# SHYFEM
# Finite Element Model for Coastal Seas

## User Manual

The SHYFEM Group
Georg Umgiesser
ISMAR-CNR, Castello 1364/A
30122 Venezia, Italy

georg.umgiesser@ismar.cnr.it

Version 5.29

July 3, 2009

# Contents

# Disclaimer

Comments and additions should be sent to the author:

```
Georg Umgiesser
ISMAR-CNR
Castello 1364/A
30122 Venezia
Italy

Tel.   : ++39-41-2404773
Fax    : ++39-41-5204126
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, coastal 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 application to a complicated geometry and bathymetry.
This version of the program `SHYFEM` resolves the depth integrated shallow water equations. It is therefore recommended for the application of very shallow basins or well mixed estuaries. Storm surge phenomena can be investigated also. This two-dimensional version of the program is not suited for the application to baroclinic driven flows or large scale flows where the the Coriolis acceleration is important.
Finite elements are superior to finite differences when dealing with complex bathymetric situations and geometries. Finite differences are limited to a regular outlay of their grids. This will be a problem if only parts of a basin need high resolution. The finite element method has an advantage in this case allowing more flexibility with its subdivision of the system in triangles varying in form and size.
This model is especially adapted to run in very shallow basins. 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 [3, 9, 5, 4, 6].
The model presented here [10, 11] 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 [7, 2, 1].

# Chapter 2

# Installing SHYFEM

## 2.1 Installation

The source code of the model is provided in a file named `fem_VERS_N_nn.tar.gz`, with `N` and `nn` the number of version and subversion of the code. The file can be downloaded by the SHYFEM web-site[1], or by contacting directly its authors.
The source code is composed mainly by Fortran 77 files, but files written in C, Fortran 90, 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.
Please see the next section for software that is helpful for running and using the model. However, the software absolutely necessary is the following:

- A Fortran 77 and 90 compiler. Supported compilers are the Gnu `g77` and `g90` compilers, the new Gnu compiler `gfortran`, the Intel Fortran compiler `ifort` or the Portland group Fortran compiler.
- The package `make` is required for compilation.
- The `perl` interpreter, the `bash` shell and the `gcc` c compiler are necessary for compiling.

Please note that you might already have everything available in your Linux distribution, except might be from the fortran compiler. Please see the section on "Needed software" for more information on the recommended software you should have installed on your computer.
Once you have downloaded the model open a shell as a normal user, go to your home directory and run the following command:
```
tar -xzvf path_to_file/fem_VERS_N_nn.tar.gz
```
where `path_to_file` is the directory where you have downloaded the zipped model file.
At this point a new folder named `fem_VERS_N_nn` has been created. Move into this directory (`cd fem_VERS_N_nn`) and open the file `Rules.make` with a text editor. In this file all the compilation options can be set. These are the sections inside it:

- `Parameters`. In this section you have to set the maximum number of nodes (`NKNDIM`) and elements (`NELDIM`) used by your grids. You might also have to set also the maximum number of elements attached to a node (`NGRDIM`) and the maximum bandwidth (`MBWDIM`) of the z-level matrix. You can find these numbers when you create the

[1] http://www.ve.ismar.cnr.it/shyfem/

basin file with `vpgrd`. Finally you have to specify the maximum number of vertical levels (`NLVDIM`). It is advisable to set this value close to the desired number of vertical levels, since it affects the model speed performance. So, if you want to run the model in 2D mode, please set `NLVDIM` to 1.

- `Compiler`. Set the compiler you want to use.
- `Parallel compilation`. Some parts of the code are parallelized with OpenMP statements. Here you can set if you want to use it or not. Intel and Gnu gfortran compilers accept OpenMP statements.
- `Solver for matrix solution`. There are three different solvers implemented. The `GAUSS` solver is the most robust and best tested solver, but it is quite slow. The `PARDISO` solver needs an external library available at the Intel web-site[2], that is freely downloadable for non-commercial use. The Pardiso solver is parallelized, but it seems to be a little slower than `SPARSKIT` solver. The `SPARSKIT` solver is suggested solver, since it seems to be the fastest one. However, if you are ever in doubt about your results you might to revert back to the `GAUSS` solver and check the results.
- `NetCDF library`. If you want the 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 oreder to use it please the flag to true. This is the recommended choice.
- `ERSEM library`. The ERSEM ecological model. It is still experimental. Use it at your own risk.
- `Compiler options`. Here several sections are present, one for each supported compiler. Normally it should not be necessary to change anything beyond this point.

Once you have set all these options you can run the command
`make fem`
to compile everything.
Another useful command is `make clean` to delete files of previous compilations.
It is advisable to make in your home directory a symbolic link named `fem` to the fem main directory, otherwise some of the commands and shell scripts might not work properly. This can be done with the command
`ln -s fem_VERS_N_nn fem`
from your home directory. If there is already such a link existing you first have to delete it (`rm fem`).
Another useful option is to add the fembin path to your default paths to have the main utility commands always available. To do this open your `.bashrc` file in the home directory and add the following lines at the end of the file:

```
FEMDIR=$HOME/fem
PATH=$PATH:$FEMDIR/fembin
export FEMDIR PATH
```

If you have not set a symbolic link from `fem` to the actual model directory, please substitute `fem` with the appropriate directory `fem_VERS_N_nn`.
Summarizing, the following steps have to be taken in order to properly install the model on your computer:

- download the model and unpack it into your home directory

[2]http://software.intel.com/en-us/articles/code-downloads/

- `cd` to the model directory and adjust `Rules.make`
- compile the model with `make fem`

In addition, you also might

- set a symbolic link from `fem` to the model directory `ln -s fem_VERS_N_nn fem`
- adjust the file `.bashrc` adding the `FEMDIR` shell variable and a new entry for the `PATH`

# Chapter 3

# Preprocessing: The grid

## 3.1 Creating the basin file

The pre-processing routine `vp` is used to generate an optimized version of the file that describes the basin where the main program is to be run. In the following a short introduction in using this program is given.

### 3.1.1 The pre-processing routine `vp`

The main routine `hp` reads the basin file generated by the pre-processing routine `vp` and uses it as the description of the domain where the hydrodynamic equations have to be solved.
The program `vp` is started by typing `vp` on the command line. From this point on the program is interactive, asking you about the basin file name and other options. Please follow the online instructions.
The routine `vp` reads a file of type GRD. This type of file can be generated and manipulated by the program `grid` which is not described here. In short, the file GRD consists of nodes and elements that describe the geometrical layout of the basin. Moreover, the elements have a type and a depth.
The depth is needed by the main program `hp` to run the model. The type of the element is used by `hp` to determine the friction parameter on the bottom, since this parameter may be assigned differently, depending on the various situations of the bottom roughness.
This file GRD is read by `vp` and transformed into an unformatted file BAS. It is this file that is then read by the main routine `hp`. Therefore, if the name of the basin is `lagoon`, then the file GRD is called `lagoon.grd` and the output of the pre-processing routine `vp` is called `lagoon.bas`.
The program `vp` normally uses the depths assigned to the elements in the file GRD to determine the depth of the finite elements to use in the program `hp`. In the case that these depth values are not complete, and that all nodes have depths assigned in the GRD file, the nodal values of the depths are used and interpolated onto the elements. However, if also these nodal depth values are incomplete or are missing altogether, the program terminates with an error.

### 3.1.2 Optimization of the bandwidth

The main task of routine `vp` 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 `vp` 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 `vp`. 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.

### 3.1.3 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 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 How to run: the Parameter Input File (STR)

The model reads one input file that determines the behavior of the simulation. All possible parameters can and must be set in this file. If other data files are to be read, here is the place where to specify them.
The model reads this parameter file from standard input. Thus, if the model binary is called `hp` and the parameter file `param.str`, then the following line starts the simulation

```
ht < param.str
```

and runs the model.

### 4.1.1 The General Structure of the Parameter Input File

The input parameter file is the file that guides program performance. It contains all necessary information 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

```
$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)

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 usages

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

```
$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)

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.
- 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 and its time step idt.

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.

### 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. For lagoons it would 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.

## 4.3 Advanced usages

### 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 introduction of the advective terms (`ilin=0`) or horizontal viscosity (`ahpar` greater 0) can introduce 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, $\Delta t$ the time step and $\Delta x$ 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

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 on the basis of z layers. In this representation 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.
In order to use 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. Please, make sure that in the file Rules.make, the number of allowed levels nlvdim is greater or equal than the ones actually used in the STR file. 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
2 4 6 8 10 13 16 20
$end
```

Figure 4.5: Example of section `$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.

layers. The bathymetric value normally does not coincide with one of the layer depths, and therefore the last layer must be treated seperately.
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 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 seperate 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.
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 coeficients through a turbulence closure scheme. This usage will be described in the scetion on turbulence.

### 4.3.3 Baroclinic terms

The baroclinic pressure gradient term permits to compute the variation of velocity due to the horizontal gradients of temperature and/or salinity. These gradients act on the horizontal variation of density.
If the variations of temperature and salinity and the baroclinic pressure gradient term has to be computed, 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. 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 will not be computed but 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 density field to the currents.
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. 4.6.
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.

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

Figure 4.6: Example of baroclinic simulation. The initial values for temperature and salinity are set to 18 C and 35.

### 4.3.4 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 turbulence closure scheme applied is the $k-\varepsilon$ model. If iturb=2 the GOTM turbulence closure module is used. In this case 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=2 is recommended. An example of the settings for the turbulence closure scheme is given in Fig. 4.7.

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

```
$para
        iturb = 2
$end
$name
        gotmpa = 'gotmturb.nml'
$end
```

Figure 4.7: Example of turbulence settings. The GOTM module for the turbulence closure is used. The parameters are contained in file gotmturb.nml.

# 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 $\zeta$ 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 $\Delta t$ 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 $\Delta x$ 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

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

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

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

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

$$U^{n+1} = \frac{1}{1+\Delta tR}\left(U^n - \Delta t gH\tfrac{1}{2}\frac{\partial(\zeta^{n+1}+\zeta^n)}{\partial x} - \Delta tX\right) \qquad \text{(A.9)}$$

$$V^{n+1} = \frac{1}{1+\Delta tR}\left(V^n - \Delta t gH\tfrac{1}{2}\frac{\partial(\zeta^{n+1}+\zeta^n)}{\partial y} - \Delta tY\right) \qquad \text{(A.10)}$$

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

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

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

The terms on the left hand side contain the unknown $\zeta^{n+1}$, 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 $\zeta^{n+1}$ is obtained and can be solved by standard methods. Once the solution for $\zeta^{n+1}$ 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 $\zeta$ 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 $\phi_m$ as

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

where $u_m$ is the coefficient of the function $\phi_m$ 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 $\Psi_n$ and integrating over the whole domain leading to

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

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

$$a_{nm} u_m = 0 \qquad n = 1, K \quad m = 1, K \tag{A.14}$$

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

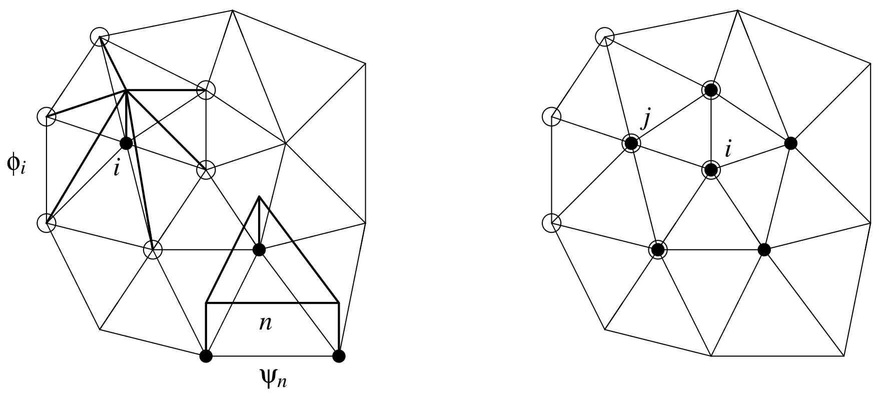

Figure A.1: a) form functions in domain b) domain of influence of node $i$

**Staggered Finite Elements**

For decades finite elements have been used in fluid mechanics in a standardized manner. The form functions $\phi_m$ 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 $\phi_m$ 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 $\phi_i$ 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 $\psi_n$ 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 $\psi_n$ 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 $\zeta_m$ 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 [8]. He showed that the un-staggered finite element formulation of the shallow water equations has very poor geostrophic adjustment properties. Williams [12, 13] 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 $\zeta$ are expanded in piecewise linear functions $\phi_m,\ m = 1, K$ and the transports are expanded in piecewise constant functions $\psi_n,\ n = 1, J$ where $K$ and $J$ are the total number of nodes and elements respectively. As weighting functions we use $\psi_n$ for the momentum equations and $\phi_m$ for the continuity equation. In this way there will be $K$ equations for the unknowns $\zeta$ (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 $\Phi_M$ 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

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

Here we have introduced the symbol $\alpha$ 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 $\zeta$ 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 $\zeta_i$, 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 $\Omega_i$ around node $i$ but also all the triangles that have nodes in common with the triangles $\Omega_i$ 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 $\zeta$ 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 $\Omega_i$. 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

# Parameter list

## B.1 Parameter list for the SHYFEM model

### B.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 B.1.

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

Figure B.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.

### B.1.2 Section `$para`

This section 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** This parameters are compulsory parameters that define the period of the simulation. They must be present in all cases.

| | |
|---|---|
| `itanf` | Start of simulation. (Default 0) |
| `itend` | End 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 0.

| | |
|---|---|
| `idtout` `itmout` | Time step and start time for writing to file OUT, the file containing the general hydrodynamic results. |
| `idtext` `itmext` | Time step and start time for writing to file EXT, the file 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` `itmrst` | Time step and start time for writing the restart |

file (extension RST). No restart file is written with `idtrst` equal to 0. (Default 0) `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. `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 be not doing what you want. (Default 0)

| | |
|---|---|
| `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. |
| `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) |

**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 YYMMDD and for the time HHMMSS. Please note that the date should not be given as YYYYMMDD because due to precision problems this will not work. 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 choses 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.

| | |
|---|---|
| `itsplt` | Type of variable time step computation. If this value is 0, the time step will 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 will never be exceeded. (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 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. This parameter is useful only for `itsplt` = 2 and its default value of 0 does not make any syncronization. |

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) |

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

**Coriolis parameters** The next parameters define the parameters to be used in 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 $f$. If not given the latitude in the basin file is used. If given the value of `dlat` in the input parameter file effectively substitues the value given in the basin file. |
| `isphe` | If 0 a cartesian coordinate system is used (default), if 1 the coordinates are in the spherical system (lat/lon). |

**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 $\lambda$ is specified that is used in the formulas below.

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

**0** No friction used

**1** $R = \lambda$ is constant

**2** $\lambda$ 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** $\lambda$ 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

**5** $R = \lambda \frac{|u|}{H}$

| | |
|---|---|
| `czdef` | The default value for the friction parameter $\lambda$. Depending on the value of `ireib` the coefficient $\lambda$ is describing linear friction, constant drag coefficient or a Chezy or Strickler form of friction (default 0). |
| `iczv` | Normally $R$ is evaluated at every time step (`iczv` = 1). If for some reason this behavior is not desirable, `iczv` = 0 evaluates the value of $R$ only before the first time step, keeping it constant for the rest of the simulation. (default 1) |

The value of $\lambda$ 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

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

where $\rho_a, \rho_0$ 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):<br>**0** No wind data is processed<br>**1** The components of the wind is given in [m/s]<br>**2** The stress $(\tau^x, \tau^y)$ is directly specified<br>**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.<br>**4** As in 3 but the speed is given in knots |
| `dragco` | Drag coefficient used in the above formula. The default value is 0 so it must be specified. Please note also that in case of `iwtype` = 2 this value is of no interest, since the stress is specified directly. |

**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`. |
| `diftur` | Vertical turbulent diffusion parameter for the tracer. (Default 0) |
| `difmol` | Vertical molecular diffusion parameter for the tracer. (Default 1.0e-06) |
| `dhpar` | Horizontal diffusion parameter (general). (Default 0) |
| `itvd` | Type of advection scheme used for the transport and diffusion equation. Normally an upwind scheme is used (0), but setting the parameter `itvd` to 1 choses a TVD scheme. This feature is still experimental, so use with care. (Default 0) |

**Temperature and salinity** The next parameters deal with the transport and diffusion of temperature and salinity. Please note that in order to compute T/S the parameter `ibarcl` must be different from 0. In this case T/S advection is computed, but 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) |
| `temref` | Reference (ambient) temperature of the water in centigrade. (Default 0) |
| `salref` | Reference (ambient) salinity of the water in psu (practical salinity units, per mille). (Default 0) |
| `thpar` | Horizontal diffusion parameter for temperature. (Default 0) |
| `shpar` | Horizontal diffusion parameter for salinity. (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 (ambient) concentration of the tracer in any unit. (Default 0) |
| `contau` | If different from 0 simulates decay of concentration. In this case `contau` is the decay rate (e-folding time) in days. (Default 0) |
| `chpar` | Horizontal diffusion parameter for the tracer. (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 CON (concentration), TEM (temperature) and SAL (salinity), or generally to file NOS. |

### B.1.3 Section `$name`

In this sections names of directories or 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.

**Directory specification** This parameters define directories for various input and output files.

| | |
|---|---|
| `basdir` | Directory where basin file BAS resides. (Default .) |
| `datdir` | Directory where output files are written. (Default .) |
| `tmpdir` | Directory for temporary files. (Default .) |
| `defdir` | Default directory for other files. (Default .) |

**File names** The following strings enable the specification of files that account for initial conditions or forcing.

| | |
|---|---|
| `bound` | File with initial water level distribution. This file must be constructed by the utility routine `zinit`. |

| | |
|---|---|
| `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. |
| `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. |
| `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`. |
| `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). |

### B.1.4 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. At least two nodes must be given. |

| | |
|---|---|
| `ibtyp` | Type of open boundary.<br>**0** No boundary values specified<br>**1** Level boundary. At this open boundary the water level is imposed and the prescribed values are interpreted as water levels in meters.<br>**2** Flux boundary. Here the discharge in $m^3\ s^{-1}$has to be prescribed.<br>**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.<br>**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. |

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`<br>`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`<br>`tempn`<br>`saltn` | File name that contains 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. |

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 |
| `tox3dn` | File name that contains values for the toxicological module. |
| `intpol` | Order of interpolation for the boundary values read through 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.

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

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.

| | |
|---|---|
| `umom`<br>`vmom` | Constant values for momentum input. (Default 0) |

### B.1.5 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`.

### B.1.6 Section `$extra`

In this section the node numbers of so called “extra” points are given. These are points where water level and velocities 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 hydrodynamical output. The output is written to file EXT.
The node numbers are specified in a free format on one ore more lines. An example can be seen in figure 4.1. No keywords are expected in this section.

### B.1.7 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. Between two transects one or more 0’s must be inserted. An example is given in figure B.2.

```
$flux
1001 1002 1004 0
35 37 46 0 0 56 57 58 0
407
301
435 0 89 87
$end
```

Figure B.2: Example of section `$flux`

The example shows the definition of 5 transects. As can be seen, the nodes of the transects can be given on one line alone (first transect), two transects on one line (transect 2 and 3), spread over more lines (transect 4) and a last transect.

## B.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.

### B.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 B.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.

### B.2.2 Section `$para`

These parameters 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`<br>`y0` | Lower left corner of the plotting area. (Default is whole area) |
| `x1`<br>`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.

| | |
|---|---|
| `x0leg`<br>`y0leg` | Lower left corner of the area where the legend is plotted. |
| `x1leg`<br>`y1leg` | Upper right corner of the area. where the legend is plotted. |
| `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 velocitiy 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) |

| | |
|---|---|
| `isphe` | If 1 spherical coordinates are used (lat/lon). This indicates that the $x$-coordinates will be multiplied by a factor that accounts for the visual deformation using lat/lon coordinates. (Default 0) |
| `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. (Default 0) |
| `reggry` | If plotting the regular overlay grid this gives the grey value used for the grid. 0 is black, and 1 is white. (Default 0.5) |
| `bndlin` | Name of file that gives the boundary line that is not part of the finite element domain. The file must be in BND format. You can use the program grd2bnd.pl to create the file from a GRD file. (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) |

### B.2.3 Section `$color`

The next parameters deal 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). Values 5 and 6 indicate non-linear HSB color tables. (Default 0) |
| `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. |
| `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. (Default -1) |
| `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. |
| `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 lables 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) |
| `bgray` | Gray value used for the finite element grid when plotting the bathymetry. (Default 0.8) |

### B.2.4 Section `$arrow`

The next parameters 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. |
| `facvel` | Factor for the value that are 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) |

### B.2.5 Section `$legend`

In this section 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` and also `wid` and `col`. These correspond to different types of information that is inserted into the plot such as text, line, a vector or a rectangle (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. 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, tipsize 0.1
rect  30500 11800 35000 15000 0.1      #rectangle, fill color 0.1
rect   30500 11800 35000 15000 -1       #rectangle (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.

### B.2.6 Section `$legvar`

In this section 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 strech 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') |

### B.2.7 Section `$name`

**Directory specification** This parameters define directories for various input and output files.

|  |  |
|---|---|
| `basdir` | Directory where basin file BAS resides. (Default .) |
| `datdir` | Directory where output files are written. (Default .) |
| `tmpdir` | Directory for temporary files. (Default .) |
| `defdir` | Default directory for other files. (Default .) |

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

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

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

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

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

[7] 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.

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

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

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

[11] 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.

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

[13] 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.