulb temperature (`ihtype`=2), dew point temperature (`ihtype`=3), or specific humidity (`ihtype`=4) can be given. (Default 1).

`isolp` The type of solar penetration parameterization by one or more exponential decay curves. `isolp` = 0 sets an e-folding decay of radiation (one exponential decay curve) as function of depth `hdecay`. `isolp` = 1 sets a profile of solar radiation with two length scale of penetration. Following the Jerlov (Jerlov, N. G., 1968 Optical Oceanography, Elsevier, 194pp) classification the type of water is clear water (type I). (Default 0)

`iwtyp` The water types from clear water (type I) to the most turbid water (coastal water 9) following the classification of Jerlov (Jerlov, N. G., 1968 Optical Oceanography, Elsevier, 194pp). The possible values for `iwtyp` are:

0 clear water type I

1 type IA

2 type IB

3 type II

4 type III

5 type 1

6 type 3

7 type 5

8 type 7

9 type 9

`hdecay` Depth of e-folding decay of radiation [m]. If `hdecay` = 0 everything is absorbed in first layer (Default 0).

`botabs` Heat absorption at bottom [fraction] (Default 0).

=0 everything is absorbed in last layer

=1 bottom absorbs remaining radiation

`albedo` General albedo (Default 0.06).

`albed4` Albedo for temp below 4 degrees (Default 0.06).

`ievap` Compute evaporation mass flux (Default 0).

Parameters for 3d The next parameters deal with the layer structure in 3D.

`dzreg` Normally the bottom of the various layers are given in section `$levels`. If only a regular vertical grid is desired then the parameter `dzreg` can be used. It specifies the spacing of the vertical layers in meters. (Default is 0, which means that the layers are specified explicitly in `$levels`.

The last layer (bottom layer) is treated in a special way. Depending on the parameter `ilytyp` there are various cases to be considered. A value of 0 leaves the last layer as it is, even if the thickness is very small. A value of 1 will always eliminate the last layer, if it has not full layer thickness. A value of 2 will do the same, but only if the last layer is smaller than `hlvmin` (in units of fraction). Finally, a value of 3 will add the last layer to the layer above, if its layer thickness is smaller than `hlvmin`.

`ilytyp` Treatment of last (bottom) layer. 0 means no adjustment, 1 deletes the last layer, if it is not a full layer, 2 only deletes it if the layer thickness is less than `hlvmin`, and 3 adds the layer thickness to the layer above if it is smaller than `hlvmin`. Therefore, 1 and 2 might change the total depth and layer structure, while 3 only might change the layer structure. The value of 1 will always give you full layers at the bottom. (Default 3)

`hlvmin` Minimum layer thickness for last (bottom) layer used when `ilytyp` is 2 or 3. The unit is fractions of the nominal layer thickness. Therefore, a value of 0.5 indicates that the last layer should be at least half of the full layer. (Default 0.25)

With $z-$layers the treatment of the free-surface must be addressed. What happen if the water level falls below the first z-level? A $z-$star type vertical grid deformation can be deployed. The next parameter specify the number of surface layers that are moving.

`nzadapt` Parameter that controls the number of surface $z-$layers that are moving. The value `nzadapt` $= 0$ corresponds to standard $z-$layers (Default). Then, some care is needed to define the first interface sufficiently deep to avoid the well-known "drying" of the first layer. The value of `nzadapt` $= N_{tot}$, with N_{tot} the total number of z-layers, is $z-$star (all layers are moving). Other values of $0 <$ `nzadapt` $< N_{tot}$ corresponds to move, at minimum, the first `nzadapt` surface layers with $z-$star. These feature is still experimental.

The above parameters are dealing with zeta layers, where every layer has constant thickness, except the surface layer which is varying with the water level. The next parameters deal with sigma layers where all layers have varying thickness with the water level.

`nsigma` Number of sigma layers for the run. This parameter can be given in alternative to specifying the sigma layers in `$levels`. Only regularly spaced sigma levels will be created. (Default 0)

`hsigma` This is still an experimental feature. It allows to use sigma layers above zeta layers. `hsigma` is the depth where the transition between these two types of layers is occurring. (Default 10000)

The next parameters deal with vertical diffusivity and viscosity.

`difmol` Vertical molecular diffusivity parameter for temperature, salinity, and tracer. (Default 1.0e-06)

`diftur` Vertical turbulent diffusivity parameter for temperature, salinity, and tracer. (Default 0)

`vismol` Vertical molecular viscosity parameter for momentum. (Default 1.0e-06)

`vistur` Vertical turbulent viscosity parameter for momentum. (Default 0)

Instead of setting fixed values for viscosity and diffusivity, these parameters can also be

computed by a turbulence closure scheme. The parameter iturb defines what turbulence scheme is going to be used. A value of 0 uses no turbulence scheme. In this case be sure that vistur and diftur have been set manually. iturb=1 uses the GOTM routines for turbulence. In order to use this value SHYFEM has to be compiled with GOTM support. iturb=2 uses a k-epsilon module, and a value of 3 uses the Munk-Anderson model. The recommended value for iturb is 1 (GOTM module). (Default 0)
The next parameters deal with horizontal diffusion.

dhpar Horizontal diffusion parameter (general). (Default 0)

The next parameters deal with the control of the scalar transport and diffusion equation. You have possibility to prescribe the tvd scheme desired and to limit the Courant number.

itvd Type of the horizontal advection scheme used for the transport and diffusion equation. Normally an upwind scheme is used (0), but setting the parameter itvd to a value greater than 0 choses a TVD scheme. A value of 1 will use a TVD scheme based on the average gradient, and a value of 2 will use the gradient of the upwind node (recommended). This feature is still experimental, so use with care. (Default 0)

itvdv Type of the vertical advection scheme used for the transport and diffusion equation. Normally an upwind scheme is used (0), but setting the parameter itvdv to 1 choses a TVD scheme. This feature is still experimental, so use with care. (Default 0)

rstol Normally the internal time step for scalar advection is automatically adjusted to produce a Courant number of 1 (marginal stability). You can set rstol to a smaller value if you think there are stability problems. (Default 1)

Various parameters The next parameters describe various parameters not related to the above parameters.

tauvel If you have velocity observations given in file surfvel then you can specify the relaxation parameter τ in the variable tauvel. (Default 0, which means no assimilation of velocities)

rtide If rtide = 1 the model calculates equilibrium tidal potential and load tides and uses these to force the free surface (Default 0).

ltidec Calibration factor for calculating the loading tide, which is computed in function of the total water depth as $\beta = ltidec * H$. Usually it has a value of order 1e-6. If 0 no loading tide is computed (Default 0).

ibstrs Call parameter for the routine bstress. If equal to 1 it computes (in function of currents and waves) and writes the bottom shear stess into a .bstress.shy file. If 0 no bottom stess is computed (Default 0).

Temperature and salinity The next parameters deal with the transport and diffusion of temperature and salinity (T/S).

`ibarcl` In order to compute T/S the parameter `ibarcl` must be different from 0. Different values indoicate different ways to compute the advection and diffusion of the variables T/S and how their values is used in the momentum equation. (Default 0)

0 No computation of T/S.

1 Computation of T/S. The T/S field has a feedback through the baroclinic term on the momentum equation. This corresponds to a full baroclinic model.

2 The model runs in diagnostic mode. No advection of the T/S field is done, but the baroclinic term is computed and used in the momentum equations. For this to work T/S fields have to be provided in external files `tempobs` and `saltobs`.

3 The model computes the advection and diffusion of T/S, but their value is not used in the baroclinic terms in momentum equation. This corresponds to treating T/S as tracers.

4 The model runs in baroclinic mode, but uses nudging for a basic data assimilation of T/S. For this to work T/S fields have to be provided in external files `tempobs` and `saltobs`. Moreover, a time scale for the nudging has to be provided, either as a constant using `temptaup` and `salttaip` or in spatially and temporarily varying fields given in external files `temptau` and `salttau`.

In case `ibarcl` is different from 0 the variables T/S will be computed. However, they may be selectively turned off setting one of the two parameters `itemp` or `isalt` explicitly to 0.

`itemp` Flag if the computation on the temperature is done. A value different from 0 computes the transport and diffusion of the temperature. (Default 1)

`isalt` Flag if the computation on the salinity is done. A value different from 0 computes the transport and diffusion of the salinity. (Default 1)

The next parameters set the initial conditions for temperature and salinity. Both the average value and and a stratification can be specified. Initial conditions can also be given in external files `tempin` and `saltin`.

`temref` Reference (initial) temperature of the water in centigrade. (Default 0)

`salref` Reference (initial) salinity of the water in psu (practical salinity units) or ppt. (Default 0)

`tstrat` Initial temperature stratification in units of [C/km]. A positive value indicates a stable stratification. (Default 0)

`sstrat` Initial salinity stratification in units of [psu/km]. A positive value indicates a stable stratification. (Default 0)

The next parameters deal with horizontal diffusion of temperature and salinity. These parameters overwrite the general parameter for horizontal diffusion `dhpar`.

`thpar` Horizontal diffusion parameter for temperature. (Default 0)

`shpar` Horizontal diffusion parameter for salinity. (Default 0)

When using nudging (`ibarcl=4`) time scales for nudging have to given. They can be given in the parameters `temptaup` and `salttaup` or in external files `temptau` and `salttau`.

`temptaup` Time scale for temperature nudging in seconds. (Default 0)

`salttaup` Time scale for salinity nudging in seconds. (Default 0)

Concentrations The next parameters deal with the transport and diffusion of a conservative substance. The substance is dissolved in the water and acts like a tracer.

`iconz` Flag if the computation on the tracer is done. A value different from 0 computes the transport and diffusion of the substance. If greater than 1 `iconz` concentrations are simulated. (Default 0)

`conref` Reference (initial) concentration of the tracer in any unit. (Default 0)

`taupar` Decay rate for concentration if different from 0. In this case `taupar` is the decay rate (e-folding time) in days. This parameter is also used for multi-concentration runs. In this case either one value has to be given that is used for all concentrations, or `iconz` values have to be given, one for each concentration. (Default 0)

`idecay` Type of decay used. If 0 no decay is used. A value of 1 uses the value of `taupar` as exponential decay. A value of 2 uses a formulation of Chapra, where the decay rate depends on T,S,light and settling. In this case the value of `taupar` is ignored. (Default 0)

`chpar` Horizontal diffusion parameter for the tracer. This value overwrites the general parameter for horizontal diffusion `dhpar`. (Default 0)

Output for scalars The next parameters define the output frequency of the computed scalars (temperature, salinity, generic concentration) to file.

`idtcon`
`itmcon` Time step and start time for writing to file conz.shy (concentration) and ts.shy (temperature and salinity).

`irho` Flag if the density is also written together with T/S. A value different from 0 writes the density to file. (Default 0)

C.1.3 Section `$proj`

Section `$proj` handles the projection from cartesian to geographical coordinate system. If `iproj` >0 the projected geographical coordinates can be used for computing spatially variable Coriolis parameter and tidal potential even if the basin is in cartesian coordinate system (`isphe` = 0) .
Please find all details here below.

`iproj` Switch that indicates the type of projection (default 0):

0 do nothing

1 Gauss-Boaga (GB)

2 Universal Transverse Mercator (UTM)

3 Equidistant cylindrical (CPP)

4 UTM non standard

`c_fuse` Fuse for GB (1 or 2, default 0)

`c_zone` Zone for UTM (1-60, default 0)

`c_lamb` Central meridian for non-std UTM (default 0)

`c_x0` x0 for GB and UTM (default 0)

`c_y0` y0 for GB and UTM (default 0)

`c_skal` Scale factor for non-std UTM (default 0.9996)

`c_phi` Central parallel for CPP (default 0.9996)

`c_lon0` Longitude origin for CPP (default 0)

`c_lat0` Latitude origin for CPP (default 0)

C.1.4 Section `$waves`

Parameters in section `$waves` activate the wind wave module and define which kind of wind wave model has to be used. These parameters must be in section `waves`.

`iwave` Type of wind wave model and coupling procedure (default 0):

0 No wind wave model called

1 The parametric wind wave model is called (see file subwave.f)

>1 The spectral wind wave model WWMIII is called

2 ... wind from SHYFEM, radiation stress formulation

3 ... wind from SHYFEM, vortex force formulation

4 ... wind from WWMIII, radiation stress formulation

5 ... wind from WWMIII, vortex force formulation

11 The spectral wind wave model WaveWatch WW3 is called

When the vortex force formulation is chosen the wave-supported surface stress is subtracted from the wind stress, in order to avoid double counting of the wind forcing in the flow model. Moreover, the use of the wave-depended wind drag coefficient could be adopted setting `itdrag` = 3.

`dtwave` Time step for coupling with WWMIII. Needed only for `iwave` > 1 (default 0).

`idtwav` Time step and start time for writing to file wav, the
`itmwav` files containing output wave variables (significant wave height, wave period, mean wave direction).

C.1.5 Section `$sedtr`

The following parameters activate the sediment transport module and define the sediment grainsize classes to the simulated.

`sedtr` Sediment transport module section name.

`isedi` Flag if the computation on the sediment is done:

0 Do nothing (default)

1 Compute sediment transport

`idtsed` Time step and start time for writing to files sed e sco, the
`itmsed` files containing sediment variables and suspended sediment concentration.

`sedgrs` Sediment grainsize class vector [mm]. Values has be ordered from the finest to the more coarse.
`example: sedgrs = '0.1 0.2 0.3 0.4'`

`irocks` Element type where do not compute erosion-deposition (Default -1).

`sedref` Initial sediment reference concentration [kg/m3] (Default 0).

`sedhpar` Sediment diffusion coefficient (Default 0).

`adjtime` Time for sediment initialization [s]. The sediment model needs a initialization time in which the system goes to a quasi steady state. When t = `adjtime` the bed evolution change in the output is reset. Keep in mind that `adjtime` has to be chosen case by case in function of the morphology and the parameters used for the simulation (Default 0).

`percin` Initial sediment distribution (between 0 and 1) for each grainsize class. The sum of percin must be equal to 1.
`example: percin = 0.25 0.25 0.25 0.25`
If percin is not selected the model impose equal percentage for each grainsize class (percin = 1/nrs). In case of spatial differentiation of the sediment distribution set a number of percin equal to the number of grainsize classes per the number of area types. Element types should be numbered consecutively starting from 0.
`example: percin = 0.25 0.25 0.25 0.25`
`0.20 0.20 0.30 0.30`
`0.45 0.15 0.15 0.15`

`tauin` Initial dry density or TAUCE. In function of the value:
0-50 : critical erosion stress (Pa)
>50 : dry bulk density of the surface (kg/m**3).
In case of spatial differentiation set a number of tauin equal to the number of area type. Element types should be numbered consecutively starting from 0.
`example: tauin = '0.9 1.4 2.5 1.1'`

`sedp` File containing spatially varying initial sediment distribution for each grid node. Values are in percentage of each class and the file should be structured with number of columns equal the number of grainsize classes and the number of row equal the number of nodes (the order should follow the internal node numbering).

`sedt` File containing spatially varying initial critical erosion stress (Pa) or dry bulk density (kg/m3). One value for each node (the order should follow the internal node numbering).

`sedcon` File containing the additional constants used in sediment model. These parameters are usually set to the indicated default values, but can be customized for each sediment transport simulation. The full parameter list together with their default value and brief description is reported in Table C.1. Most of the parameter, especially the ones for the cohesive sediments, have been calibrated for the Venice Lagoon. For more information about these parameters please refer to Neumeier et al. [14] and Ferrarin et al. [6].

Additional parameters The full parameter list is reported in Table C.1. An example of the settings for the `sedcon` file is given in Fig. 4.9. Please note that is not necessary to define all parameters. If not defined the default value is imposed.

```
IOPT = 3
SURFPOR = 4
DOCOMPACT = 1
```

Figure C.2: Example of the secon file.

C.1.6 Section `$wrt`

Section `$wrt` contains parameters for computing water renewal time. During runtime if writes a .jas file with time series of total tracer concentration in the basin and WRT computed according to different methods. Nodal values of computed WRT are written in the .wrt file. Frequency distributions of WRTs are written in the .frq file.
Please find all details here below.

`idtwrt` Time step to reset concentration to c0. Use 0 if no reset is desired. Use -1 if no renewal time computation is desired (Default -1).

`itmin` Time from when to compute renewal time (-1 for start of sim) (Default -1)

`itmax` Time up to when to compute renewal time (-1 for end of sim) (Default -1).

`c0` Initial concentration of tracer (Default 1).

`iaout` Area code of elements out of lagoon (used for init and retflow). Use -1 to if no outside areas exist. (Default -1).

`percmin` Percentage to reach after which the computation is stopped. Use 0 if no premature end is desired (Default 0).

`iret` Equal to 1 if return flow is used. If equal to 0 the concentrations outside are explicitly set to 0 (Default 1).

`istir` If equal to 1 simulates completely stirred tank (replaces at every time step conz with average conz) (Default 0).

`iadj` Adjust renewal time for tail of distribution (Default 1).

`ilog` Use logarithmic regression to compute renewal time (Default 0).

`ctop` Maximum to be used for frequency curve (Default 0).

`ccut` Cut renewal time at this level (for res time computation) (Default 0).

`wrtrst` If reset times are not regularly distributed (e.g., 1 month) it is possible to give the exact times when a reset should take place. `wrtrst` is a file name where these reset times are specified, one for each line. For every line two integers indicating date and time for the reset must be specified. If only one value is given, time is taken to be 0. The format of date is "YYYYMMDD" and for time "hhmmss". If the file wrtrst is given `idtwrt` should be 0.

C.1.7 Section `$lagrg`

Section `$lagrg` describes the use of the Lagrangian Particle Module.
The lagrangian particles can be released:

- inside the given areas (filename `lgrlin`). If this file is not specified they are released over the whole domain. The amount of particles released and the time step are specified by `nbdy` and `idtl`.
- at selected times and location, e.g. along a drifter track (filename `lgrtrj`). `nbdy` particles are released at the times and location specified in the file.
- as initial particle distribution (filename `lgrini`) at time `itlgin`. This file has the same format as the lagrangian output.
- at the open boundaries either as particles per second or per volume flux (parameter `lgrpps`).

`lgrlin` and `lgrtrj` are mutually exclusive.
The lagrangian module runs between the times `itlanf` and `itlend`. If one or both are missing, the simulation extremes are substituted. Inside the lagrangian simulation window, the release of particles inside a given area is controlled by the parameters `idtl`, `itranf` and `itrend`. `itranf` gives the time of the first release, `itrend` the time for the last release. If not given they are set equal to the extremes of the lagrangian simulation. `idtl` is giving the time step of release.
The output frequency of the results can be controlled by `idtlgr` and `itmlgr`.
Please find all details here below.

`ilagr` Switch that indicates that the lagrangian module should be run (default 0):

0 do nothing

1 surface lagrangian

2 2d lagrangian

3 3d lagrangian

`nbdymax` Maximum numbers of particles that can be in the domain. This should be the maximum number of particles that can be created and inserted. Use 0 to not limit the number of particles (on your own risk). This parameter must be set and has no default.

`nbdy` Total numbers of particles to be released in the domain each time a release of particles takes place. (Default 0)

`rwhpar` A horizontal diffusion can be defined for the lagrangian model. Its value can be specified in `rwhpar` and the units are [m**2/s]. If `rwhpar` ¡ 0 the diffusion parameter depends on the local diffusivity (see `idhtyp`) (Default 0)

`itlanf` `itlend` The start and end time for the lagrangian module. If not given, the module runs for the whole simulation.

`itmlgr` `idtlgr` Initial time and time step for the output to file of the particles. if `idtlgr` is 0, no output is written. (Default 0)

`idtl` The time step used for the release of particles. If this is 0 particles are released only once at the beginning of the lagrangian simulation. No particles are released for a value of less than 0. (Default 0)

`itranf` `itrend` Initial and final time for the release of particles. If not specified the particles are released over the whole lagrangian simulation period.

`ipvert` Set the vertical distribution of particles:

0 releases one particles only in surface layer

$>$**0** release n particles regularly

$<$**0** release n particles randomly

`linbot` Set the bottom layer for vertical releases (Default -1, bottom layer)

`lintop` Set the top layer for vertical releases (Default 1, surface layer)

`stkpar` Calibration parameter for parameterizing the stokes drift induced by waves (and wind). Only affect particle of the sea surface (layer = 1). The wind file is needed even in offline mode (Default 0).

`dripar` Parameter to account for drifter inertia by multiplying the advective transports. Usually it assumes values between 0.9 and 1.2 (Default 1).

`lbeach` Parameter to account for particles beaching on the shore. It assumes values between 0 (no beaching) and 1 (Default 0).

`lgrlin` File name that contains closed lines of the area where the particles have to be released. If not given, the particles are released over the whole domain.

`lgrtrj` File name that contains a drifter trajectory with time (yyyy-mm-dd::HH:MM:SS) and position (x and y) of release of `nbdy` particles.

`lgrini` File name that contains initial particle distribution. It has the same format as the lagrangian output.

`itlgin` Time to use for the initialization of the particle distribution from file (`lgrini`).

C.1.8 Section `$name`

In section `$name` the names of input files can be given. All directories default to the current directory, whereas all file names are empty, i.e., no input files are given.

File names Strings in section `$name` enable the specification of files that account for initial conditions or forcing.

`zinit` Name of file containing initial conditions for water level

`uvinit` Name of file containing initial conditions for velocity

`wind` File with wind data. The file may be either formatted or unformatted. For the format of the unformatted file please see the section where the WIN file is discussed. The format of formatted ASCII file is in standard time-series format, with the first column containing the time in seconds and the next two columns containing the wind data. The meaning of the two values depend on the value of the parameter `iwtype` in the `para` section.

`qflux` File with heat flux data. This file must be in a special format to account for the various parameters that are needed by the heat flux module to run. Please refer to the information on the file `qflux`.

`rain` File with rain data. This file is a standard time series with the time in seconds and the rain values in mm/day. The values may include also evaporation. Therefore, also negative values (for evaporation) are permitted.

`ice` File with ice cover. The values range from 0 (no ice cover) to 1 (complete ice cover).

`surfvel` File with surface velocities from observation. These data can be used for assimilation into the model.

`restrt` Name of the file if a restart is to be performed. The file has to be produced by a previous run with the parameter `idtrst` different from 0. The data record to be used in the file for the restart must be given by time `itrst`.

`gotmpa` Name of file containing the parameters for the GOTM turbulence model (iturb = 1).

`tempin` Name of file containing initial conditions for temperature

`saltin` Name of file containing initial conditions for salinity

`conzin` Name of file containing initial conditions for concentration

`tempobs` Name of file containing observations for temperature

`saltobs` Name of file containing observations for salinity

`temptau` Name of file containing the time scale for nudging of temperature

`salttau` Name of file containing the time scale for nudging of salinity

`bfmini` Name of file containing initial conditions for bfm

`offlin` Name of the file if a offline is to be performed. The file has to be produced by a previous run with the parameter `idtoff` greater than 0.

C.1.9 Section `$bound`

These parameters determine the open boundary nodes and the type of the boundary: level or flux boundary. At the first the water levels are imposed, on the second the fluxes are prescribed.
There may be multiple sections `bound` in one parameter input file, describing all open boundary conditions necessary. Every section must therefore be supplied with a boundary number. The numbering of the open boundaries must be increasing. The number of the boundary must be specified directly after the keyword `bound`, such as `bound1` or `bound 1`.

`kbound` Array containing the node numbers that are part of the open boundary. The node numbers must form one contiguous line with the domain (elements) to the left. This corresponds to an anti-clockwise sense. The type of boundary depends on the value of `ibtyp`. In case this value is 1 or 2 at least two nodes must be given.

`ibtyp` Type of open boundary.

0 No boundary values specified

1 Level boundary. At this open boundary the water level is imposed and the prescribed values are interpreted as water levels in meters. If no value for `ibtyp` is specified this is the default.

2 Flux boundary. Here the discharge in $m^3\ s^{-1}$has to be prescribed.

3 Internal flux boundary. As with `ibtyp = 2` a discharge has to be imposed, but the node where discharge is imposed can be an internal node and need not be on the outer boundary of the domain. For every node in `kbound` the volume rate specified will be added to the existing water volume. This behavior is different from the `ibtyp = 2` where the whole boundary received the discharge specified.

4 Momentum input. The node or nodes may be internal. This feature can be used to describe local acceleration of the water column. The unit is force / density [$m^4\ s^{-2}$]. In other words it is the rate of volume [$m^3\ s^{-1}$] times the velocity [m/s] to which the water is accelerated.

`iqual` If the boundary conditions for this open boundary are equal to the ones of boundary `i`, then setting `iqual = i` copies all the values of boundary `i` to the actual boundary. Note that the value of `iqual` must be smaller than the number of the actual boundary, i.e., boundary `i` must have been defined before. (This feature is temporarily not working; please do not use.)

The next parameters give a possibility to specify the file name of the various input files that are to be read by the model. Values for the boundary condition can be given at any time step. The model interpolates in between given time steps if needed. The grade of interpolation can be given by `intpol`.
All files are in ASCII and share a common format. The file must contain two columns, the first giving the time of simulation in seconds that refers to the value given in the second column. The value in the second column must be in the unit of the variable that is given. The time values must be in increasing order. There must be values for the whole simulation, i.e., the time value of the first line must be smaller or equal than the start of the simulation, and the time value of the last line must be greater or equal than the end of the simulation.

`boundn` File name that contains values for the boundary condition. The value of the variable given in the second column must be in the unit determined by `ibtyp`, i.e., in meters for a level boundary, in $m^3\ s^{-1}$for a flux boundary and in $m^4\ s^{-2}$for a momentum input.

`zfact` Factor with which the values from `boundn` are multiplied to form the final value of the boundary condition. E.g., this value can be used to set up a quick sensitivity run by multiplying all discharges by a factor without generating a new file. (Default 1)

`levmin` `levmax` A point discharge normally distributes its discharge over the whole water column. If it is important that in a 3D simulation the water mass discharge is concentrated only in some levels, the parameters `levmin` and `levmax` can be used. They indicate the lowest and deepest level over which the discharge is distributed. Default values are 0, which indicate that the discharge is distributed over the whole water column. Setting only `levmax` distributes from the surface to this level, and setting only `levmin` distributes from the bottom to this level.

`conzn` `tempn` `saltn` File names that contain values for the respective boundary condition, i.e., for concentration, temperature and salinity. The format is the same as for file `boundn`. The unit of the values given in the second column must the ones of the variable, i.e., arbitrary unit for concentration, centigrade for temperature and psu (per mille) for salinity.

`vel3dn` File name that contains current velocity values for the boundary condition. The format is the same as for file `tempn` but it has two variables: current velocity in x and current velocity in y. Velocity can be nudged or imposed depending on the value of `tnudge` (mandatory). The unit is [m/s].

`tnudge` Relaxation time for nudging of boundary velocity. For `tnudge` = 0. velocities are imposed, for `tnudge` ¿ 0. velocities are nudged. The default is -1 which means do nothing. Unit is [s]. (Default -1)

The next variables specify the name of the boundary value file for different modules. Please refer to the documentation of the single modules for the units of the variables.

`bio2dn` File name that contains values for the ecological module (EUTRO-WASP).

`sed2dn` File name that contains values for the sediment transport module. The unit of the values given in the second and following columns (equal to the number of defined grainsize in parameter `sedgrs`).

`mud2dn` File name that contains values for the fluid mud module.

`lam2dn` File name that contains values for the fluid mud module (boundary condition for the structural parameter, to be implemented).

`dmf2dn` File name that contains values for the fluid mud module (boundary conditions for the advection of flocsizes, to be implemented).

`tox3dn` File name that contains values for the toxicological module.

`bfmbcn` File name that contains values for the bfm module.

`mercn` File name that contains values for the mercury module.

`s4mern` File name that contains values for the mercury module.

`intpol` Order of interpolation for the boundary values read in files. Use for 1 for stepwise (no) interpolation, 2 for linear and 4 for cubic interpolation. The default is linear interpolation, except for water level boundaries (`ibtyp=1`) where cubic interpolation is used.

The next parameters can be used to impose a sinusoidal water level (tide) or flux at the open boundary. These values are used if no boundary file `boundn` has been given. The values must be in the unit of the intended variable determined by `ibtyp`.

`ampli` Amplitude of the sinus function imposed. (Default 0)

`period` Period of the sinus function. (Default 43200, 12 hours)

`phase` Phase shift of the sinus function imposed. A positive value of one quarter of the period reproduces a cosine function. (Default 0)

`zref` Reference level of the sinus function imposed. If only `zref` is specified (`ampli = 0`) a constant value of `zref` is imposed on the open boundary.

With the next parameters a constant value can be imposed for the variables of concentration, temperature and salinity. In this case no file with boundary values has to be supplied. The default for all values is 0, i.e., if no file with boundary values is supplied and no constant is set the value of 0 is imposed on the open boundary. A special value of -999 is also allowed. In this case the value imposed is the ambient value of the parameter close to the boundary.

`conz` Constant boundary values for concentration, temperature
`temp` and salinity respectively. If these values are set no boundary file has to be supplied. (Default 0)
`salt`

The next two values are used for constant momentum input. This feature can be used to describe local acceleration of the water column. The values give the input of momentum in x and y direction. The unit is force / density ($m^4\,s^{-2}$). In other words it is the rate of volume ($m^3\,s^{-1}$) times the velocity (m/s) to which the water is accelerated.
These values are used if boundary condition `ibtyp = 4` has been chosen and no boundary input file has been given. If the momentum input is varying then it may be specified with the file `boundn`. In this case the file `boundn` must contain three columns, the first for the time, and the other two for the momentum input in x, y direction.
Please note that this feature is temporarily not available.

`umom` Constant values for momentum input. (Default 0)
`vmom`

The next two values can be used to achieve the tilting of the open boundary if only one water level value is given. If only `ktilt` is given then the boundary values are tilted to be in equilibrium with the Coriolis force. This may avoid artificial currents along the boundary. `ktilt` must be a boundary node on the boundary.
If `ztilt` is given the tilting of the boundary is explicitly set to this value. The tilting of the first node of the boundary is set to $-$`ztilt` and the last one to $+$`ztilt`. The total amount of

tilting is therefore is $2 \cdot$ ztilt. If ktilt is not specified then a linear interpolation between the first and the last boundary node will be carried out. If also ktilt is specified then the boundary values are arranged that the water levels are tilted around ktilt, e.g., $-$ztilt at the first boundary node, 0 at ktilt, and $+$ztilt at the last boundary node.

ktilt — Node of boundary around which tilting should take place. (Default 0, i.e., no tilting)

ztilt — Explicit value for tilting (unit meters). (Default 0)

Other parameters:

igrad0 — If different from 0 a zero gradient boundary condition will be implemented. This is already the case for scalars under outflowing conditions. However, with igrad0 different from 0 this conditions will be used also for inflow conditions. (Default 0)

tramp — Use this value to start smoothly a discharge boundary condition. If set it indicates the time (seconds) that will be used to increase a discharge from 0 to the desired value (Default 0)

levflx — If discharge is depending on the water level (e.g., lake outflow) then this parameter indicates to use one of the possible outflow curves. Please note that the flow dependence on the water level must be programmed in the routine level_flux(). (Default 0)

nad — On the open boundaries it is sometimes convenient to not compute the non-linear terms in the momentum equation because instabilities may occur. Setting the parameter nad to a value different from 0 indicates that in the first nad nodes from the boundary the non linear terms are switched off. (Default 0)

lgrpps — Indicates the number of particles released at the boundary for the lagrangian module. If positive it is the number of particles per second released along the boundary. If negative its absolute value indicates the particles per volume flux (unit $m^3\ s^{-1}$) released along the boundary. (Default 0)

C.1.10 Section $wind

In this section the wind data can be given directly without the creation of an external file. Note, however, that a wind file specified in the name section takes precedence over this section. E.g., if both a section wind and a wind file in name is given, the wind data from the file is used.
The format of the wind data in this section is the same as the format in the ASCII wind file, i.e., three columns, with the first specifying the time in seconds and the other two columns giving the wind data. The interpretation of the wind data depends on the value of iwtype. For more information please see the description of iwtype in section para.

C.1.11 Section `$extra`

In this section the node numbers of so called "extra" points are given. These are points where the value of simulated variables (water level, velocities, temperature, salinity, tracer, etc.) are written to create a time series that can be elaborated later. The output for these "extra" points consumes little memory and can be therefore written with a much higher frequency (typically the same as the integration time step) than the complete hydrodynamic output. The output is written to file EXT.
The format of the section is the following:

```
$extra
node1   'string1'
node2   'string2'
etc..
$end
```

where node is the node number and string is a description of the node. If no description strings are needed the nodes can also be specified by just giving their values:

```
$extra
node1  node2  node3
node4  etc..
$end
```

This format is however deprecated.

C.1.12 Section `$flux`

In this section transects are specified through which the discharge of water is computed by the program and written to file FLX. The transects are defined by their nodes through which they run. All nodes in one transect must be adjacent, i.e., they must form a continuous line in the FEM network.
The nodes of the transects are specified in free format and are ended with the description of the section. An example is given here:

```
$flux
1001 1002 1004 'section 1'
35 37 46 'special section'
407
301 'section given on two lines'
$end
```

The example shows the definition of 3 transects. As can be seen, the nodes of the transects can be given on one line alone (first transect), or on more than one lines (transect 3). There is also an old format that seperates one section from the other by inserting the value 0. However, this format is deprecated.

C.2 Parameter list for the post processing routines

The format of the parameter input file is the same as the one for the main routine. Please see this section for more information on the format of the parameter input file.
Some sections of the parameter input file are identical to the sections used in the main routine. For easier reference we will repeat the possible parameters of these section here.

C.2.1 Section `$title`

This section must be always the first section in the parameter input file. It contains only three lines. An example has been given already in figure C.1.
The only difference with respect to the `$para` section of the main routine is the first line. Here any description of the output can be used. It is just a way to label the parameter file. The other two line with the name of simulation and the basin are used to open the files needed for plotting.

C.2.2 Section `$para`

The parameters in section `$para` set generic values for the plot.
Some of the parameters set coordinates in the plot. For example, the values `x0, y0` and `x1, y1` indicate the actual plotting area, which can be bigger or smaller than the extension of the numerical grid.
Normally, values have to be in meters (the same as the coordinates in the numerical grid). However, also relative coordinates can be used. If all values given are in the range between -1 and +2, then these values are interpreted as relative coordinates. Therefore, *x* coordinates of 0 indicate the left border, and 1 the right border. The upper left quarter of the domain can be chosen with (`x0, y0`) = (0,0.5) and (`x1, y1`) = (0.5,1).

`x0`
`y0` — Lower left corner of the plotting area. (Default is whole area)

`x1`
`y1` — Upper right corner of the plotting area. (Default is whole area)

The next values give the position, where the legend (scale bar and true north) is plotted. This legend will only be plotted if the coordinates are not geographical (lat/lon) but cartesian.

`x0leg`
`y0leg` — Lower left corner of the area where the legend is plotted.

`x1leg`
`y1leg` — Upper right corner of the area. where the legend (north and scale) is plotted.

`lblank` — The legend is plotted over a white rectangle. Sometimes this blanking is not desirable. If you do not want to have a white box below the legend set `lblank` to 0. (Default 1)

`cislnd` — It is possible to plot all islands in gray color. Setting `cislnd` to a value between 0 (black) and 1 (white) will achieve this. A negative value will not fill islands with gray color. (Default -1)

`dgray` — It is possible to plot all dry areas in gray color. Setting `dgray` to a value between 0 (black) and 1 (white) will achieve this. A negative value will not fill dry areas with gray color. (Default -1)

`hgray` — Whereas `dgray` is normally only coloring elements that are dry, you can also color elements shallower than a given depth `hgray`. E.g., a value for `hgray` of -0.5 will plot in gray all elements with depth lower than -0.5 m (salt marshes). (Default -10000)

`dxygrd` Grid size if the results are interpolated on a regular grid. A value of 0 does not use a regular grid but the original finite element grid for plotting. (Default 0)

`typls` Typical length scale to be used when scaling velocity or transport arrows. If `dxygrd` is given this length is used and `typls` is not used. If not given it is computed from the basin parameters. (Default 0)

`typlsf` Additional factor to be used with typls to determine the length of the maximum or reference vector. This is the easiest way to scale the velocity arrows with an overall factor. (Default 1)

`velref` Reference value to be used when scaling arrows. If given, a vector with this value will have a length of `typls*typlsf` on the map, or, in case `dxygrd` is given, `dxygrd*typlsf`. If not set the maximum value of the velocity/transport will be used as `velref`. (Default 0)

`velmin` Minimum value for which an arrow will be plotted. With this value you can eliminate small arrows in low dynamic areas. (Default 0)

`velmax` Maximum value for which an arrow will be plotted. With this value you can eliminate arrows that are too big. This is useful if you would like to study an area with low current speed but adjacent area have high current speeds that would overplot the area. (Default -1, no limitation)

`isphe` If 0 a cartesian coordinate system is used, If 1 the coordinates are in the spherical system (lat/lon). Among other, this indicates that the x-coordinates will be multiplied by a factor that accounts for the visual deformation using lat/lon coordinates. The default is -1, which means that the type of coordinate system will be determined automatically. (Default -1)

`reggrd` If different from 0 it plots a regular grid over the plot for geographical reference. The value of `reggrd` gives the spacing of the regular grid lines. The units must be according to the units used for the coordinates. With value of -1 the regular grid is determined automatically. (Default -1)

`regdst` This value gives the number of intervals that are used to sub-divide the grid given by `reggrd` with a black and white scale around the plot. If 0 it tries to determine automatically the sub-intervals (2 or 4). A value of -1 does not plot the subgrid scale. (Default 0)

`reggry` If plotting the regular overlay grid this gives the gray value used for the grid. 0 is black, and 1 is white. A value of 1 does not plot the overlay grid, but still writes the labels. (Default 1)

`bndlin` Name of file that gives the boundary line that is not part of the finite element domain. The file must be in GRD format. An older BND format is also accepted, but deprecated. (Default is no file)

`ioverl` Create overlay of velocity vectors on scalar value. With the value of 0 no overlay is created, 1 creates an overlay with the velocity speed. The value of 2 overlays vertical velocities 3 water levels and 4 overlays bathymetry.(Default 0)

`inorm` Normally the horizontal velocities are plotted in scale. The value of `inorm` can change this behavior. A value of 1 normalizes velocity vectors (all vectors are the same length), whereas 2 scales from a given minimum velocity `velmin`. Finally, the value of 3 uses a logarithmic scale. (Default 0)

The next parameters give the choice to selectively avoid to plot areas of the basin and to apply different gray tones for the boundary and net lines when plotting the basin. Please remember that when working with gray tones the value should be between 0 (black) and 1 (white).

`ianopl` Area code for which no plot has to be produced. Normally the whole basin is plotted, but with this parameter some areas can be excluded. (Default -1)

`bgray` Gray value used for the finite element grid when plotting the bathymetry. (Default 0.8)

`bbgray` Gray value used for the boundary of the finite element grid. (Default 0)

`bsgray` Gray value used to plot the finite element grid over a scalar or velocity plot. This is basically useful for debugging reasons. The default is to not plot the grid (Default -1.0)

The next two parameters handle the plotting of the lagrangian particles.

`lgrtrj` If equal 1 plot trajectories instead of particle position. (Default 0)

`lgmean` Plot mean positions/trajectories. With the value of 0 no mean pos/traj are created, 1 plot mean pos/traj together with single values, 2 plot only mean pos/trajs, 3 as 2 but the first trajectory is plot in tichk line. (Default 0)

The next parameters handle the plotting of the basin.

`ibox` If set to 1 plots box information if the area code is used to indicate the box number. This parameter is only useful for the plotting of the box model. (Default 0)

`inumber` If set to 1 plots node and element numbers on top of the grid. This is only useful in debug mode. (Default 0)

C.2.3 Section `$color`

Section `$color` deals with the definition of the colors to be used in the plot. A color bar is plotted too.

`icolor` Flag that determines the type of color table to be used. 0 stands for gray scale, 1 for HSB color table. Other possible values are 2 (from white to blue), 3 (from white to red), 4 (from blue over white to red) and 5 (from blue over black to red). Values 6 and 7 indicate non-linear HSB color tables. (Default 0)

`colfil` A color table can also be read from file. An example of the format can be found in directory `femplot/color` in the file `colormap.dat`. The variable `colfil` indicates the file where the color table is being read from. The default is not to read a color table file.

`coltab` If a color table file has been read then the variable `coltab` indicates the name of the color table that is going to be used. The default is to not use any of the color tables if no name is specified.

`isoval` Array that defines the values for the isolines and colors that are to be plotted. Values given must be in the unit of the variable that will be plotted, i.e., meters for water levels etc.

`color` Array that gives the color indices for the plotting color to be used. Ranges are from 0 to 1. The type of the color depends on the variable `icolor`. For the gray scale table 0 represents black and 1 white. Values in between correspond to tones of gray. For the HSB color table going from 0 to 1 gives the color of the rainbow. There must be one more value in `color` than in `isoval`. The first color in `color` refers to values less than `isoval(1)`, the second color in `color` to values between `isoval(1)` and `isoval(2)`. The last color in `color` refers to values greater than the last value in `isoval`.

`x0col` `y0col` Lower left corner of the area where the color bar is plotted.

`x1col` `y1col` Upper right corner of the area where the color bar is plotted.

`cblank` The color bar is plotted over a white rectangle. Sometimes this blanking is not desirable. If you do not want to have a white box below the legend set `cblank` to 0. (Default 1)

`faccol` Factor for the values that are written to the color bar legend. This enables you, e.g., to give water level results in mm (`faccol = 1000`). (Default 1)

`ndccol` Decimals after the decimal point for the values written to the color bar legend. Use the value `-1` to not write the decimal point. A value of 0 automatically computes the number of decimals needed. (Default 0)

`legcol` Text for the description of the color bar. This text is written above the color bar.

It is not necessary to give all values for isolines and colors above. A faster way is to give only the minimum and maximum values and fix the number of isovalues to be used.

`niso` Total number of isolines to use. (Default is `nisodf`)

`nisodf` Default number of isolines to use. (Default 5)

`colmin` `colmax` Minimum and maximum color index used. Defaults are 0.1 and 0.9 respectively. The value of `colmax` can be smaller than `colmin` which inverts the color index used.

`valmin` `valmax` Minimum and maximum value for isovalues to be used. There is no default.

`rfiso` Defines function to be used to compute intermediate values between `valmin` and `valmax`. If 0 or 1 the values are linearily interpolated. Else they are computed by $y = x^n$ where n is `rfiso` and $x = \frac{v - v_{min}}{v_{max} - vmin}$. Values for `rfiso` greater than 0 capture higher detail in the lower values, whereas values less than 1 do the opposite. (Default 0)

`ipllog` Indicates the usage of a logarithmic color scale. The possible values are 0-3. The value of 0 indicates not to use a logarithmic scale. If 1, the values of the scale are 1,10,100,etc., if 2 the values 1,2,10,20,100,etc. are used, and for 3 the values are 1,2,5,10,20,50,100,etc. (Default 0)

`dval` Difference of values between isolines. If this value is greater then 0 the values for isolines and the total number of isolines are computed automatically using also `valmin` and `valmax`. (Default 0)

Since there is a great choice of combinations between the parameters, we give here the following rules how the values for colors and isolines are determined.
If colors are given in array `color`, they are used, else `colmin` and `colmax` or their respective defaults are used to determine the color bar. If `isoval` is given it is used, else `valmin` and `valmax` are used. If `valmin` and `valmax` are not given they are computed every time for each plot and the minimum and maximum value in the basin are used. In any case, if `isoval` is specified the total number of isovalues is known and `niso` is ignored. However, if `isoval` is not given then first `dval` is used to decide how many isovalues to plot, and if `dval` is 0 then the `niso` and finally `nisodf` is used.
Other parameters that can be changed are the following.

`nisomx` Maximum for `niso` allowed. This is especially useful when the value for `niso` is determined automatically. It avoids you to plot 1000 isolines due to wrong settings of `dval`. However, if you want to use 50 isovalues then just set `niso` and `nisomx` to 50. (Default 20)

`nctick` Number of values to be written in color bar. If `niso` is high the labels on the color bar become unreadable. Therefore you can use `nctick` to write only some of the values to the color bar. For example, if `valmin` is 0 and `valmax` is 5 and you use many isolines, then setting `nctick` to 6 would give you labels at values 0,1,2,3,4,5. If `nctick` is 0 then all labels are written. (Default 0)

`isolin` Normally the isolines are not drawn on the plot, just the colors are used to show the value in the different parts of the plot. A value different from 0 plots also the isolines. In this case `isolin` gives the number of isolines to be plotted. A good choice is to make this equal to `nctick`, so that the isolines correspond to the values written on the colorbar. For compatibility, a value of 1 plots all isolines. (Default 0)

`isoinp` Normally inside elements the values are interpolated. Sometimes it is useful to just plot the value of the node without interpolation inside the element. This can be accomplished by setting `isoinp=0`. Setting instead `isoinp` to a value of 2 plots a constant value in the element. (Default 1)

C.2.4 Section `$arrow`

Parameters in section `$arrow` deal with the reference arrow that is plotted in a legend. The arrow regards the plots where the velocity or the transport is plotted.

`x0arr`
`y0arr` Lower left corner of the area where the reference arrow is plotted.

`x1arr`
`y1arr` Upper right corner of the area where the reference arrow is plotted.

`ablank` The arrow legend is plotted over a white rectangle. Sometimes this blanking is not desirable. If you do not want to have a white box below the legend set `ablank` to 0. (Default 1)

`facvel` Factor for the value that is written to the arrow legend for the velocity. This enables you, e.g., to give velocities in mm/s (`facvel = 1000`). (Default 1)

`ndcvel` Decimals after the decimal point for the values written to the arrow legend. Use the value `-1` to not write the decimal point. (Default 2)

`legvel` Text for the description of the arrow legend. This text is written above the arrow.

`arrvel` Length of arrow in legend (in velocity units). If not given the arrow length will be computed automatically. (Default 0)

`sclvel` Additional factor to be used for the arrow in the legend. When the arrow length will be computed automatically, this parameter gives the possibility to change the length of the reference vector. This is an easy way to scale the velocity arrow with an overall factor. Not used if `arrvel` is given. (Default 1)

C.2.5 Section `$legend`

In section `$legend` annotations in the plots can be given. The section consists of a series of lines that must contain the following information:
The first value is a keyword that specifies what has to be plotted. Possible values are `text`, `line`, `vect`, `rect`, `circ` and also `wid` and `col`. These correspond to different types of information that is inserted into the plot such as text, line, vector, rectangle or circle (filled or just outline). Moreover, the color and line width of the pen can be controlled by with `wid` and `col`.
In case of `text` the starting position (lower left corner) is given, then the point size of the font and the text that is inserted. `line` needs the starting and end point of the line. The same with `vect`, but in this case also the relative tip size must be given as a final parameter. `rect` needs the coordinates of the lower left corner and upper right corner of the rectangle. It also needs the color used for the filling of the rectangle (0-1) or the flag -1 which only draws the outline of the rectangle without filling it. `circ` needs the center point and the radius and a fill color (see rectangle). Finally `wid` needs the relative width of the line and `col` the stroke color used when plotting lines.
A small example of an annotation that explains the above parameters would be:

```
$legend
text  30500 11800     15  'Chioggia'   #text, 15pt
line  30500 11800 35000 15000          #line
vect  30500 11800 35000 15000 0.1      #arrow, tip size 0.1
rect  30500 11800 35000 15000 0.1      #rectangle, fill color 0.1
rect  30500 11800 35000 15000 -1       #rectangle (outline, no fill)
circ  30500 11800 5000 -1              #circle (outline, no fill)
wid   5                                #set line width to 5
col   0.5                              #set color to 0.5
$end
```

There is also an old way to specify the legend that does not use keywords. However, this way is deprecated and unsupported and is therefore not described anymore in this manual.

C.2.6 Section `$legvar`

In section `$legvar` variable fields like the date and wind vectors may be inserted into the plot.
A time and date can be assigned to the simulation results. These values refer to the time 0 of the FEM model. The format for the date is YYYYMMDD and for the time HHMMSS. You can also give a time zone if your time is not referring to GMT but to another time zone such as MET. Please note that you have to give this information only if the simulation does not contain it already. Normally, this information is already assigned during the simulation runs.

`date` The real date corresponding to time 0. (Default 0)

`time` The real time corresponding to time 0. (Default 0)

`tz` The time zone you are in. This is 0 for GMT, 1 for MET and 2 for MEST (MET summer time). (Default 0)

`tzshow` The time zone you want to show your results. If your time zone is GMT (0) and you want to show the results referred to MET (+1) set this to +1. Please note that you have to set this variable only if you want to show results in a different time zone than the one given in `tz`. (Default 0)

The information of date and time may be written to the plot. This is done with the following parameters.

`xdate` `ydate` Starting point for the date text (lower left corner).

`sdate` Point size of the text. (Default 18)

`idate` Output mode. If 0 no date is written to the plot, else the date and time is written. (Default 0)

Wind data can be used to insert a wind vector into the figure. This is useful because in the case of variable wind the direction and speed of the wind that was blowing in the moment of the plot is shown.
Since only one wind vector can be plotted, the wind data must consist of one value for each time. The same ASCII file that is used in the STR file can be used.

`xwind` `ywind` Starting point where the wind arrow is plotted.

`iwtype` Type of wind data. The same as the one in the STR file. If this parameter is 0 then no wind vector is plotted. (Default 0)

`lwwind` Line width of the wind vector. (Default 0.1)

`scwind` Scaling parameter of the wind vector. This depends on the size of your plot. If your wind is 10 m/s and you want the vector to stretch over a distance of 5 km on the plot then you have to choose the value of 500 (10*500=5000) for `scwind`. (Default 1)

`wfile` Name of the file containing the wind data. This may be the same file than the one used in the STR file to run the program.

The wind vector is also given a text legend with the speed of the wind written out. The next parameters decide where and how this information is put.

`xtwind` `ytwind` Starting point for the legend text (lower left corner).

`stwind` Point size of the text. (Default 18)

`wtext` Text used for the legend (Default 'Wind speed')

`wunit` Unit for the wind speed (Default 'm/s')

C.2.7 Section `$name`

In section `$name` the names of input files can be given. All directories default to the current directory, whereas all file names are empty, i.e., no input files are given.

File names Strings in section `$name` enable the specification of files that account for initial conditions or forcing.

`zinit`	Name of file containing initial conditions for water level
`uvinit`	Name of file containing initial conditions for velocity
`wind`	File with wind data. The file may be either formatted or unformatted. For the format of the unformatted file please see the section where the WIN file is discussed. The format of formatted ASCII file is in standard time-series format, with the first column containing the time in seconds and the next two columns containing the wind data. The meaning of the two values depend on the value of the parameter `iwtype` in the `para` section.
`qflux`	File with heat flux data. This file must be in a special format to account for the various parameters that are needed by the heat flux module to run. Please refer to the information on the file `qflux`.
`rain`	File with rain data. This file is a standard time series with the time in seconds and the rain values in mm/day. The values may include also evaporation. Therefore, also negative values (for evaporation) are permitted.
`ice`	File with ice cover. The values range from 0 (no ice cover) to 1 (complete ice cover).
`surfvel`	File with surface velocities from observation. These data can be used for assimilation into the model.
`restrt`	Name of the file if a restart is to be performed. The file has to be produced by a previous run with the parameter `idtrst` different from 0. The data record to be used in the file for the restart must be given by time `itrst`.
`gotmpa`	Name of file containing the parameters for the GOTM turbulence model (iturb = 1).
`tempin`	Name of file containing initial conditions for temperature
`saltin`	Name of file containing initial conditions for salinity
`conzin`	Name of file containing initial conditions for concentration
`tempobs`	Name of file containing observations for temperature
`saltobs`	Name of file containing observations for salinity

`temptau`	Name of file containing the time scale for nudging of temperature
`salttau`	Name of file containing the time scale for nudging of salinity
`bfmini`	Name of file containing initial conditions for bfm
`offlin`	Name of the file if a offline is to be performed. The file has to be produced by a previous run with the parameter `idtoff` greater than 0.

Table C.1: Additional parameter for the sediment transport model to be set in the sedcon file.

Name	Default value	Description
CSULVA	159.4	Coefficient for the solid transmitted stress by Ulva
TMULVA	1.054d-3	Threshold of motion of Ulva [Pa]
TRULVA	0.0013	Threshold of full resuspension of Ulva [Pa]
E0	1.95d-5	Minimum erosion rate
RKERO	5.88	Erosion proportionality coefficient
WSCLAY	5.0	Primary median Ws class (in the range 1:NBCONC)
CDISRUPT	0.001	Constant for turbulent floc disruption during erosion
CLIM1	0.1	Lower limit for flocculation [kg/m3]
CLIM2	2.0	Limit between simple/complex flocculation [kg/m3]
KFLOC	0.001	Constant K for flocculation equation
MFLOC	1.0	Constant M for flocculation equation
RHOCLAY	2600.0	Density of clay mineral
CTAUDEP	1.0	Scaling factor for TAUCD
PRS	0.0	Resuspension probability [0-1]
RHOMUD	50.0	Density of the freshly deposited mud
DPROFA	470.0	Constants for density profile
DPROFB	150.0	A : final deep density
DPROFC	0.015	Define the shape (in conjunction with B and C)
DPROFD	0.0	Aux parameter for density profile
DPROFE	0.0	Aux parameter for density profile
CONSOA	1d-5	time constant of consolidation
TEROA	6d-10	Constant for erosion threshold from density
TEROB	3.0	Aux parameter for erosion threshold from density
TEROC	3.47	Aux parameter for erosion threshold from density
TEROD	-1.915	Aux parameter for erosion threshold from density
KCOES	0.15	Fraction of mud for sediment to be cohesive
CDRAGRED	-0.0893	Constant for the drag reduction formula
Z0COH	2.0D-4	Bed roughness length for cohesive sediments
FCWCOH	2.2D-3	Friction factor for cohesive sediments
LIMCOH	0.063	Limit of cohesive sediment grainsize [mm]
SMOOTH	1.0	Smoothing factor for morphodynamic
ANGREP	32.0	Angle of repose
IOPT	5	Sediment bedload transport formula option number
MORPHO	1.0	Morphological acceleration factor
RHOSED	2650.0	Sediment grain density
POROS	0.4	Bed porosity [0-1]
SURFPOR	0.6	Bed porosity of freshly deposited sand [0-1]
DOCOMPACT	0.0	If not zero, call COMPACT routine

Bibliography

[1] Jan O. Backhaus. A semi-implicit scheme for the shallow water equations for aplication to shelf sea modelling. *Continental Shelf Research*, 2(4):243–254, 1983.

[2] R. A. Bagnold. Mechanics of Marine Sedimentation. In *The Sea*, volume 3, pages 265–305. Hill, M.N. Ed., 1963.

[3] C. B. Brown. *Engineering Hydraulics*. Rouse H. Ed., 1950.

[4] Kurt C. Duwe and Regina R. Hewer. Ein semi-implizites gezeitenmodell für wattgebiete. *Deutsche Hydrographische Zeitschrift*, 35(6):223–238, 1982.

[5] F. Engelund and E. Hansen. *A monograph on sediment transport in alluvial stream*. Teknisk Vorlag, Copenhagen, Denmark, 1967.

[6] C. Ferrarin, G. Umgiesser, A. Cucco, T-W Hsu, A. Roland, and C. L. Amos. Development and validation of a finite element morphological model for shallow water basins. *Coast. Eng.*, 55(9):716–731, 2008.

[7] H. B. Fisher, E. J. List, R. C. j. Koh, J. Imberger, and N. H. Brooks. *Mixing in Inland and Coastal Waters*, volume 66 of *International Geophysics Series*. Academic Press, San Diego, USA, 1979. 302 pp.

[8] U. Gräwe and J.-O. Wolff. Suspended particulate matter dynamics in a particle framework. *Environ. Fluid Mech.*, 10(1):21–39, 2010.

[9] G. Grotkop. Finite element analysis of long-period water waves. *Computer Methods in Applied Mechanics and Engineering*, 2(2):147–157, 1973.

[10] B. Herrling. Computation of shallow water waves with hybrid finite elements. *Advances in Water Resources*, 1:313–320, 1978.

[11] Bruno Herrling. Ein finite-element-modell zur berechnung von Tideströmungen in ästuarien mit Wattflächen. *Die Küste*, 31:102–113, 1977.

[12] K.-P. Holz and G. Nitsche. Tidal wave analysis for estuaries with intertidal flats. *Advances in Water Resources*, 5:142–148, 1982.

[13] Michael Kwizak and André J. Robert. A semi-implicit scheme for grid point atmospheric models of the primitive equations. *Monthly Weather Review*, 99(1):32–36, 1971.

[14] U. Neumeier, C. Ferrarin, C. L. Amos, G. Umgiesser, and M. Z. Li. Sedtrans05: An improved sediment-transport model for continental shelves and coastal waters. *Comput. Geosci.*, 34(10):1223–1242, 2008.

[15] A. Roland, A. Cucco, C. Ferrarin, Tai-Wen Hsu, Jian-Ming Liau, Shan-Hwei Ou, G. Umgiesser, and U. Zanke. On the development and verification of a 2d coupled wave-current model on unstructured meshes. *J. Mar. Syst.*, 78, Supplement:S244–S254, 2009.

[16] A. Schoenstadt. A transfer function analysis of numerical schemes used to simulate geostrophic adjustment. *Monthly Weather Review*, 108:1248, 1980.

[17] J. Smagorinsky. Some historical remarks on the use of non-linear viscosities - 1.1 Introductory remarks. In B. Galperin and S. A. Orszag, editor, *Large Eddy Simulation of Complex Engineering and Geophysical Flows, Proceedings of an International Workshop in Large Eddy Simulation*, pages 1–32. Cambridge University Press, Cambridge, UK, 1993.

[18] C. Taylor and J. Davis. Tidal and long wave propagation—a finite element approach. *Computers & Fluids*, 3:125–148, 1975.

[19] Georg Umgiesser. A model for the Venice Lagoon. Master's thesis, University of Hamburg, 1986.

[20] Georg Umgiesser and Andrea Bergamasco. A staggered grid finite element model of the Venice Lagoon. In J. Periaux K. Morgan, E. Ofiate and O.C. Zienkiewicz, editors, *Finite Elements in Fluids*. Pineridge Press, 1993.

[21] U.S. Army Engineer Waterways Experiment Station. *Shore Protection Manual*. U.S. Government Printing Office, Washington DC, U.S., 1984.

[22] L. C. Van Rijn. *Principles of sediment transport in rivers, estuaries and coastal sea*. Aqua Publications, Amsterdam, The Netherlands, 1993.

[23] R. T. Williams. On the formulation of finite-element prediction models. *Monthly Weather Review*, 109:463, 1981.

[24] R. T. Williams and O. C. Zienkiewicz. Improved finite element forms for the shallow-water wave equations. *International Journal for Numerical Methods in Fluids*, 1:81, 1981.

[25] M. S. Yalin. An expression for bedload transportation. In *Journal of Hydraulics and Division*, volume ASCE 89 (HY3), pages 221–250, 1963.