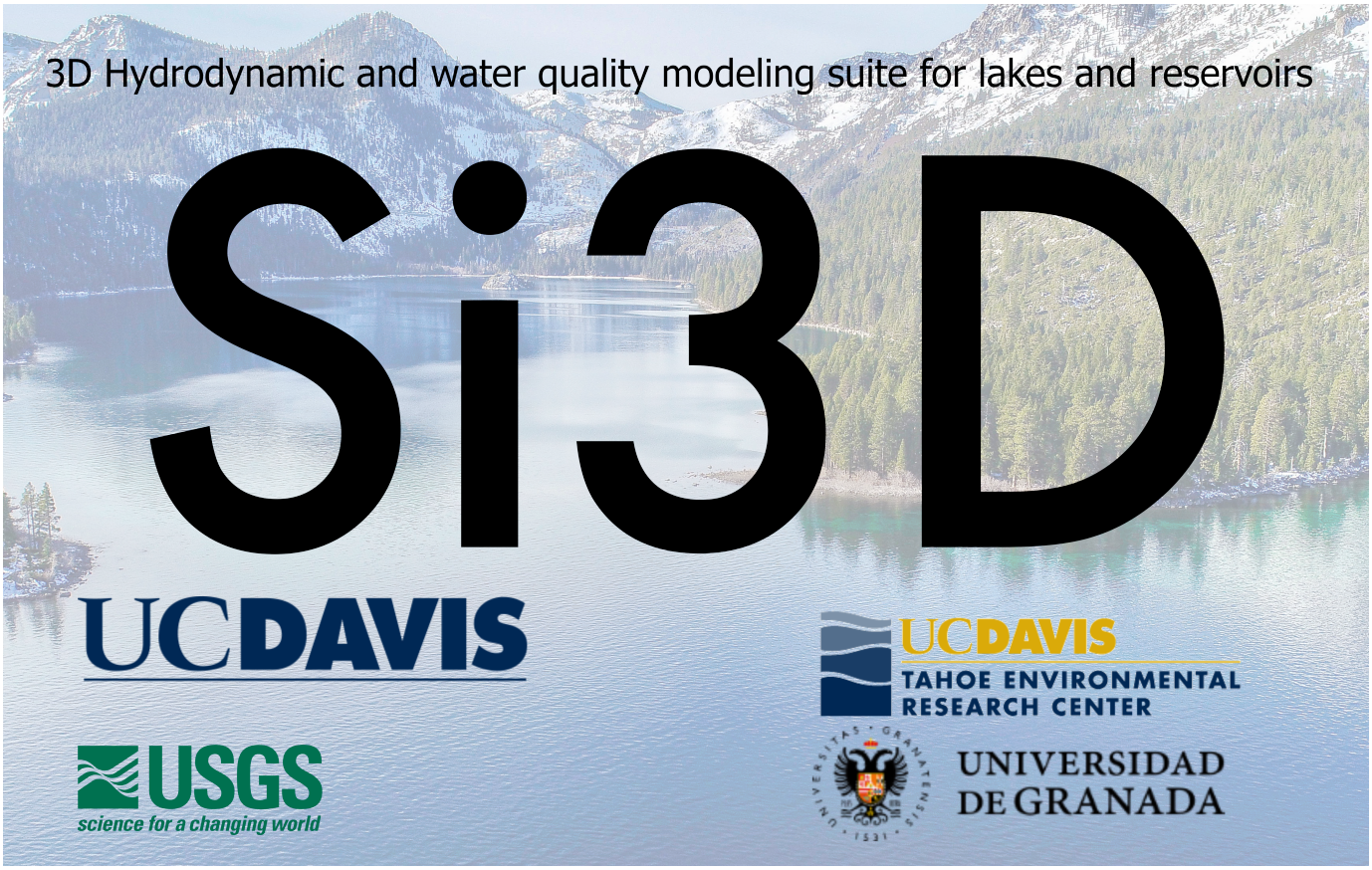

User Manual

University of California Davis

Tahoe Environmental Research Center (TERC)

Authors:
Alicia Cortés
Sergio A. Valbuena Mateus
Micah Swann
Federico Zabaleta
Kenneth Larrieu

2023 - 06 - 01

Contents

List of Figures

List of Tables

Introduction to Si3D-L

1.1. Scientific basis, main features

Si3D-L is a three-dimensional hydrodynamic numerical model which simulates physical transport and mixing phenomena in lake and estuarine environments. The model can simulate flows and sediment transport and can be coupled with the Aquatic Ecosystem Model (AEM) to support water quality and ecological modeling. The model is suitable for a wide range of natural and engineered systems ranging in scale from small, shallow to large, deep water bodies. Si3D-L source is open-source and can be customized for specific project needs.

1.2. List of publications Si3D-L

The numerical model has been applied to multiple lakes and reservoirs around the world. A list of publications available online is shown next:

1. Rueda, Francisco J. (2001). A Three-Dimensional Hydrodynamic and Transport Model for Lake Environments. University of California Davis.
2. Rueda, F. J., & Schladow, S. G. (2002). Quantitative Comparison of Models for Barotropic Response of Homogeneous Basins. Journal of Hydraulic Engineering, 128(2), 201–213.
3. Rueda, F. J., & Schladow, S. G. (2002). Surface seiches in lakes of complex geometry. Limnology and Oceanography, 47(3), 906–910.
4. Rueda, F. J., Schladow, S. G., Monismith, S. G., & Stacey, M. T. (2003). Dynamics of large polymictic lake. I: Field observations. Journal of Hydraulic Engineering, 129(2), 82–91.
5. Rueda, F. J., Schladow, S. G., & Pálmarsson, S. Ó. (2003). Basin-scale internal wave dynamics during a winter cooling period in a large lake. Journal of Geophysical Research C: Oceans, 108(3), 42–1.
6. Rueda, F. J., Schladow, S. G., Monismith, S. G., & Stacey, M. T. (2005). On the effects of topography on wind and the generation of currents in a large multi-basin lake. Hydrobiologia, 532(1), 139–151.
7. Rueda, F. J., & Cowen, E. A. (2005). Residence time of a freshwater embayment connected to a large lake. Limnology and Oceanography, 50(5), 1638–1653.
8. Smith, Peter. (2006). A Semi-Implicit, Three-Dimensional Model for Estuarine Circulation. USGS Report 176.

9. Schladow, S. G., Rueda, F. J., Fleenor, E., & Chung, E. G. (2006). Predicting the effects of configuration changes on Salton Sea stratification using a three-dimensional hydrodynamic model. TERC Rep. 06-007, Report to CH2M HILL and California Department of Water Resources.

10. Rueda, F. J., Fleenor, W. E., & de Vicente, I. (2007). Pathways of river nutrients towards the euphotic zone in a deep-reservoir of small size: Uncertainty analysis. Ecological Modelling, 202(3–4), 345–361.

11. Rueda, Francisco J., S. Geoffrey Schladow, & Jordan F. Clark. (2008). "Mechanisms of Contaminant Transport in a Multi-Basin Lake." Ecological Applications: A Publication of the Ecological Society of America 18(8 Suppl): A72-88.

12. Rueda, F. J., & MacIntyre, S. (2009). Flow paths and spatial heterogeneity of stream inflows in a small multibasin lake. Limnology and Oceanography, 54(6), 2041–2057.

13. Rueda, F. J., Vidal, J., & Schladow, G. (2009). Modeling the effect of size reduction on the stratification of a large wind-driven lake using an uncertainty-based approach. Water Resources Research, 45(3), 1–15.

14. Llebot, Clara. (2010). "Interactions between Physical Forcing, Water Circulation and Phytoplankton Dynamics in a Microtidal Estuary." 212.

15. Doyle, L. (2010). "A Three-Dimensional Water Quality Model for Estuary Environments." 213.

16. Rueda, F. J., & MacIntyre, S. (2010). Modelling the fate and transport of negatively buoyant storm-river water in small multi-basin lakes. Environmental Modelling and Software, 25(1), 146–157.

17. Ramón, Cintia L., Joan. Armengol, Josep. Dolz, Jordi. Prats, and Francisco J. Rueda. (2014). Mixing Dynamics at the Confluence of Two Large Rivers Undergoing Weak Density Variations. Journal of Geophysical Research: Oceans 119(4):2386–2402.

18. Hoyer, Andrea B., Marion E. Wittmann, Sudeep Chandra, S. Geoffrey Schladow, & Francisco J. Rueda. (2014). "A 3D Individual-Based Aquatic Transport Model for the Assessment of the Potential Dispersal of Planktonic Larvae of an Invasive Bivalve." Journal of Environmental Management 145:330–40.

19. Acosta, M., Anguita, M., Fernández-Baldomero, F. J., Ramón, C. L., Schladow, S. G., & Rueda, F. J. (2015). Evaluation of a nested-grid implementation for 3D finite-difference semi-implicit hydrodynamic models. Environmental Modelling & Software, 64, 241–262.

20. Ramón, C. L., A. Cortés, & Francisco J. Rueda. (2015). "Inflow–Outflow Boundary Conditions along Arbitrary Directions in Cartesian Lake Models." Computers & Geosciences 74:87–96.

21. Hoyer, A. B., Schladow, S. G., & Rueda, F. J. (2015). A hydrodynamics-based approach to evaluating the risk of waterborne pathogens entering drinking water intakes in a large, stratified lake. Water Research, 83, 227–236.

22. Hoyer, A. B., Schladow, S. G., & Rueda, F. J. (2015). Local dispersion of nonmotile invasive bivalve species by wind-driven lake currents. Limnology and Oceanography, 60(2), 446–462.

23. Ramón, Cintia L., Jordi Prats, & Francisco J. Rueda. (2016). The Influence of Flow Inertia, Buoyancy, Wind, and Flow Unsteadiness on Mixing at the Asymmetrical Confluence of Two Large Rivers. Journal of Hydrology 539:11–26.

24. Chen, Shengyang, Chengwang Lei, Cayelan C. Carey, Paul A. Gantzer, & John C. Little. (2017). "A Coupled Three-Dimensional Hydrodynamic Model for Predicting Hypolimnetic Oxygenation and Epilimnetic Mixing in a Shallow Eutrophic Reservoir." Water Resources Research 53(1):470–84.

25. Chen, Shengyang, John C. Little, Cayelan C. Carey, Ryan P. McClure, Mary E. Lofton, & Chengwang Lei. (2018). "Three-Dimensional Effects of Artificial Mixing in a Shallow Drinking-Water Reservoir." Water Resources Research 54(1):425–41.

26. Valbuena, S. A., Bombardelli, F. A., Cortés, A., Largier, J. L., Roberts, D. C., Forrest, A. L., & Schladow, S. G. (2022). 3D Flow Structures During Upwelling Events in Lakes of Moderate Size. Water Resources Research, 58(3), 1–35.

1.3. MANUAL SUMMARY

1. Chapter 2: Quick Start describes how to access the source code, software requirements, and how to run a simple sample case already available in our GitHub repository

2. Chapter 3: Input Files provides a detailed description of all the input files required and available to run Si3D. The files described in this chapter are the main input file, the bathymetry file, the initial condition file, the boundary conditions file (open boundary, water surface elevation, point source-sink, and nested boundary), and the surface boundary condition file.

3. Chapter 6: Theory describes the governing equations behind Si3D-L.

4. Chapter 7: Numerical Implementation presents the numerical methods employed in Si3D-L to solve the governing equations.

Quick Start

Si3D-L is a semi-implicit 3D hydrodynamic model written in Fortran 90 and parallelized for computation across CPU threads using the OpenMP directive framework. For more technical details regarding governing equations and numerical schemes, see Chapters 6 and 7.

2.1. Installation

Compilation of the source code requires an environment configured with the Intel Fortran Compiler (`ifort`) and OpenMP (bundled with `ifort` in the Intel oneAPI HPC Toolkit).

To download the source code, clone the repository:

```
git clone https://github.com/SI3DL/psi3d.git
cd psi3d
```

Then, to compile the source code into a binary file (`psi3d`):

```
\bash ./Compiler.sh
```

This should create a `psi3d` binary in the working directory. You may need to change the executable permissions of the binary:

```
chmod +rwx psi3d
```

2.2. Quickstart

The pSi3D repository comes with template input files for setting up a model (`./SampleFiles/`) as well as a sample model run (`./sampleRuns/windDrivenFlow`). Details for this sample model are included as a `readme.txt` file in its subdirectory.

To run the sample model, copy the `psi3d` binary to the sample run directory and execute it:

```
cp ./psi3d ./sampleRuns/windDrivenFlow/
cd ./sampleRuns/windDrivenFlow/
./psi3d
```

The terminal will stream information as the program runs and model logs/outputs will be stored in the working directory.

For a more detailed tutorial on running your own Si3D-L models, see Chapter 4.

2.3. TROUBLESHOOTING

Users are welcome to report any issues/bugs on the GitHub Issues page.

2.4. CONTRIBUTING

Contributions to Si3D-L are welcome. Please reach out to the current developers for contribution guidelines.

Input Files

3.1. Description of Input Files

The purpose of this section is to provide guidance on the input files required to use the Si3D-L model. To properly use Si3D-L, the input files must reside within the same directory as the executable file of the model (i.e., 'psi3d'). This directory is where output files will be created. The simplest model requires the input file ('si3d_inp.txt'), the bathymetry file ('h'), and the initial condition file ('si3d_init.txt') to properly run the numerical model Si3D-L. This simplest model considers only an enclosed basing forced by wind stress and without a heat source at the surface boundary. However, the required input files for the numerical model to run properly variate based on the application. For instance, if heating/cooling from the atmosphere is modeled, the 'surfbc.txt' file is required (see 3.6). In addition, if open boundaries are used, such as river inflows, the corresponding 'openbc##.txt' file is needed for each open boundary in the model (see **??**).

3.2. Main Input File

The main input file in Si3D-L defines model-controlling parameters that determine the model characteristics, numerical techniques for solving the governing equations, and output types for post-processing purposes. The input file name must be named 'si3d_inp.txt', and an example of the file is provided within the "SampleFiles" folder named 'si3d_inp_master.txt'.

The input file has a total of 12 sections to edit. In all these sections, proper formatting of each line within the file is REQUIRED for proper use of the Si3D-L suite. Otherwise, Si3D-L will run into a reading error, and the model will stop. Each user-editable line within the file follows:

Variable_Name ! Variable_Value ! Variable_Description

where Variable_Name must have a length of 14 spaces (including the ! sign) and defines the variables being edited, Variable_Value must have 20 spaces and assigns the values to the corresponding variable name, and Variable_Description provides a brief explanation of the variable's meaning. In the following subsections, an explanation of each variable within the input file is presented along with the different options that Si3D-L suite offers.

3.2.1. Model Title

Line 2 of the file is available for users to edit with the model run name of their application.

3.2.2. Start date and time for simulations

This section is provided to the user to define the starting date of the model run. The section considers the definition of the year, month, day, and hour. For the hour, the format must follow military time (e.g., 0300 for 3:00 am, 1800 for 6:00 pm).

3.2.3. Space-time domains, cell size & time step

This section is provided for the user to define the space and time domains of the numerical model. The variables to be defined are:

xl – (m) stands for the length of the model domain in the EW direction or x coordinate. This variable divided by idx must be equal to the imx parameter within the bathymetry file.

yl – (m) stands for the length of the model domain in the SN direction or y coordinate. This variable divided by idy must be equal to the jmx parameter within the bathymetry file.

zl – (m) stands for the maximum depth of the model or the length of the domain in the vertical direction z.

tl – (sec) stands for the total time of model run. This should be at least equal to the number of hours defined in the input files for open boundaries and surface boundary conditions.

idx – (m) stands for the cell size of the model domain in the x coordinate.

idy – (m) stands for the cell size of the model domain in the y coordinate.

idz – (m) stands for the cell size of the model domain in the z coordinate. This value is used within the numerical model when a constant layer thickness is used to define the vertical mesh size.

$dzmin$ – (m) stands for the minimum cell size in the vertical direction. This value is used when .... NEEDS TO BE COMPLETED

$datadj$ – (m) variable to adjust the datum of the depths within the bathymetry file. The default value is 0.0

$zeta0$ – (m) variable to define the initial location of the water surface with respect to $z = 0$

idt – (sec) stands for the time step used to solve the numerical model. The user must consider the Courant–Friedrichs–Lewy criterion to define this parameter based on the model application.

$ibathf$ – defines if constant or variable layer thickness in the vertical mesh size is used. $ibathf = 0$ for constant layer thickness, and $ibathf = -1$ for variable layer thickness in the vertical direction. If $if bathf = -1$, a 'si3d_layer.txt' file, detailing the thicknesses of each layer, is needed as an input file (see 3.4)

3.2.4. Parameters controlling solution algorithm

This section allows users to define the parameters used within the numerical model to solve the governing equations. The variables to be defined are:

$itrap$ – defines whether the trapezoidal iteration or the single-step Leap Frog method is used in the solution of the finite difference discretization. $itrap = 0$ for single-step Leap Frog, and $itrap = 1$ for trapezoidal iterations.

$niter$ – defines the number of trapezoidal iterations to be used in the numerical methods. The greater the number of iterations, the longer time for the numerical model to find a solution for a time step.

$smooth$ – binary parameter for smoothing of Leap Frog solution. 1 is on and 0 is no smoothing.

$beta$ – parameter controlling the smoothing filter of the solution. $beta = 0.05 - 0.2$ is recommended.

$iturb$ – parameter controlling the method used for the turbulent closure method within the governing equations for the vertical direction. $iturb = 0$ for constant vertical eddy viscosity and diffusivity, and $iturb = 1$ to use the implemented 2-equation Mellor & Yamada turbulent closure method.

$az0$ – $(m2/s)$ constant vertical eddy viscosity. Used only when $iturb = 0$

$dz0$ – $(m2/s)$ constant vertical eddy diffusivity. Used only when $iturb = 0$

$iadv$ – binary parameter controlling if advection terms within the momentum governing equation are solved. $iadv = 0 \rightarrow$ OFF, and $iadv = 1 \rightarrow$ ON.

$itrmom$ – parameter controlling the algorithm used for solving the momentum horizontal advection. $itrmom = 1 \rightarrow$ centered scheme, $itrmom = 2 \rightarrow$ upwind scheme, $itrmom = 3 \rightarrow$, and $itrmom = 4 \rightarrow$ ... TO BE COMPLETED

ihd – parameter controlling horizontal diffusive terms within the governing equations. $ihd = 0 \rightarrow$ OFF, $ihd = 1 \rightarrow$ constant values, and $ihd = 2 \rightarrow$ Smagorinsky closure scheme.

$ax0$ – $(m2/s)$ constant horizontal eddy diffusivity and viscosity in the EW direction or x coordinate. This value is used when $ihd = 1$ for the governing equations of the velocity field, and at all times for the solution of the scalar transport equation.

$ay0$ – $(m2/s)$ constant horizontal eddy diffusivity and viscosity in the SN direction or y coordinate. This value is used when $ihd = 1$ for the governing equations of the velocity field, and at all times for the solution of the scalar transport equation.

f – $(1/s)$ constant value that defined the Coriolis frequency parameter.

$theta$ – defines the weighting parameter for the semi-implicit solution. TO BE COMPLETED

ibc – binary controlling parameter defining if baroclinc terms are included in the solution of the governing equation. $ibc = 0 \rightarrow$ OFF, and $ibc = 1 \rightarrow$ ON

$isal$ – binary parameter controlling if scalar transport equations is solved. $isal = 0 \rightarrow$ OFF, and $isal = 1 \rightarrow$ ON.

$itrsch$ – parameter controlling the algorithm used for solving the scalar transport equation. $itrsch = 1 \rightarrow$ centered scheme, $itrsch = 2 \rightarrow$ upwind scheme, $itrsch = 3 \rightarrow u$ at layer $k = k1z + 1$, and $itrsch = 4 \rightarrow$ for flux limiter scheme. NEEDS TO BE VERIFIED, I BELIEVE ONLY THE FLUX LIMITED IS USED

cd – parameter that defines the bottom drag coefficient of the model bottom. This value is constant in the whole domain.

$ifsbc$ – parameter controlling the type of surface boundary condition to be used in the solution of the numerical model. $ifsbc = 0 \rightarrow$ constant wind velocity w/o heat source. $ifsbc = 1 \rightarrow$ pre-process mode (i.e., heat budget prior to model run. $ifsbc = 2 \rightarrow$ runtime mode I, cloud cover as input. $ifsbc = 3 \rightarrow$ runtime II, incoming longwave is used as input. $ifsbc = 10 \rightarrow$ spatially variable runtime I. $ifsbc = 11 \rightarrow$ spatially variable runtime II. $ifsbc = 20 \rightarrow$ variable wind speed w/o heat sources. We refer the reader to Section 3.6 for more details on the different type of surface boundary conditions.

$dtsbc$ – (sec) parameter controlling the time step of the records used within the surface boundary conditions file (i.e., 'si3d_surbc.txt'). Only used if $ifsbc > 0$

cw – defines the wind drag coefficient for the quadratic stress law implemented for the wind-induced shear stress. This parameter is only used when $ifsbc = 0$

ws – (m/s) defines the wind speed for the numerical model when $ifsbc = 0$

phi – $(^\circ)$ defines the wind direction in degrees where the wind is coming from. This parameter is only used when $ifsbc = 0$

$idbg$ – binary parameter controlling if log messages are within the output of the model. This parameter is used for debugging purposes. $idbg = 0 \rightarrow$ OFF, and $idbg = 1 \rightarrow$ ON.

nth – parameter controlling the number of threads used for the parallelization of the model. This number coincides with the number of subdomains into which the model domain is divided into.

3.2.5. Output specifications for time files

This section allows the user to define outputs at specific nodes within the domain. The node location i, j must match the node definition from the bathymetry file. The user-editable parameters are:

ipt – parameter controlling the number of steps between consecutive output records. The output files are created for each node and follow 'tf*inodes_jnode*.txt'

$nnodes$ – defines the number of nodes to create outputs for. Integer number between 0 and 20

$inodes$ – defines the $i-location$ for the nodes to output. The $i-location$ for multiple nodes is specified in a comma-separated format.

$jnodes$ – defines the $j-location$ for the nodes to output. The $j-location$ for multiple nodes is specified in a comma-separated format.

3.2.6. Output specifications for horizontal planes

This section allows the user to define outputs of horizontal planes within the model domain. The number of the plane corresponds to a layer within the vertical grid dimensions of the model (e.g., for a constant layer thickness $dz = 1\ m$, plane 10 is to output results at a depth of $10\ m$. The following parameters are user-editable:

iht – parameter controlling the number of steps between consecutive output records. The output files are created for plane and follow 'plane_plane#'

$itspfh$ – parameter controlling the first time step at which results are saved at the specified planes.

$nplanes$ – number of planes to save and output results. If $ipxml < 0$ at least $plane\ 2$ must be saved as output.

$planes$ – vertical index of the layer/s to be saved and output results. The planes must be specified in a comma-separated format. Plane 2 corresponds to the surface layer.

3.2.7. Output specifications for 3D domain files

This section allows the user to define if the 3D domain will be saved during the model run. The user-editable parameters are:

$ipxml$ – parameter controlling the number of steps between consecutive outputs to the 3D file. The output file is 'ptrack_hydro.bnr'.

$itspf$ – parameter controlling the first time step at which results are saved. This parameter is recommended to be the same as $itspfh$ if *Paraview* is planned to be used for visualization and post-processing.

$iTurbVars$ – binary parameter controlling the outputs of the 3D file. $iTurbVars = 0 \rightarrow$ only velocity and temperature outputs, and $iTurbVars = 1 \rightarrow$ velocity, temperature, and turbulent parameter outputs.

3.2.8. Open boundary condition specifications

This section allows the user to define open boundaries in the model domain. Common open boundaries are inflows and outflows out of the lake. The user-editable parameters are:

$nopen$ – parameter controlling the number of open boundaries in the model

$dtscopenbc$ – (sec) parameter controlling the periodicity of records in the open boundary files used as input in the numerical model

$iside$ – comma-separated parameter controlling the side in the domain where the open boundaries are located. $1 \rightarrow$ West, $2 \rightarrow$ North, $3 \rightarrow$ East, and $4 \rightarrow$ South. The sides can only be specified in the cartesian coordinates.

$itype$ – comma-separated parameter controlling the type of open boundary. Thus, the inputs associated with the file for the open boundary. $1 \rightarrow$ Water Surface Elevation (WSE), $2 \rightarrow$ Surface Flow (Q), $3 \rightarrow$ subsurface flow (Q), and $4 \rightarrow$ when running a nested simulation with previously obtained results in a coarser grid.

$isbc$ – comma-separated parameter defining the initial i-location (i.e., cell) for the open boundary. This value is based on the bathymetry file

$jsbc$ – comma-separated parameter defining the initial j-location for the open boundary. This value is based on the bathymetry file

$iebc$ – comma-separated parameter defining the last i-location for the open boundary. This value is based on the bathymetry file

$jebc$ – comma-separated parameter defining the last j-location for the open boundary. This value is based on the bathymetry file

3.2.9. Open boundary conditions for nesting procedures

This section controls the creation of open boundary files when using the nesting procedures for models. The output files from this section are for each open boundary created for nesting procedures and follow the names 'nbofilei##' and 'nbofilex##'. The used-editable parameters are:

$nxBDO$ – parameter controlling the number of open boundaries created for nesting procedures.

iob – parameter controlling the number of time steps between consecutive records saved in the output files (i.e., the frequency of numerical results saved for nesting procedures in a finer grid model).

xxb – parameter controlling the scale between the coarse and fine numerical models for nesting procedures.

$iside$ – comma-separated parameter controlling the side in the domain where the open boundaries are located. $1 \rightarrow$ West, $2 \rightarrow$ North, $3 \rightarrow$ East, and $4 \rightarrow$ South. The sides can only be specified in the cartesian coordinates.

$isbc$ – comma-separated parameter defining the initial i-location (i.e., cell) for the open boundary. This value is based on the bathymetry file

$jsbc$ – comma-separated parameter defining the initial j-location for the open boundary. This value is based on the bathymetry file

$iebc$ – comma-separated parameter defining the last i-location for the open boundary. This value is based on the bathymetry file

$jebc$ – comma-separated parameter defining the last j-location for the open boundary. This value is based on the bathymetry file

3.2.10. Specification for water quality and tracer simulation

This section controls the use of passive tracers in the model or the use of the water quality model with the solution of the hydrodynamics. The user-editable parameters are:

ntr – parameter controlling the number of tracers to be modeled along the hydrodynamic model. This parameter must be equal to the parameters used for water quality and defined in 'si3d_wq_input.txt' when $ecomod == 1$.

$ecomod$ – parameter controlling the type of tracer modeling. $-1 \rightarrow$ tracer cloud modeling, $0 \rightarrow$ passive tracer, and $1 \rightarrow$ water quality model (AEM).

$iotr$ – parameter controlling the frequency of tracer solution outputs.

$itspftr$ – parameter controlling the first time step at which results of tracer concentrations in the 3D domain are saved.

$ipwq$ – parameter controlling the frequency of water quality solution (i.e., how often the water quality model is run).

3.2.11. Specification for oxygen system simulations

This section controls the use of models that consider analyses of oxygenation systems. The user-editable parameters are:

$ipts$ – parameter controlling the number of columns with point source-sinks

$Nodev$ – parameter controlling the number of devices acting as source-sinks

$iodt$ – (sec) parameter controlling the period between consecutive records in pss file

$ipss$ – $i - location$ of the device. The definition of multiple devices at different locations follows the comma-separated format

$jpss -$ $j - location$ of the device. The definition of multiple devices at different locations follows the comma-separated format

$iodev -$ id of the device controlling the source and sinks

3.2.12. Specification for interpolation method

This section controls the method used by Si3D-L to interpolate the spatially variable surface boundary conditions. The user-editable parameters are:

$iinterp -$

$gammaB -$

$delNfactor -$

3.3. Bathymetry File

The physical domain of a study volume is discretized on Cartesian rectangular grid cells horizontally of size $dx \times dy$. The model uses a z-coordinate system in the vertical direction. Thus, mesh cells are organized in a (i, j, k) space and k positive in the downward direction (Figure 3.1). The bathymetry file h' is designed to allow the user to input the geometry of the simulated domain at the resolution that the simulation will be conducted. The format of the bathymetry file is rigid about the amount of data provided and the order in which it is listed, as described at the end of this section.

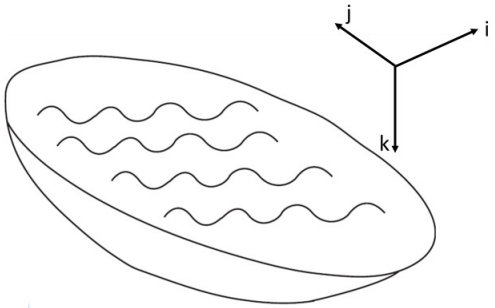

Figure 3.1: *Schematic of coordinate system*

3.3.1. Measured bathymetry (DEM, XYZ)

The horizontal grid is rectangular and divided into 'cells' whose centers and faces are respectively represented by solid dots and dashed lines in Figure 3.2. The horizontal mesh dimensions are $imax \times jmax$, which is large enough to fit the

domain of the water body. The pair (i, j) represents the location of each cell, with the (1,1) position always in the bottom left corner. The i index is the column index and increases to the right, and the j index is the row index and increases upward. The cells that fall outside of the water body domain are deactivated and called 'dry cells'. The active 'wet cells' vary for each row and column based on the shape of the water body. Note that there is a dry cell around the whole domain, which means that the bottom left corner starts on (2,2).

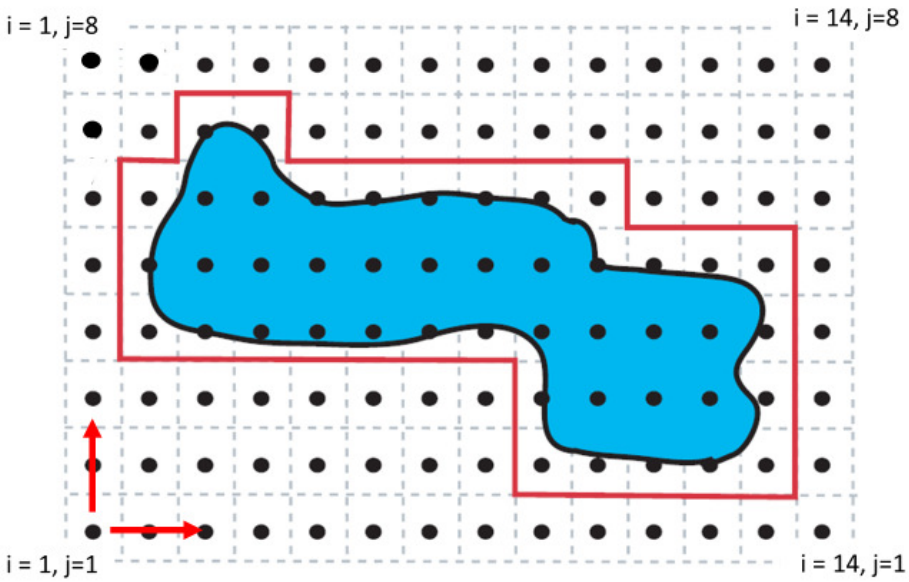

Figure 3.2: *Plan view of Cartesian mesh*

As a Cartesian grid, the horizontal mesh has a uniform spacing between the cell centers. The distance between two consecutive cells in the i and j directions are $dx = dy$.

Figure 3.3 shows the physical space outline of the lake as the border of the shaded area. The cell center dots have been replaced with numbers representing each water column's depth in $decimeters$. The dry cells have a value of -99. The wet cells have a positive value in decimeters.

The dimensions of the domain (xl, yl) must match the product between the number of columns/raw and the grid size. That is, $xl = imax * dx$ and $yl = jmax * dy$. In general, the time step (dt) should be chosen according to the Courant condition $dt < dx/u_{max}$, where u_{max} is the maximum modeled velocity. However, note that other stability restrictions may be limiting your time step (see Chapter 7, Table 7.1)

3.3.2. Canonical shapes

The physical domain of the simulation can be a canonical shape for exploratory analysis or any other research purpose. For Si3D-L we have tested the following

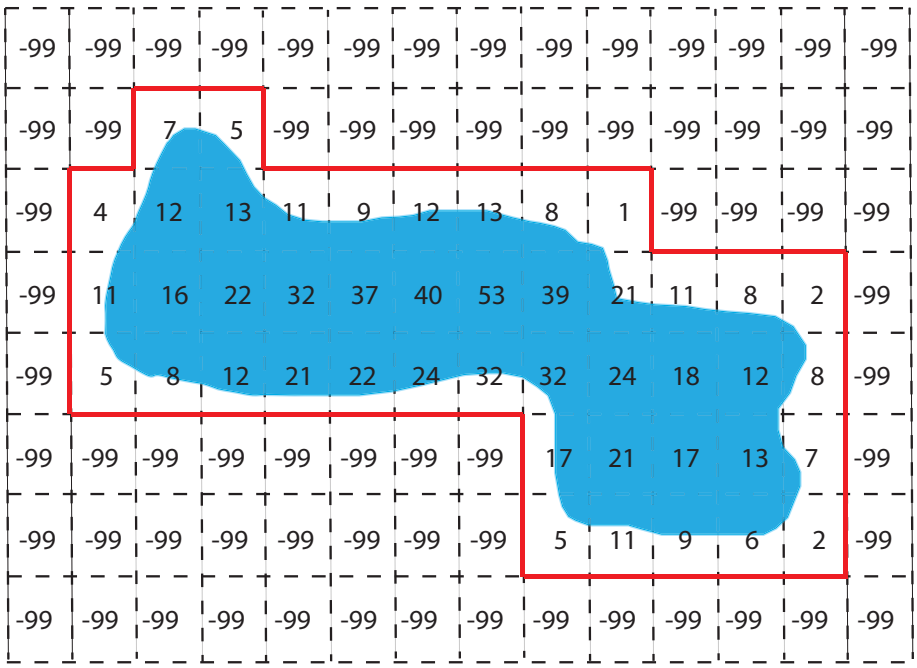

Figure 3.3: *Plan view of bathymetry data*

shapes:

- Rectangular, which characteristic dimensions are L = length, B = width, and H = depth
- Spherical, which characteristic dimensions are D = diameter, and H = depth
- Cylindrical, which characteristic dimensions are D = diameter, and H = depth

The file name will have the description of the grid size dx and the type of basin. This is for referencing, but the user must always change this name to 'h' in order for the bathymetry file to be properly read into the Si3D-L model.

3.3.3. Description of the bathymetry 'h' file

The first line of the 'h' file header contains dx = the grid dimensions, $imax$ = the number of cells in the i-direction (columns), $jmax$ = the number of cells in the j-direction (rows), and ncolums = the number of columns at each row ($= imax$). Please, notice the limited space for each number at this line of the header. For users reference, the Fortran format is as follows: Fortran format (37X,I5,6X,I5,8X,I5), and an example of the expected text is: $ClearLake(dx = 100m), imx = 239, jmx = 193, ncols = 239$. An incorrect number of spaces in the header will cause the model to crash during start up.

The second and third line of the header and first column are not read by the model. The third header line designates the number of the cell in the i-direction (i) and the first column represents the number of the cell in the j-direction (j). The matrix delimited by these two vectors contains the depth of the cells in the

study system, which must be written in decimeters. Dry cells are represented by the value -99. An example of bathymetry file 'h' can be found in the GitHub repository sub-folder si3d input

3.3.4. Generation of the bathymetry 'h' file

The 'h' file can be generated by using the function 'bathy4si3d' within the master function 'si3dInputs.py'. There is a similar function in MATLAB named 'bathy4si3d.m'. The code considers canonical and real basins depending on the option you choose.

The description of the function is provided in the function si3dInputs.py, and an example of how to use this function is provided in the function 'bathymetry.py'.

There is also a function available for generating an 'h' file from Georaster, reformatted original script with sphinx-autodoc formatting and more consistent PEP8 style 'bathy-file-maker.py'

3.4. INITIAL CONDITION FILE

Initial conditions for the velocity field, temperature, and tracers are required when using the Si3D-L suite. At the beginning of the simulation ($t = 0$), the values for the velocity field are assumed to follow those of a stagnant fluid (i.e., $u = v = w = 0\ ms^{-1}$), and the conditions for the water temperature and tracer concentrations are assumed to be horizontally uniform. Thus, to properly use the numerical model, it is necessary to define the vertical distribution for the temperature of the water column and the tracer concentration when $ntr > 0$. The initial vertical distribution of the water temperature and tracer concentration are defined within the 'si3d_init.txt' file, and two types of initial conditions are available.

3.4.1. Constant Vertical Distribution

The first type of initial condition defines a constant Δz at which the temperatures are defined at $zl/\Delta z + 2$ independent depths. The extra 2 depths correspond to the initial condition at the surface and bottom boundaries for the Dirichlet boundary condition of the numerical model and are often defined to be the same as the first and last depths of the water column. The 'si3d_init.txt' file is obtained by using the function 'initCond4si3d.py' embedded in si3dInputs.py script. The description of the function is provided in the Github repository, and an example of how to use this function is provided in 'InitialConditions.py'. The depth column in the generated file serves as a reference for the depth at the center of each layer in the numerical model, and only provides the number of rows to be read by the Si3D-L suite and thus assigns the corresponding water temperature and tracer concentration. A sample file for the'si3d_init.txt' file is provided in the SampleFiles folder.

3.4.2. Variable Vertical Distribution

The second type of initial condition defines variable Δz at which the temperatures and tracer concentrations are defined. For this case, $ibathf = -1$, and the 'si3d_layer.txt' file is necessary along with the 'si3d_init.txt' as input files for the Si3d-L suite. When specifying a variable Δz, the user must use $DeltaZ =' variable'$ and must define the maximum depth of the space domain H in the 'initCond4si3d.py' function embedded in si3dInputs.py script. The maximum depth H must be equal to zl from the 'si3d_inpt.txt' file. A sample file for the'si3d_layer.txt' file is provided in the SampleFiles folder.

The user has the option to choose from three different types of vertical spacing to define the distribution for depth, water temperature, and tracer concentration within the numerical model. The spacing methods available include exp, $sbconc$, and $surfvarBotconsta$, and a description of each is provided below. The selection of the spacing method depends on the user's discretion and the specific application, as the most suitable method may vary.

$exp -$ This method defines an initial layer thickness at the surface boundary and exponentially increases with depth. The input parameters for this option are $dz0s$ and $dzxs$ for the initial layer thickness and base of the exponential relationship, respectively.

$sbconc -$ This method specifies an initial layer thickness and base and the bottom and surface boundaries. This method results in a vertical distribution to be finer towards the boundary and coarser in the middle of the water column. In addition to the inputs for the $exp -$ method, the user must specify $dz0b$, $dzxb$, and Hn. The first two refer to the thickness and base of the exponential relationship at the bottom boundary, and the third parameter (i.e., Hn) defines the depth at which both depth-increasing layers meet.

$surfvarBotconsta -$ This method defines an initial layer thickness $dz0s$ at the surface boundary and a base $dzxs$ that increases exponentially with depth until a defined depth Hn. After this user-defined depth, the layers are distributed with a constant spacing dzc until the depth H.

3.5. BOUNDARY CONDITIONS FILE

Boundary conditions are set up by the user to provide environmental forcing information at the perimeter or the surface of the domain regarding changes in water temperature, velocity, or passive scalars (tracers) as a function of time. Therefore, here we introduce the velocity fields into and out of the domain (open boundary) or we force the model with a specific water surface elevation (WSE). Moreover, we should introduce the value of the scalars (and tracers) at the boundaries if we desire to model them.

When we set the velocity fields in or out of the domain (open boundary), we use flow discharges, thus, considering a homogeneous velocity field at the corresponding inflow-outflow section. Si3D-L computes the corresponding velocity

to each boundary cell according to its area $dxdz$ (or $dydz$). We have two methods to introduce a homogeneous flow into or out of the domain and the choice depends on the inclination of the boundary with respect to the x and y axes of the Cartesian grid in the 'h' file. When the boundary forms 90 degrees with the inflow/outflow, we implement the boundary condition as an $openbc.txt$ file. For other cases, we use a $pss.txt$ file.

3.5.1. Open Boundary Conditions (openbc)

With this condition, we can prescribe inflows or outflows (discharges) perpendicular at each boundary.

SI3D-L treats East and North as positive directions, therefore inflows are signed positive when prescribed at the West and South boundaries and negatives at North and East boundaries (i.e. they are flowing into the domain); the opposite is valid for outflows.

Time series for each inflow and/or outflow is specified in an individual text file (e.g. openbc01.txt, openbc02.txt, etc.). At each of the $openbc$ files, we include a value of discharge and temperature (and tracers if needed) for each of the time steps in the boundaries or dt_b. This dt_b is given in the $si3d.inp.txt$ file and represents the time during which discharges and scalars at a given boundary have constant values. The dt_b is independent of the computational time step (although it cannot be lower than the computational time step) and therefore, during several computational time steps, discharges and scalars can remain the same at a given boundary. Si3D-L solver does not read the time in the $openbc.txt$ file (first column), but it counts lines and assigns the corresponding value of discharge and temperature to the corresponding dt_b. For example if $dt_b = 1hour$, the code will read a new line each hour and within this hour the values for each computational dt will be the same.

In the case of outflow boundary conditions, we do probably not want to introduce a known value for the scalar or the tracer fields, but the boundary adopts the value of the immediately adjacent cell in the domain. In this case, we fill the columns of temperature and tracers with a constant through-time value of -20.

At the $si3d.inp.txt$ file we need to include the type of boundary ($itype$ = 2 = surface flow), the number of boundary conditions $nopen$ we have in our domain, dt_b and their locations (i, j cells). At each of the inflows/outflows, we specify if it is entering/leaving at the North, South, East, or West boundaries ($iside$) and the initial (is, js) and final (ie, je) wet cells of the inflow/outflow cross-section.

The $openbc$ file can be generated by using the function 'generate-OpenBoundary.m'. The code considers constant flow and scalars but can be modified to print variables values over time.

An example run with open boundary conditions can be found in the GitHub repository folder 'si3dBoundaryConditions'.

3.5.2. Water Surface Elevation Condition (openbc)

With this condition, we can restrict the oscillation of the WSE at each boundary. Boundaries must be perpendicular to the Cartesian grid. We can also restrict the oscillation of the WSE or completely neglect them (constant WSE) at a given boundary.

We use the same file name ($openbc.txt$) and structure as described in the Open Boundary Conditions section to prescribe WSE conditions, except for the column of discharges which is replaced by the values of WSE oscillations (m above, >+, or below,<0, the mean water surface elevation).

At the $si3d.inp.txt$ file we need to include the type of boundary ($itype$ = 1 = wse), the number of boundary conditions $nopen$ we have in our domain, dt_b and their locations (i, j cells). At each of the boundaries, we specify the side that is affected (North, South, East, or West boundaries $iside$), and the initial (is, js) and final (ie, je) wet cells of the WSE cross-section.

The $openbc$ file can be generated by using the function 'generate-WSEboundary.m'. The code considers constant WSE but can be modified to print variables values over time.

An example run with WSE boundary conditions can be found in the GitHub repository folder 'si3dBoundaryConditions'.

3.5.3. Point Source-Sink Boundary (pss)

When the boundary does not form 90 degrees with the inflow/outflow, we implement the boundary condition as a $pss.txt$ file. In this case, we can only simulate flow-type boundaries and not WSE. The pss files have several utilities besides the open boundary condition one. Si3D-L will treat this file as a boundary condition only if we indicate type -2 in the $Type$ line inside this file. In addition, this file will contain the following information: the number of records ($NumRec$s) in the pss file, the face of the cells through which the flow enters or leaves the domain (see below), the discharge threshold above which this boundary is activated ($qthresh$), and finally the columns of time, discharges (positive or negative according to the sign of the velocity fields), temperature and tracers as in the $openbc$ file.

In order to include the face of the cell through which the flow enters or leaves the domain we have to take into account the following criteria: uEpss, uWpss, vNpss, and vSpss represent the E, W, N, and S faces of the water cells (columns) at a given boundary. Therefore we introduce horizontal flows (the vertical component of the velocity is always 0 at the boundary). A value of 1 for this parameter implies the flow passes only through this face, and so the rest of the parameters must have a value of 0 (no flux through these faces). Values between 0 and 1 represent the fraction of each face through which the flow passes. We can combine these fractions to adjust the boundary to our objectives but taking into account that the value of the 4 faces must always add up to 1 (Figure 3.4)

To include the $pss.txt$ file information in the $si3d.inp.txt$ file, we have to introduce the total number of water columns that act as sources/sinks. For example, if we have two pss-type boundaries with 10 columns each, we indicate $iopss$ = 20.

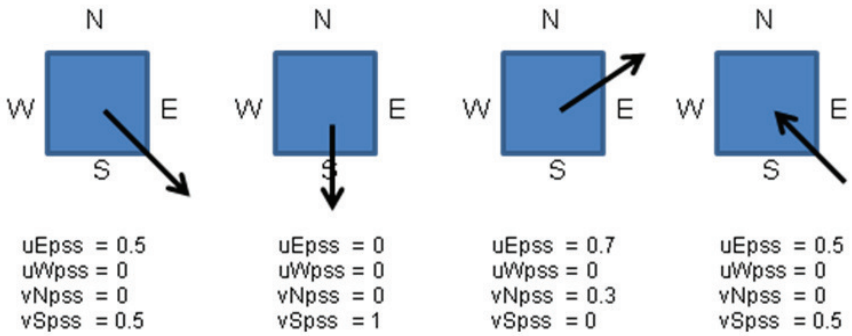

Figure 3.4: *Examples of values of uEpss, uWpss, vNpss, and vSpss and the resultant direction of the flow (pss)*

$Nodev$ refers to the number of boundaries, and $iodt = dt_b$. After this, we introduce the location (i, j) of each of the water columns in the boundaries ($iopssLoc$) followed by a number that identifies the number of the boundary of which they form part.

The $pss.txt$ file can be generated by using the function 'generate-pss.m'. The code considers constant values but can be modified to print variable values over time.

An example run with point source-sink boundary conditions can be found in the GitHub repository folder 'si3dBoundaryConditions'.

3.5.4. Nested Boundary Conditions

The main goal of this model capability is to run simulations of small portions of the complete domain at higher resolution in order to better reproduce small-scale processes at specific locations. Figure 3.5 illustrates a schematic setup of a nested grid. The nesting implementation allows defining the nesting boundary as any path of segments in the x and y (horizontal) directions, selecting an inner domain inside the outer domain. Figure 3.5 example uses a high-resolution in the inner grid (dx_{ig}) to resolve the sub-domain formed by the northeastern corner of the outer model with lower grid resolution (dx_{og}).

The outer domain is discretized using a structured grid with square cells of horizontal size dx_{og}. The boundary that separates the sub-basin from the boundary cells will be referred to as the I/O (Inner/Outer) boundary. The sub-basin in the inner model is discretized with cells of size $dx_{ig} < dx_{og}$. The ratio dx_{og}/dx_{ig} is the grid scaling parameter xxb which must be an integer value.

To run a nested simulation, we need to run two cases: one run with the outer coarse domain ($coarse$), followed by a second case within an inner fine domain run ($fine$). Each run is stored in an independent folder.

In the coarse run, the user must specify the I/O boundaries in the $si3d.inp.txt$ file in the section 'Open boundary conditions for nested procedures'. Here, the user will indicate the number of I/O boundaries to simulate $nxBDO$, the steps between consecutive records to output I/O conditions iob, the grid scaling parameter xxb, and for each I/O boundary the side that is affected (North, South, East, or West

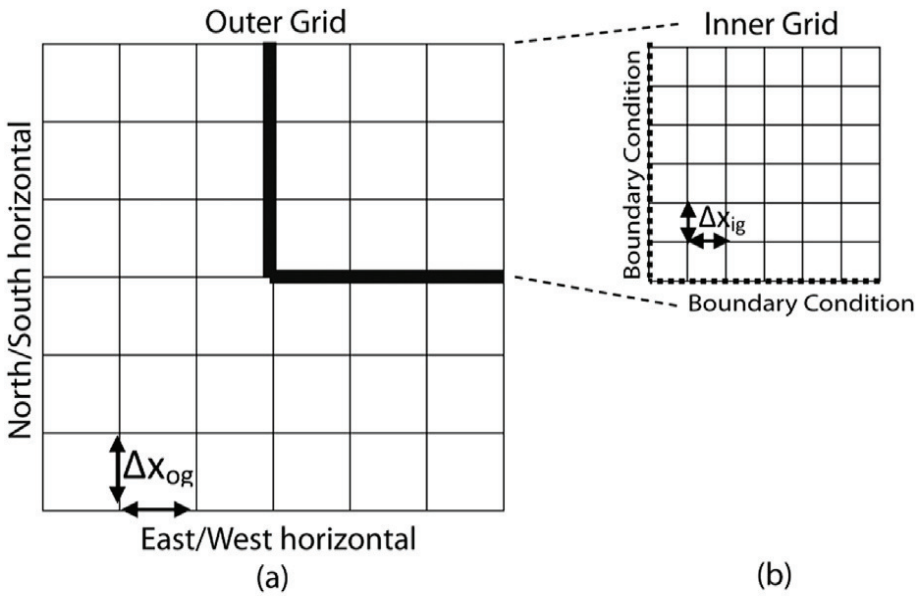

Figure 3.5: *Nesting grid example, (a) outer grid model or basin where dx_{og} is East-West and North-South horizontal resolution and (b) inner grid model or sub-basin inside the outer grid where horizontal resolution dx_{ig} is half the outer model horizontal resolution.*

boundaries $iside$), and the initial (is, js) and final (ie, je) wet cells of I/O cross-section. The coarse run will generate an output file starting with 'nbo-'. This file must be moved to the fine folder run BUT it should be renamed and start with 'nbi-' instead. This/these file(s) will act as open boundary conditions (openbc) in the fine run.

In the fine run, the user will specify the I/O boundaries in the $si3d.inp.txt$ file in the section 'Open boundary conditions specifications'. As described in the 'Open Boundary Conditions' section, we will indicate the number of boundaries $nopen$, the time step between consecutive records in seconds $dtsecopenbc$, the side that is affected, the type ($itype$ = 4 = nested boundary), and the initial (is, js) and final (ie, je) wet cells of open boundary cross-section in the inner domain.

Another important aspect of the nested runs is creating the correct bathymetry files (h) for the coarse and fine runs to match the grid scaling parameter. The user should use the script available in the GitHub repository folder 'si3dBoundaryConditions'

An example run nested boundary conditions can be found in the GitHub repository folder 'si3dBoundaryConditions'.

3.6. Surface Boundary Condition File

The Si3D-L suite utilizes surface boundary conditions to define wind forcing and heat sources during model runs. Users provide this boundary forcing through the 'si3d_surfbc.txt' file, with the data varying based on the user-defined variable $ifsbc$. If $ifsbc = 0$, the 'si3d_surfbc.txt' file is not required, and users must specify the wind speed ws, wind drag coefficient cw, and wind direction phi. In this model setting, there are no heat sources from atmospheric conditions. If $ifsbc > 0$,

there are options available for surface boundary conditions in the form of spatially uniform and spatially variable conditions. In these cases, users need to specify the time step, denoted as $dtsbc$, in the si3d_inp.txt' file which specifies the frequency at which data is provided in the si3d_surfbc.txt' file. The 'si3d_surfbc.txt' file is created using the script 'surfbc4si3d.py' from the GitHub repository si3dInputs embedded in si3dInputs.py script, and an example of how to run the script is provided in si3dSurfBoundCond.

3.6.1. Spatially uniform surface boundary conditions

Spatially uniform conditions are used when $ifsbc = 1$, 2, 3, or 20. The forcing is then assumed to be from only one meteorological station, and differences among the options lie in the data used by the Si3D-L suite. Sample files for each type of surface boundary condition can be accessed in the 'SampleFiles' folder.

$ifsbc = 1$ corresponds to a *preprocess mode*, where a heat budget is calculated by the user to estimate the heat fluxes prior to the model run and used to create the 'si3d_surfbc.txt' file. Users may estimate their own heat budget or use one of the available options within the 'surfbc4si3d.py' script. Moreover, if $ifsbc = 2$ or $ifsbc = 3$ are specified, the model calculates the surface heat fluxes and reads the meteorological conditions from the surface boundary conditions file. While $ifsbc = 2$ uses cloud cover to estimate incoming longwave radiation, $ifsbc = 3$ uses measured incoming longwave radiation. Finally, when $ifsbc = 20$, heat sources are assumed to be negligible and only wind forcing time series is considered and read by the model from the 'si3d_surfbc.txt' file. A summary of the inputs required by the Si3D-L suite for each type of surface boundary condition (i.e., $ifsbc = 1$, 2, 3, or 20) is presented below. The parameters are required within the 'si3d_surfbc.txt' file if the type of surface boundary condition is specified in the **ifsbc** column.

Table 3.1: *Summary of variables needed in the 'si3d_surfbc.txt' file for the Si3D-L suite for each $ifsbc$ option.*

Parameter	Description	Units	ifsbc
thr	Time of data for meteorological conditions	hrs	1, 2, 3, 10,11, 20
eta	Light extinction coefficient	m^{-1}	1, 2, 3, 10, 11
Qn	Net radiation. Includes shortwave radiation	Wm^{-2}	1
SW_{net}	Net shortwave radiation	Wm^{-2}	1, 2, 3, 10, 11
Ta	Air Temperature	$^{\circ}C$	2, 3, 10, 11
Pa	Atmospheric pressure	Pa	2, 3, 10, 11
RH	Relative humidity	$Fraction$	2, 3, 10, 11
cc	Cloud cover	$Fraction$	2, 10
LW_{in}	Incoming longwave radiation	Wm^{-2}	3, 11
cw	Wind drag coefficient	$-$	1, 2, 3,10, 11, 20
u_{air}	u-direction wind speed (EW)	ms^{-1}	1, 2, 3, 10, 11, 20
v_{air}	v-direction wind speed (SN)	ms^{-1}	1, 2, 3, 10, 11, 20

3.6.2. Spatially variable surface boundary conditions

Spatially variable conditions are used when $ifsbc = 10$ or 11, which correspond to the methods used for calculating uniform surface heat fluxes when $ifsbc = 2$ and $ifsbc = 3$ respectively. Forcing data is provided at specified nodes within the model domain and linearly interpolated to provide a spatially variable wind and heat flux field. An example file for when $ifsbc = 11$ can be accessed in the 'Sample Input Files' folder.

When spatially variable conditions are specified Pa and eta are assumed uniform while other input parameters can be kept constant or varied between each specified node. The first 6 heading lines of the 'si3d_surfbc.txt' file for variable conditions is the same as for uniform conditions but two additional lines are required to specify the number and location of nodes use for interpolation. On line 7 the number of nodes should be stated (e.g. if four stations stations are being interpolated between this line should read $No.Stats4$). On line 8, the corresponding i-location and j-location of each note should be stated. The subsequent lines show the time series of surface forcing, with data in the first 3 columns being thr Pa and eta shared among all stations followed by columns of each parameter set and grouped by station. The order of the input data should follow the station order specified on line 8.

Tutorials

Output Files

Theory

In this section, the theory and governing equations behind Si3D-L are presented. The model solved the Reynolds-Averaged Navier-Stokes equations, using the Boussinesq approximation for density variations, and assuming hydrostatic pressure distribution. $ifsbc = 11$ vs `ifsbc = 11`

6.1. GOVERNING EQUATIONS

6.1.1. Fluid flow equations

The mass conservation equation for an incompressible fluid is:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{6.1}$$

where u_i is the $i-$th component of the fluid's velocity vector. The three components x_1, x_2 and x_3 correspond to x, y and z, respectively, and will be used interchangeably. The x and y directions correspond to the horizontal direction, while z corresponds to the vertical direction, with ascending values going from the bottom of the lake toward the free surface. The momentum conservation equation with the Boussinesq approximation (only conserve variation in density in terms that affect weight) reads:

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_j u_i}{\partial x_j} = -\frac{1}{\rho_0}\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu_{\text{eff},i}\frac{\partial u_i}{\partial x_j} - \frac{2}{3}k\delta_{ij}\right) + \frac{\rho}{\rho_0}g_i - 2\epsilon_{ijk}\Omega_j u_k \tag{6.2}$$

where Ω_j is the component of the angular velocity of the earth in the direction x_j, ρ_0 is the reference value for density, $\nu_{\text{eff},i} = \nu + \nu_{t,j}$ is the effective viscosity, and ρ is the density[1][2]. An equation of state gives density as a function of temperature (T) and salinity (s):

$$\rho = f(s, T) \tag{6.3}$$

Treatment of the pressure term

Only the pressure and gravity terms are retained in the x_3-momentum equation

$$\rho g + \frac{\partial p}{\partial z} = 0 \tag{6.4}$$

[1]Note that in Smith they denote $p = p_0 + p'$ and $\partial p_0/\partial x_i = -\rho_0 g_i$, and in the final equation he has $\partial p'/\partial x_i$ (instead of $\partial p/\partial x_i$) and $\rho' g_i/\rho_0$ (instead of $\rho g_i/\rho_0$).

[2]The turbulence viscosity $\nu_{t,j}$ has a sub-index j because in lakes and estuaries, horizontal and vertical gradients are sometimes treated separately. Smith ignores the molecular viscosity and the $2/3k\delta_{i,j}$ term.

Using this assumption, the pressure gradient terms in the x and y momentum equation can be rewritten as,

$$\frac{1}{\rho_0}\frac{\partial p}{\partial x_i} = \frac{1}{\rho_0}\frac{\partial p_a}{\partial x_i} + g\frac{\partial \zeta}{\partial x_i} + g\frac{1}{\rho_0}\int_z^\zeta \frac{\partial \rho}{\partial x_i} dz' \tag{6.5}$$

where ζ is the free-surface elevation measured from a reference height. The first term on the RHS considers the variation in atmospheric pressure (p_a) (this term can be neglected). The second term is referred to as the barotropic term and considers the variation in pressure due to water surface slope, and the third term is known as the baroclinic term and considers gradients in pressure due to variation in density.

With this simplification, Eq. 6.2 can be rewritten as:

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_j u_i}{\partial x_j} = -g\frac{\partial \zeta}{\partial x_i} - g\frac{1}{\rho_0}\int_z^\zeta \frac{\partial \rho}{\partial x_i} dz' + \frac{\partial}{\partial x_j}\left(\nu_{\text{eff},i}\frac{\partial u_i}{\partial x_j} - \frac{2}{3}k\delta_{ij}\right) + \frac{\rho}{\rho_0}g_i - 2\epsilon_{ijk}\Omega_j u_k \tag{6.6}$$

Boundary conditions

The elevation of the free-surface can be calculated by considering continuity (kinematic boundary condition):

$$\frac{\partial \zeta}{\partial t} + u_1\frac{\partial \zeta}{\partial x_1} + u_2\frac{\partial \zeta}{\partial x_2} = 0 \tag{6.7}$$

For the velocity, equilibrium of shear stresses can be performed. Neglecting the slope of the free surface, the BC results from the equilibirum between shear-stresses and the stress created by wind:

$$\tau_{is} = \tau_{i3} \tag{6.8}$$

$$c_w \rho_a W_a^2 \hat{\mathbf{x}}_i = \rho_0 \nu_{\text{eff},i}\frac{\partial u_i}{\partial x_3} \tag{6.9}$$

where c_w is the dimensionless drag coefficient, W_a is the wind speed 10 m above the free surface, and $\hat{\mathbf{x}}_i$ is a unit vector in direction of x_i

Similarly, in the bottom, the kinematic boundary condition states

$$u_1\frac{\partial h}{\partial x_1} + u_2\frac{\partial h}{\partial x_2} + u_3 = 0 \tag{6.10}$$

where h represents a vertical distance measured positive downward from a datum (such as a mean sea level height) above the bottom. The equilibirum of shear stresses is considered:

$$\tau_{ib} = \tau_{i3} \tag{6.11}$$

$$\rho_0 C_d (u_j u_j)^{0.5} u_i = \rho_0 \nu_{\text{eff},i}\frac{\partial u_i}{\partial x_3} \tag{6.12}$$

In the lateral walls, slip conditions are applied.

6.1.2. Conservation of salinity

6.1.3. Conservation of temperature

6.2. LAYERED AVERAGED EQUATIONS

Governing equations are formulated in terms of layer-averaged variables. To introduce the layer-averaged formulation, a small description of the mesh structure is necessary. The domain is divided into vertical layers of variable height as shown in Figure 6.1. Layers are indexed with the index k from 1 (top layer) up to k_m (bottom layer). The height of each layer is indicated with the variable h_k. The location of the top boundary of the layer i is represented by the variable $z_{i-1/2}$, while the lower boundary is indicated with the variable $z_{i+1/2}$. Intermediate layers ($k \in [2, k_m - 1]$) have a uniform height, i.e, $h_k =$ constant and $z_{i\pm 1/2} =$ constant. The top and the bottom layers have variable height, i.e., $h_1(x, y)$ and $h_{km}(x, y)$. The top layer has a uniform lower boundary ($z_{i+1/2} =$ constant), and a variable upper boundary that follows the position of the free-surface ($z_{i-1/2} = \zeta$). The bottom layer, has a uniform upper boundary ($z_{km-1/2} =$ constant), and a variable lower boundary that follows the position of the bottom ($z_{km+1/2} = z_b$).

Figures/layers_scheme.pdf

Figure 6.1: *Caption*

Layer-averaged equations are formulated in terms of layer-averaged variables. The average of any 3-D variable over a layer k will be indicated by a subscript k. The layer-averaged value of a variable ϕ is defined as,

$$\phi_k = \frac{1}{h_k} \int_{z_{k-1/2}}^{z_{k+1/2}} \phi \, dz \tag{6.13}$$

where the integration limits $z_{k-1/2}$ and $z_{k+1/2}$ define the z coordinates of the upper and lower layer interfaces, respectively. The coordinate $z_{1/2} = \zeta$ represents the free surface elevation, and $z_{km+1/2} = z_b$ represents the bottom elevation measured from the datum. The quantities $u_k h_k$ and $v_k h_k$ are volumetric transports and are represented by the symbols $U_k = u_k h_k$ and $V_k = v_k h_k$. The volumetric transports are main variables in the model; the velocities u_k and v_k are computed from the volumetric transports by dividing by h_k.

To derive the governing equations based on layer-averaged variables, the Leibnitz' rule is used. For a generic variable $\phi(x, y, z, t)$, the Leibnitz rule, applied to an integral over a layer is:

$$\frac{\partial}{\partial x_i}\int_{z_{k-1/2}}^{z_{k+1/2}} \phi \, dz = \int_{z_{k-1/2}}^{z_{k+1/2}} \frac{\partial \phi}{\partial x_i} \, dz + \phi_{k-1/2}\frac{\partial z_{k-1/2}}{\partial x_i} - \phi_{k+1/2}\frac{\partial z_{k+1/2}}{\partial x_i} \tag{6.14}$$

which rearranging leads to:

$$\int_{z_{k-1/2}}^{z_{k+1/2}} \frac{\partial \phi}{\partial x_i} \, dz = \frac{\partial (h_k \phi_k)}{\partial x_i} - \phi_{k-1/2}\frac{\partial z_{k-1/2}}{\partial x_i} + \phi_{k+1/2}\frac{\partial z_{k+1/2}}{\partial x_i}. \tag{6.15}$$

In addition, the fundamental theorem of calculus is used:

$$\int_{z_{k-1/2}}^{z_{k+1/2}} \frac{\partial \phi}{\partial z} \, dz = \phi_{z_{k+1/2}} - \phi_{z_{k-1/2}} \tag{6.16}$$

6.2.1. Continuity equation

Integrating Equation (6.1) over layer k, and applying the identities (6.15) and (6.16) we get,

$$\begin{aligned} \frac{\partial U_k}{\partial x} + \frac{\partial V_k}{\partial y} - \left[u_{k-1/2}\frac{\partial z_{k-1/2}}{\partial x} + v_{k-1/2}\frac{\partial z_{k-1/2}}{\partial y} \right] \\ + \left[u_{k+1/2}\frac{\partial z_{k+1/2}}{\partial x} + v_{k+1/2}\frac{\partial z_{k+1/2}}{\partial y} \right] + \left[w_{k+1/2} - w_{k-1/2} \right] = 0 \end{aligned} \tag{6.17}$$

The terms between brackets are null for every layer ($z_{k+1/2}$ and $z_{k-1/2}$ are constant in the horizontal directions), except the surface and bottom layers. In the middle layers, the continuity equation is rewritten as:

$$\frac{\partial U_k}{\partial x} + \frac{\partial V_k}{\partial y} + \left[w_{k+1/2} - w_{k-1/2} \right] = 0 \tag{6.18}$$

In the surface layer ($k = 1$), $\dfrac{\partial z_{3/2}}{\partial (x, y)} = 0$ and $w_{1/2} = 0$. In addition, $\dfrac{\partial z_{1/2}}{\partial (x, y)} = \dfrac{\partial \zeta}{\partial (x, y)}$. This results in:

$$\frac{\partial U_1}{\partial x} + \frac{\partial V_1}{\partial y} - \left[u_{1/2}\frac{\partial \zeta}{\partial x} + v_{1/2}\frac{\partial \zeta}{\partial y} \right] - w_{3/2} = 0 \tag{6.19}$$

Noting that $z_{1/2}$ represents the upper boundary, the term between brackets can be replaced with the kinematic BC (Eq. (6.7)), then we obtain the mass conservation equation for the upper layer:

$$\frac{\partial \zeta}{\partial t} + \frac{\partial U_1}{\partial x} + \frac{\partial V_1}{\partial y} - w_{3/2} = 0 \tag{6.20}$$

It can be shown that

$$w_{3/2} = -\sum_{k=2}^{k_m} \left[\frac{\partial U_k}{\partial x} + \frac{\partial V_k}{\partial y} \right], \tag{6.21}$$

and therefore Equation (6.20) can be rewritten as:

$$\frac{\partial \zeta}{\partial t} + \frac{\partial}{\partial x} \sum_{k=1}^{k_m} U_k + \frac{\partial}{\partial x} \sum_{k=1}^{k_m} V_k = 0 \tag{6.22}$$

which is the equation used in the code. Similarly, applying the kinematic boundary condition for the bottom (Eq. (6.9)),

$$\frac{\partial U_{km}}{\partial x} + \frac{\partial V_{km}}{\partial y} - w_{km-1/2} = 0 \tag{6.23}$$

6.2.2. Momentum equation

Integrating Equation (6.2) over layer k we get:

$$\int_{z_{k-1/2}}^{z_{k+1/2}} \left[\frac{\partial u_i}{\partial t} + \frac{\partial u_j u_i}{\partial x_j} \right] dz = \int_{z_{k-1/2}}^{z_{k+1/2}} \left[-\frac{1}{\rho_0} \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \left(\nu_{\text{eff},i} \frac{\partial u_i}{\partial x_j} - \frac{2}{3} k \delta_{ij} \right) + \frac{\rho}{\rho_0} g_i - 2\epsilon_{ijk} \Omega_j u_k \right] dz \tag{6.24}$$

The temporal derivative and advective terms (L.H.S) can be rewritten as:

$$\begin{aligned} &\frac{\partial U_k}{\partial t} + \frac{\partial}{\partial x} \int_{z_{k-1/2}}^{z_{k+1/2}} uu \, dz + \frac{\partial}{\partial y} \int_{z_{k-1/2}}^{z_{k+1/2}} uv \, dz \\ &\qquad - u_{k-1/2} \left[\frac{\partial z_{k-1/2}}{\partial t} + u_{k-1/2} \frac{\partial z_{k-1/2}}{\partial x} + v_{k-1/2} \frac{\partial z_{k-1/2}}{\partial y} - w_{k-1/2} \right] \\ &\qquad + u_{k+1/2} \left[0 + u_{k+1/2} \frac{\partial z_{k+1/2}}{\partial x} + v_{k+1/2} \frac{\partial z_{k+1/2}}{\partial y} - w_{k+1/2} \right] \end{aligned} \tag{6.25}$$

At the free surface and bottom, the bracketed terms disappear because of the kinematic boundary conditions. At all other layer interfaces, the derivatives of $z_{k+1/2}$ and $z_{k+1/2}$ equal zero. The integral terms can be rewritten as:

$$\begin{aligned} \int_{z_{k-1/2}}^{z_{k+1/2}} uu \, dz &= (Uu)_k + \int_{z_{k-1/2}}^{z_{k+1/2}} (u - u_k)^2 \, dz \\ \int_{z_{k-1/2}}^{z_{k+1/2}} uv \, dz &= (Vu)_k + \int_{z_{k-1/2}}^{z_{k+1/2}} (u - u_k)(v - v_k) \, dz \end{aligned} \tag{6.26}$$

The second terms are sometimes referred to as the momentum dispersion terms, and they will be neglected. The final formulation for the acceleration terms in the layer-integrated equations is

$$\int_{z_{k-1/2}}^{z_{k+1/2}} \left[\frac{\partial u_i}{\partial t} + \frac{\partial u_j u_i}{\partial x_j} \right] dz = \frac{\partial U_k}{\partial t} + \frac{\partial (Uu)_k}{\partial x} + \frac{\partial (Vu)_k}{\partial y} \qquad k = (1, k_m) \tag{6.27}$$

$$= \frac{\partial U_k}{\partial t} + \frac{\partial (Uu)_k}{\partial x} + \frac{\partial (Vu)_k}{\partial y} + (uw)|_{k+1/2}^{k-1/2} \qquad k \neq (1, k_m) \tag{6.28}$$

Applying Leibniz rule, the pressure term can be rewritten as:

$$\frac{1}{\rho}\int_{z_{k-1/2}}^{z_{k+1/2}} \frac{\partial p}{\partial x}\, dz = \frac{1}{\rho}\left[\frac{\partial h_k p_k}{\partial x} - p_{k-1/2}\frac{\partial z_{k-1/2}}{\partial x} + p_{k+1/2}\frac{\partial z_{k+1/2}}{\partial x}\right] \tag{6.29}$$

Considering that for the interfaces between layers, $\partial z_{k\pm 1/2}/\partial x = 0$, and that the atmospheric pressure $p_{1/2}$, the expression can be simplified as:

$$\frac{1}{\rho}\frac{\partial h_1 p_1}{\partial x} = \frac{h_1}{\rho}\frac{\partial p_1}{\partial x} + \frac{p_1}{\rho}\frac{\partial \zeta}{\partial x} = \frac{h_1}{\rho}\frac{\partial p_1}{\partial x} + \frac{h_1 g \rho_1}{2\rho}\frac{\partial \zeta}{\partial x} + E_1 \qquad \text{[First layer]} \tag{6.30}$$

$$\frac{h_k}{\rho}\frac{\partial p_k}{\partial x} \qquad \text{[Middle layers]} \tag{6.31}$$

$$\frac{1}{\rho}\frac{\partial h_{km} p_{km}}{\partial x} + \frac{p_{km+1/2}}{\rho}\frac{\partial z_{km+1/2}}{\partial x} = \frac{h_{km}}{\rho}\left(\frac{\partial p_{km}}{\partial x} - \frac{g\rho_{km}}{2}\frac{\partial h_{km}}{\partial x}\right) \qquad \text{[Bottom layer]} \tag{6.32}$$

Last one I'm confused what Smith does (3.37) (6.33)

...

By the end, we have for the top layer

$$\frac{1}{\rho}\int_{z_{1-1/2}}^{z_{1+1/2}} \frac{\partial p}{\partial x} dz = \frac{h_1}{\rho_0}\left[g\rho_1\frac{\partial \zeta}{\partial x} + \frac{g h_1}{2}\frac{\partial \rho_1}{\partial x}\right]; \tag{6.34}$$

for the intermediate layers

$$\frac{1}{\rho}\int_{z_{k-1/2}}^{z_{k+1/2}} \frac{\partial p}{\partial x} dz = \frac{h_k}{\rho_0}\left[g\rho_1\frac{\partial \zeta}{\partial x} + \frac{g h_1}{2}\frac{\partial \rho_1}{\partial x} + \sum_{m=2}^{k}\left(\frac{g h_{m-1}}{2}\frac{\partial \rho_{m-1}}{\partial x} + \frac{g h_m}{2}\frac{\partial \rho_m}{\partial x}\right)\right]; \tag{6.35}$$

and for the last layer...

For the shear stresses, the result is

$$\frac{\partial}{\partial x}\left(\nu_{\text{eff},H} h \frac{\partial v}{\partial x}\right)_k + \frac{\partial}{\partial y}\left(\nu_{\text{eff},H} h \frac{\partial v}{\partial y}\right)_k + \frac{\tau_{yz}|_{k+1/2}^{k-1/2}}{\rho} \tag{6.36}$$

And for the coriolis force...

6.2.3. Summary of layer-averaged equations

Continuity equations

$$\frac{\partial \zeta}{\partial t} + \frac{\partial}{\partial x}\sum_{k=1}^{k_m} U_k + \frac{\partial}{\partial x}\sum_{k=1}^{k_m} V_k = 0 \tag{6.37}$$

$$\frac{\partial U_{km}}{\partial x} + \frac{\partial V_{km}}{\partial y} - w_{km-1/2} = 0 \tag{6.38}$$

Momentum equations

$$\frac{\partial U_k}{\partial t} + \frac{\partial (Uu)_k}{\partial x} + \frac{\partial (Vu)_k}{\partial y} + (uw)|_{k+1/2}^{k-1/2} - fV_k + \frac{h_k}{\rho_k} g\rho_1 \frac{\partial \zeta}{\partial x} = - \frac{h_k}{\rho_k}\left[\frac{gh_1}{2}\frac{\partial \rho_1}{\partial x} + \sum_{m=2}^{k}\left(\frac{gh_{m-1}}{2}\frac{\partial \rho_{m-1}}{\partial x} + \frac{gh_m}{2}\frac{\partial \rho_m}{\partial x}\right)\right] + \frac{\partial}{\partial x}\left(A_H h \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(A_H h \frac{\partial u}{\partial y}\right)_k + \left(\frac{\tau_{xz}}{\rho}\right)\Bigg|_{k+1/2}^{k-1/2} \tag{6.39}$$

$$\frac{\partial V_k}{\partial t} + \frac{\partial (Uv)_k}{\partial x} + \frac{\partial (Vv)_k}{\partial y} + (vw)|_{k+1/2}^{k-1/2} + fU_k + \frac{h_k}{\rho_k} g\rho_1 \frac{\partial \zeta}{\partial y} = - \frac{h_k}{\rho_k}\left[\frac{gh_1}{2}\frac{\partial \rho_1}{\partial y} + \sum_{m=2}^{k}\left(\frac{gh_{m-1}}{2}\frac{\partial \rho_{m-1}}{\partial y} + \frac{gh_m}{2}\frac{\partial \rho_m}{\partial y}\right)\right] + \frac{\partial}{\partial x}\left(A_H h \frac{\partial v}{\partial x}\right)_k + \frac{\partial}{\partial y}\left(A_H h \frac{\partial v}{\partial y}\right)_k + \left(\frac{\tau_{yz}}{\rho}\right)\Bigg|_{k+1/2}^{k-1/2} \tag{6.40}$$

The mean density ρ has been substituted for the layer-averaged density ρ_k in the denominator of the pressure, vertical stress, and vertical salt flux terms; this substitution requires little effort to implement in the numerical model and reduces any error caused by the Boussinesq approximation.

Numerical Implementation

7.1. MESH DEFINITION

The mesh layers are organized using a z-coordinate system. Layers are indexed with $k \in (1, k_m)$, where $k = 1$ represents the surface layer and $k = k_m$ is the bottom layer. Mesh cells are organized in a (i, j, k) coordinate. Intermediate layers ($k \in [2, k_m - 1]$) have a uniform height, i.e, $h_k =$ constant and $z_{i\pm 1/2} =$ constant. The surface layer height (h_1) varies as a function of both time and space because of the propagation of the tides and other waves ($h_1 = (i, j, t)$). The bottom layer height (h_{km}) varies with the changing bathymetry as a function of space only ($h_{km}(i, j)$). Because no wetting and drying of nodal points is considered, it is assumed that the free surface never drops below the bottom of the first layer. The use of equal layer heights provides a convenient implementation of a vertical finite-difference approximation that is second-order accurate in space. If uneven layer heights are chosen, the vertical numerical approximation becomes only first-order accurate.

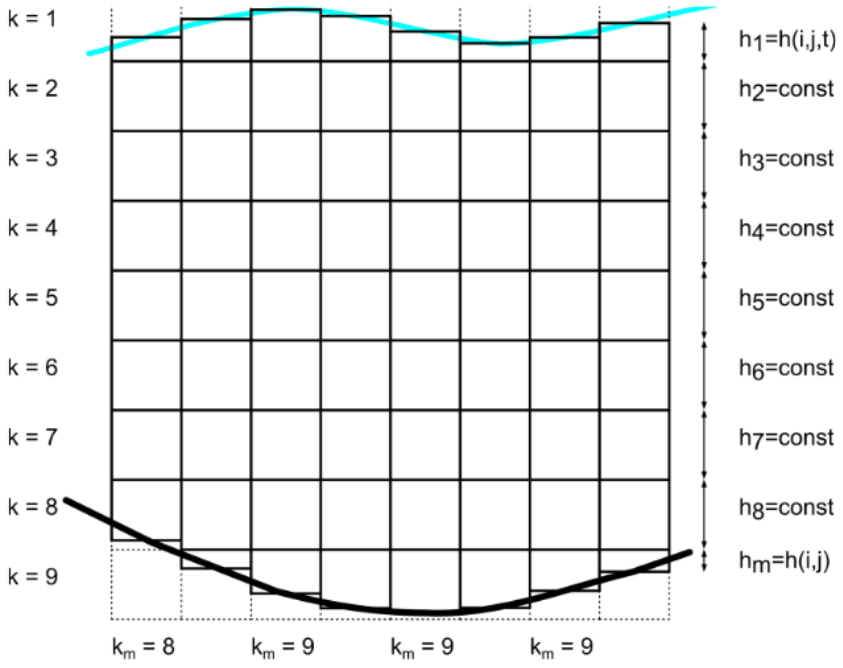

Figure 7.1: *Caption*

Horizontally, the grid is rectangular with dimensions $i_{\max} \times j_{\max}$. Cells that fall outside of the boundaries are deactivated. The number of activated cells vary for each row and column, based on the shape of the water bodies. Wet cells are indexed for each row j by indices $i_j \in (i_{1,j}, i_{m,j})$, where $i_{1,j}$ indicates the i

index of the first cell activated in row j, while $i_{m,j}$ indicates the i index of the last cell activated in row j. Similarly, for each colum i, wet cells are indexed by $j_i \in (j_{1,i}, j_{m,i})$, where $j_{1,i}$ indicates the j index of the first cell activated in column i, while $j_{m,i}$ indicated the i index of the last cell activated in column i.

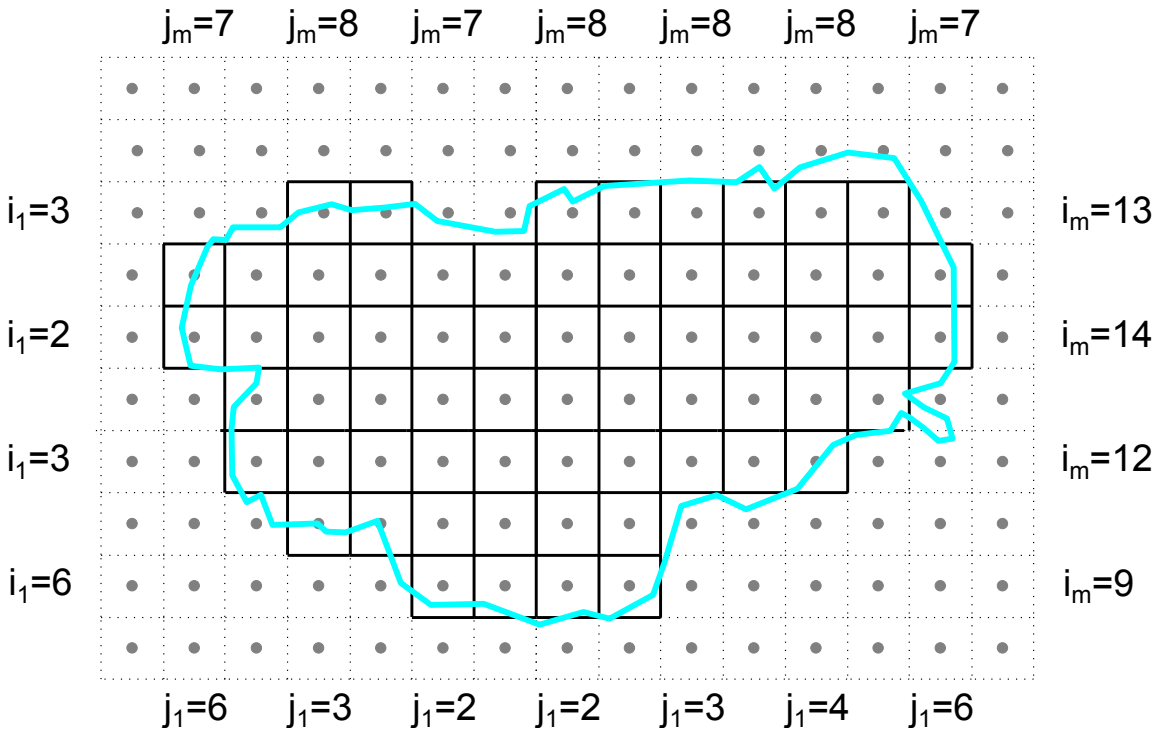

Figure 7.2: *Caption*

Variables are organized in a staggered grid, which means that pressure ($p_{i,j,k}$), density ($\rho_{i,j,k}$), salinity ($s_{i,j,k}$), and temperature ($T_{i,j,k}$), are defined at cell centers, while velocity components are defined in cell faces. The x-component of velocity (u) is defined in the x-normal face ($u_{i+1/2,j,k}$). Similarly, the y and z-components are defined in the y and z-normal faces ($v_{i,j+1/2,k}$, $w_{i,j,k+1/2}$), respectively. The water level elevation is a two-dimensional variable, and it's defined at cell centers as ($\zeta_{i,j}$). The cell height can vary vertically for all cells, and horizontally for the first and last row, and therefore is defined individually for each cell center as $h_{i,j,k}$.

7.2. IMPLICIT-EXPLICIT SCHEME

Physical process	Stability condition	Typical Δ t [s]
Surface wave	$\Delta t \leq \frac{\Delta x}{a}$	400
Internal wave	$\Delta t \leq \frac{\Delta x}{\sqrt{2\epsilon g C}}$	6000
Coriolis (inertial wave)	$\Delta t \leq \frac{1}{f}$	10^4
Horizontal diffusion	$\Delta t \leq \frac{\Delta x^2}{4A}$	2.5×10^6
Vertical diffusion	$\Delta t \leq \frac{\Delta z^2}{4\nu_t}$	100
Horizontal advection	$\Delta t \leq \frac{\Delta x}{u}$	10^4
Vertical advection	$\Delta t \leq \frac{\Delta z}{w}$	4×10^4

Table 7.1: *Caption*

Numerical theory of WQ-AEM

Appendices