User's Manual for

General Ocean Model

(GOM)

Draft v1.0 (based on GOM_vd77)

for

Any users who are interested in numerical modeling.

By

Jungwoo Lee

January 31, 2023

Preface

Why is it so complicated? Is there any possibility I can make it easy to use? These were two questions I had whenever I tried to learn a new numerical model; at least to me, it was not that easy to learn a new model. There is a proverb in Korea, "A thirsty man digs a well," and I decided to develop an easy-to-use model by myself. I proposed a project "Development of the algal bloom forecasting model", which aimed to develop a three-dimensional (3D) hydrodynamic model, right after I joined the Korea Institute of Civil Engineering and Building Technology (KICT) in 2015. I especially would like to thank Dr. Seog-Ku Kim, who was my supervisor at that time, for encouraging me to pursue the project even though most of the other modelers in Korea did not favor developing a new model but favored to use existing models. Even though I had experience with some well-known ocean circulation models, it was not easy to develop a model from the scratch in a few years. Luckily, I had my academic big brother Dr. Jun Lee; I and Dr. Jun Lee both graduated from the University of Florida. Dr. Jun Lee is a true expert in numerical modeling, and he had 3D model development experience. We pulled out his old source codes which were buried in an old USB drive, and we started to develop a new 3D model that aims to be an easy-to-use model.

My first numerical modeling experience goes back to 2003 in my master's degree with the Environmental Fluid Dynamics Code which was developed by Dr. John Hamrick. Without any knowledge of numerical modeling or even without the knowledge of Fortran, I studied how to run the EFDC model by just reading the manual. Since 2003, I have experienced several other well-known ocean circulation models such as the Regional Ocean Modeling System (ROMS), the Finite Volume Community Ocean Model (FVCOM), Advanced Circulation Model (ADCIRC), and, recently, the Semi-Implicit Cross-Scale Hydroscience Integrated System Model (SCHISM). Each model has pros and cons, but overall, I loved the EFDC the most with its simplicity in the compiling process and user-friendly input and output file structures. For this reason, I benchmarked the input file structure of the EFDC when developing this model, the General Ocean Model (GOM). For this reason, users may find great similarities in input files between GOM and EFDC.

When we, I and Dr. Jun Lee, started to code this model, GOM, we aimed to achieve the following three main goals: (1) the model must be easy to use (i.e., user-friendly) even for beginners, (2) the model must be fast enough, and (3) the model must be stable and provide exact solutions. As I mentioned above, the first goal could be achieved by benchmarking the EFDC. To achieve the second and the third goal, we focused on the Casulli and Walters (2000) work. Following Casulli and Walters's (2000) paper, we could achieve the second and third goals. Now, at this point, I believe our model, GOM, fulfills the three proposed goals, thus I am proud of our model.

As one of the most ordinary modelers, I coded this model at the eye level of a common modeler. Thus, our model, GOM, might not have distinctive features as other models, which some genius modelers developed, have. However,

GOM might give users the most general functionalities which may most users ask the models. That is why I gave the name, the General Ocean Model, to our model. Also, the pronunciation of the GOM in Korean means “Bear,” and a bear is smart, strong, and even fast, and that is why I selected the name, GOM.

The journey to today, which I am writing this “preface”, was not that easy. While I was working in the KICT, I decided to move to the USA, and I quit the job in 2018. My first job since I moved to the USA was the St. Johns River Water Management District (SJRWMD). I worked at SJRWMD for eight months, and I loved my job at the SJRWMD, and I would like to thank my supervisor Dr. Peter Sucsy for his kind support. Then, I had to find (for some reason) the next job, and I joined the Texas Water Development Board in 2019. During this time, I had time to work on the GOM only at night and on weekends, since I had to do assigned works at the company. My lovely two sons, Jaden and Jasen, always asked me “Daddy, take a rest, otherwise, you will die.” Well, I have survived sons!

I spent a long time and a lot of effort learning what I now know about numerical methods including programming languages and Linux systems. Most of this learning process was quite painful at least to me. Because I know how painful this learning process is, I focused on the easiness of use when developing the GOM. I hope the GOM helps users to alleviate stresses as which I had when learning a new model, compiling source codes, and digging into some error sources. I can be reached at my email address at jungwoo33@gmail.com. Then, good luck with your journey to numerical modeling!

Jungwoo Lee
At my kids’ room, Austin, Texas
October 26, 2020

Acknowledgments

The initial development of the three-dimensional unstructured grid coastal and estuarine circulation model, which named the General Ocean Model (GOM), started in 2015 supported by funds from the Korea Institute of Civil Engineering and Building Technology (KICT) (project number: 20150121-0001-01), and the first deliverable stage was achieved with the support from the National Research Council of Science & Technology (NST) (No. CAP-18-07-KICT). The GOM source code has been developed led by Dr. Jungwoo Lee at the KICT in collaboration with Dr. Jun Lee at the Korea Coastal Ocean Modeling Systems (KCOMS).

GOM License Agreement

All copyrights to the GOM code are reversed. I, Jungwoo Lee, as a principal developer and investigator, release source codes of the GOM to the public only for research and study purposes. A user who wants to get the GOM source code should contact Dr. Jungwoo Lee personally (jungwoo33@gmail.com). A user should sign the license agreement, which will be sent to a user from Dr. Jungwoo Lee, to get the GOM source codes. Unauthorized reproduction and redistribution of the GOM source code are prohibited.

Table of Contents

List of Figures

List of Tables

1 Introduction

A three-dimensional unstructured grid finite-volume model for coastal and estuarine circulation, which I (Jungwoo Lee) named the General Ocean Model (GOM), has been developed. Combining the finite volume and finite difference methods, GOM achieved both the exact conservation and computational efficiency. The propagation term was implemented by a semi-implicit numerical scheme, so-called θ scheme, and the time-explicit Eulerian-Lagrangian Method (ELM) was used to discretize the nonlinear advection term to remove the major simulation limitations of the time step, which appears when solving shallow water equations, by the Courant-Friedrichs-Lewy stability condition. Because the GOM uses orthogonal unstructured computational grids, allowing both triangular and quadrilateral grids, considerable flexibility to resolve complex coastal boundaries is allowed without any transformation of governing equations. More fundamental details of the GOM can be found in the original development paper (Lee et al., 2020), thus details of the governing equations and the solution algorithms, which used in GOM, are not duplicated in this manual.

The general organization of this manual is as follows. Chapter 2 presents the general structures of the GOM. Chapter 3 describes essential input files including the master input file, *main.inp*, which controls the overall execution of a model simulation. Chapter 4 explains additional input files. Chapter 5 explains how to compile and make an executable file from the source codes. A list of subroutines and a brief description of their functions are presented in Chapter 6.

GOM is originally coded in standard FORTRAN 90 and tested with GNU Fortran (64 bit) both on a Windows 64-bit computer and Linux system.

2 General Structure of the GOM

The current version of the GOM (GOM_v1.0) is comprised of more than one hundred standard FORTRAN 90 source codes including one main program (*main.f90*) and 4 module files. Global variables are defined in *mod_global_variables.f90*, and all of the names of input files and output files are defined in *mod_file_definition.f90*, and users can easily modify the input or output file names and folders in which output files will be stored during the simulation. A general flow structure of the GOM is coded in the main source code, *main.f90*; users can find brief descriptions about source codes in the GOM in Section 6.

In the current version of the GOM (version 1.0), three essential input files are required, and users can specify additional input files for their simulation purpose. I will explain with the following example project folder structure:

- project_folder (this is the name of your project folder)
    - input (all input files are in this folder)
    - output (all output files will be in this folder)
    - run.exe (this is the binary executable generated from the source code folder)

All input files have an extension with '**.inp*', and they should be in the '*input*' folder, which should be located under the same folder executable file (e.g., *run.exe*) locates in. Note that a user can change this folder's location and name easily in *mod_file_definition.f90* as a user's preference. A brief description of input is presented in Table 2.1.

Table 2.1. Input files for the GOM.

Group	File Name	Description
Essential files	main.inp	Master input file.
	cell.inp	Horizontal cell (grid) information file. It includes a total cell number and cell connectivity.
	node.inp	Horizontal node information file. It includes a total node number and x and y coordinates of each node.
Optional input files	eta_ser.inp	Open boundary water surface elevation (η) time series file.
	harmonic_ser.inp	Open boundary tidal (harmonic) elevation time series file.
	q_ser.inp	River discharge (Q) time series file.
	windp_ser.inp	Wind and air pressure time series file.
	hurricane_ser.inp	Wind and air pressure time series at each node (another form of windp_ser.inp.

hurricane_center.inp	Hurricane center information; it is required if hurricane simulation is on.
bottom_roughness.inp	Bottom roughness at each node; use if the spatially varying bottom friction option, **bf_varying**, is selected.
salt.inp	Initial salinity distribution at each node.
salt_ser.inp	Salinity time series file.
temp.inp	Initial temperature distribution at each node.
temp_ser.inp	Temperature time series file.
Obc_info.inp	The open boundary information file
Qbc_info.inp	River boundary information file

Simple test cases without open boundaries can be simulated only with three essential files: *main.inp*, *cell.inp*, and *node.inp*. If open boundaries are forced by harmonic tides (tidal boundary), users should provide *harmonic_ser.inp*, which includes a series of tidal periods, tidal amplitudes, and tidal phases. If open boundaries are forced by the measured surface elevation time series data, the information can be provided with *eta_ser.inp*. If rivers are described in the model grid system, the river discharge time series should be provided with *q_ser.inp*. Wind information can be applied either with *windp_ser.inp* or *hurricane_ser.inp*.

Most output files have an extension with '**.dat*', and they will be automatically located in the 'output' folder, which should be located under the same folder executable file (*run.exe*) locates in; note that there is one exception, and ITPACK2C log file, fort.600, will be stored alongside the executable (this exception will be discussed later). A brief description of output files is presented in Table 2.2.

Table 2.2. Output files for the GOM.

Group	File Name	Description
Check files	run.log	The log file for the model simulation.
	main_mirr.dat	The mirror image file for *main.inp*. It will be automatically generated while reading *main.inp*. So, you must see this file whenever you run a model.
	cell_mirr.dat	The mirror image file for *cell.inp*. It will be generated while reading *cell.inp* if **cell_mirr** is set to 1 in *main.inp*.
	node_mirr.dat	The mirror image file for *node.inp*.

		It will be generated while reading *node.inp* if **node_mirr** is set to 1 in *main.inp*.
	dia_check_geometry.dat	Model grid information file. It will be automatically generated while reading input files. It contains grid related information, so a user can diagnose his/her model grid.
	check_grid_2D.dat (vtk)	2D horizontal model grid file; origin shifted version. It will be generated if **check_grid_2D** is activated.
	check_grid_2DO.dat (vtk)	2D horizontal model grid file; original version. It will be generated if **check_grid_2DO** is activated.
	check_grid_3D.dat (vtk)	3D model grid file. It will be generated if **check_grid_3D** is activated.
	voronoi_center.dat	Voronoi center information at each element. It will be generated if **voronoi** is activated.
	voronoi_face_center.dat	Voronoi face center information. It will be generated if **voronoi** is activated.
	check_eta.dat	Water surface elevation checking file for the given water surface elevation time series (*eta_ser.inp*). It will be generated if **check_eta** is set to 1. It will show the interpolation results every 30 minutes. Not yet implemented.
	check_Q.dat	Given river discharge time series (*q_ser.inp*) checking file. It will be generated if **check_Q** is set to 1. It will show the interpolation results every 30 minutes.
Simulation result files	eta_tser_from_msl.dat (txt)	Water surface elevation time series file. It will be generated if **tser_eta** is on.
	H_tser.dat (txt)	Total water depth time series file. It will be generated if **tser_H** is on.
	u_tser.dat (txt)	True-east velocity time series file. It will be generated if **tser_u** is on.
	v_tser.dat (txt)	True-north velocity time series file. It will be generated if **tser_v** is on.
	salt_tser.dat (txt)	Salinity time series file. It will be generated if **tser_salt** is on.
	temp_tser.dat (txt)	Temperature time series file. It will be generated if **tser_temp** is on.

	airp_tser.dat (txt)	Atmospheric pressure time series file. It will be generated if **tser_pressure** is on.
	tec2D_#.dat vtk2D_#.txt	2D horizontal time series file for selected variables. It will be generated if **IS2D_switch** is on.
	tec3D_full_#.dat vtk3D_full_#.txt	3D horizontal time series file for grid only. It will be generated if **IS3D_full_switch** is on.
	tec3D_surf_#.dat	3D horizontal time series file for the surface only. It will be generated if **IS3D_surf_switch** is on.
	tec2D_flood_map.dat vtk2D_flood_map.txt	2D flood map file. It will be generated if **IS2D_flood_map** is on.
	dump2D_#.dat	2D dump file. It will be generated if **IS2D_dump_switch** is on.
	dump3D_#.dat	3D dump file. It will be generated if **IS3D_dump_switch** is on.
	dia_nonlinear_advection.dat dia_momentum_equation.dat dia_freesurface_equation.dat dia_eta_at_ob.dat dia_bottom_friction.dat dia_face_velocity_uv.dat dia_face_velocity_w.dat dia_node_velocity.dat	Diagnostic files for some important subroutines.

3 Essential Input Files

The GOM requires three essential input files: *cell.inp*, *node.inp*, and *main.inp*. First, users should prepare geometric grid information files, and both unstructured triangular and rectangular grids or combined triangular/rectangular grids can be used in the GOM. Users can use any commercial or non-commercial unstructured grid generation programs. After preparing the two geometric information files, users should provide the master input file, *main.inp*.

All input files can be started with header lines which start either with one of "!", "C", or "c", at the first column of the line; i.e., If a line starts with these characters, the line will be treated as a header line (or comment line), and it will not be read. There is no limit for this, i.e., users can insert or exclude header lines as many as users want, and it depends on the user's preference. I believe this approach gives users much flexibility when handling input files. **Note: If space exists before these comment characters, the line will not be treated as a comment line, and the reading process might be terminated.**

However, there is one important thing users need to keep in mind when dealing with *main.inp*. Even though "C" or "c" which appears in the first column of a line is treated as a comment line, the card number should not be changed, so use them as given; It is because the card number is used as identification when finding each card information. Note that I use the term "Card" when explaining the input file, and I borrowed the term from a "punch card" (see more details here: https://en.wikipedia.org/wiki/Computer_programming_in_the_punched_card_era).

Some card images have names with an underbar, then this card is, mostly, a child card; e.g., C2_1 is a child card of C2. In this case, this child card depends on the parent card, and users must carefully turn on/off this child card. However, some cards are independent cards even though it contains n underbar, e.g., C5_01 ~ C5_04. Users can easily distinguish these independent cards from dependent child cards since these independent card starts with "0" at the right sub-number (i.e., C5_01 not C5_1 in this example).

One more important thing, which may be worthwhile to mention especially for those who are not familiar with Fortran, is the variable name. Most of the variable names in GOM are shown in lower cases since it is my (Jungwoo Lee) preference as the main model developer, however, users should know that Fortran does not distinguish lower or upper cases. Thus, the following variables are identical in Fortran; for example, 'MAXELE', 'Maxele', 'maxele', or even 'maxELE' are the identical variables in GOM.

In the following subsections, I will show an example template figure first, then some explanations will be followed. In most cases, users may be able to get a clear idea by just reading each card image, thus I will only explain something additional only if it requires.

3.1 Grid Files (*cell.inp* & *node.inp*)

```
! cell.inp:
! If you want to add more comments, then put '!' at the first column of the line.
!
! cell_num
! ELEMENT(I), Tri(=3)/Quad(=4), (NODE(I,K),K=1,Tri/Quad)
!
48
1 4 2 1 3 5
2 4 2 5 8 4
3 4 3 6 9 5
. . . . . .
. . . . . .
. . . . . .
46 4 55 59 61 58
47 4 56 60 62 59
48 4 61 59 62 63
```

Figure 3-1. File cell.inp.

The *cell.inp* starts with the total cell number, and rows of the total cell number follow (Figure 3-1). Each row consists of either 5 or 6 columns depending on each element's geometry, i.e., triangular, or quadrilateral element. The 1st column is the element's ID number; the 2nd column is the element's shape type defined as 3 (for triangular) or 4 (for quadrilateral); the 3rd ~ 5th or 6th columns are node numbers that construct the element.

```
!C node.inp:
!C If you want to add more comments, then put '!' at the first column of the line.
!C
!C totla_node_number
!C node_num            x[m]             y[m]         depth[m]
!C
548
1   0.0000000000E+00  0.3000000000E+05  0.100000E+02
2   0.0000000000E+00  0.2800000000E+05  0.100000E+02
3   0.2000000000E+04  0.3000000000E+05  0.100000E+02
.   .                 .                  .
.   .                 .                  .
.   .                 .                  .
546   0.5810000000E+05  0.0000000000E+00  0.100000E+02
547   0.6000000000E+05  0.2000000000E+04  0.100000E+02
548   0.6000000000E+05  0.0000000000E+00  0.100000E+02
```

Figure 3-2. File node.inp.

Similarly, the *node.inp* starts with the total node number, then rows of the total node number follow (Figure 3-2). The 1st column is the node ID number; the 2nd and 3rd columns are x- and y-coordinates in meters, respectively; the 4th line is the water depth (positive for a wet (i.e., water) element and negative for a dry (i.e., land) element).

3.2 GOM Master Input File (main.inp)

The master input file, *main.inp*, consists of several card images. The file, *main.inp* is self-documented with definitions and guidance for each input variable contained in its card image section. Description lines start with an exclamation mark, "!", thus users can freely insert more information lines or delete lines. Description parts in each

card image section include variable names and the explanation of the variable. In addition, a type of variable is shown in square brackets after the name of each variable. For example, [c] represents 'character' type of a variable; [i] represents 'integer' type of a variable; and [r] represents 'real' type of a variable. I think this will help users to understand what kinds of data types are used in GOM.

Note, as I mentioned previously, that there is one more comment method used mostly in *main.inp*. If a line starts with "C" or "c" will be treated as a comment line as well as "!", and one more trick is used here. Each card number must start either with "C" or "c" even though it is treated as a comment line. With this trick, users can insert as many comment lines, or users even can delete all the comment lines except the first card number line. I think this trick provides users a lot of freedom when using this model. Now, let's start taking a look at each card; note the card images used here are from the Mobile Bay test case.

```
!========================================================================!
!
! Main input file for the General Ocean Model (GOM)
!
! Welcome to the GOM model
! Developed by Jungwoo Lee & Jun Lee
! This file is self-documented with definitions and guidance for each
! input variable contained in its card image section.
!
!========================================================================!
! C0 ~ C19: General Model Configuration Information                      !
!========================================================================!
C00   TITLE FOR RUN
!
!     project_name:  [c]   TITLE OR IDENTIFIER FOR THIS INPUT FILE AND RUN.
!                          (LIMIT TO 200 CHARACTERS LENGTH)
!
C00   project_name
      Mobile_Bay:_GOM_v77_synchronized
!========================================================================!
```

Figure 3-3. Card Image C00.

```
!========================================================================!
C01   Define some global parameters before we start
!     Generally, users don't need to change these values.
!
!     gravity              [r]   Gravitational acceleration, g [m/s2]
!     rho_o                [r]   Reference density of seawater [kg/m3]
!     rho_a                [r]   Reference density of air [kg/m3] @ 0 degree Celsius
!     max_no_neighbor_node [i]   Maximum allowed number of neighbor nodes and elements in your grid
!                                Typically, 10 is o.k. for general grid.
!
C01   gravity     rho_o    rho_a    max_no_neighbor_node
      9.81        1025.0   1.29     10
!========================================================================!
```

Figure 3-4. Card Image C01.

```
!========================================================================!
C1    Read coordinate frame flag and check it
!
!     maxnod             [i]   Number of nodes in your grid
!     maxele             [i]   Number of elements in your grid
!     ana_depth          [i]   0/1   off/on, overwrite water depth with analytical initial conditions
!                                    If 1, re-define water depth in "ana_depth.f90".
!                                    In this case, water depth defined in node.inp will be replaced with the given analytic informa
!     coordinate_system [i]    1  Cartesian or UTM system
!                              2  Longitude/Latitude system. Lon. and lat. should be in [degree]
!     voronoi            [i]   0/1   off/on, use voronoi center of elements, otherwise Centroid of a triangle will be used.
!     node_mirr:         [i]   0/1   off/on, Write mirror image file (node_mirr.dat)
!     cell_mirr:         [i]   0/1   off/on, Write mirror image file (cell_mirr.dat)
!
C1    maxnod   maxele   ana_depth   coordinate_system    voronoi  node_mirr   cell_mirr
      18718    35127    0           1                   0        1           1
!========================================================================!
```

Figure 3-5. Card Image C1.

A user can identify the project title in Card Image C00 (Figure 3-3), and Card Image C01 (Figure 3-4) includes some basic global variables. Card Image C1 (Figure 3-5) consists of a grid and coordinate related variables. The **maxnod** and **maxele** are the total node and element numbers in the grid, respectively, and these numbers must be identical to the total node number and element number provided in *node.inp* and *cell.inp*, respectively. If the **ana_depth** is on, then users can define water depth analytically, overwriting the provided depth information in **node.inp**, in the *ana_initial.f90*, then users must compile source codes again and build a new executable. This option is specifically useful when users want to test the water depth effect.

The **Voronoi** is the off/on option for the Voronoi center calculation. GOM uses the unstructured orthogonal grid,

and it requires some restrictions when generating grids. Thus, GOM allows using a less restrictive geographical limit as using the centroid of an element rather than using a strict Voronoi center. Thus, users can try to turn on/off this **Voronoi** option with the users' grid. If GOM fails with the **Voronoi** option, turn off this option. When using a numerical model, most of the errors are associated with the main input files, and sometimes it is quite difficult to find the source of the error. For this reason, GOM generates mirror image files when reading *main.inp*, *node.inp*, and *cell.inp*, to make users easy to check the input error. The main input file's mirror image file, *main_mirr.dat* will be generated automatically, but the other two files have a turning on/off option.

```
!=========================================================================!
C2    Read vertical layer information
!
!     maxlayer          [i]   Number of vertical layers
!     MSL               [r]   Mean Sea Level elevation from the bottom (z_level(0)) [m]
!     z_level(0)        [r]   Reference bottom level elevation [m]
!
C2    maxlayer    MSL      z_level(0)
      5           0.0      -100.0
!=========================================================================!
C2_1  [maxlayer] lines are required
!
!     serial_num  [i]   Layer number (dummy number)
!     delta_z     [r]   dz [m] of each vertical grid
!     z_level     [r]   each vertical grid level [m] (each grid's top elevation) from the bottom (i.e., z_level(0))
!                       Provide this in surface-to-bottom order.
!
C2_1  serial_num  delta_z(i)  z_level(i)
       5          23.0        20.0     ! 5 layers
       4          3.0         -3.0
       3          3.0         -6.0
       2          3.0         -9.0
       1         88.0         -12.0
!=========================================================================!
```

Figure 3-6. Card Image C2 & C2_1.

Card Images C2 and C2_1 (Figure 3-6) are about the vertical layer information. The current version of GOM uses the z-grid system, and the **maxlayer** defines the number of vertical z layers; the vertical layer number starts from the bottom. There is another variable that defines the z-grid system in GOM, **z_level**. Each vertical layer has top and bottom faces, and they are defined as, assuming the *k*th vertical layer, **z_level**(k) for the top face, and **z_level**(k-1) for the bottom face (Figure 3-7). The **MSL** is the mean sea level elevation from the arbitrary bottom level (**z_level**(0)), and the **MSL** must be located above the following **z_level**(0). For example, let's assume the maximum depth in the model grid is 50.0 m, and users want to set the mean sea level at z = 0.0 m. Then, **z_level**(0) must be equal to or less than -50.0 m; i.e., **z_level**(0) can be any number below -50.0 m (in this example, I used z_level(0) = -100.0 m). In other words, if users want to set the arbitrary bottom at z = 0.0 m, then the **MSL** can be located at any elevation greater than z = 50.0 m. So, it depends on the users' preference. Figure 3-7 shows details of the z-grid system used in GOM. As mentioned previously, the example shown in Figure 3-6 includes extra comment lines for the "1-layer" example. Thus, the "1-layer" information will be automatically commented out, and GOM will run successfully.

The C2_1 is a child card of the C2, and it requires **maxlayer** lines. Users must provide vertical layer information from the surface to the bottom order (e.g., as shown in Figure 3-6). The **serial_num** is the vertical layer number, and it starts from the surface layer. Thus, for example, if the **maxlayer** is 5, this variable starts from 5 to 1. However, it is a dummy number, so GOM will not use this number as given but use it as its order. The **delta_z** is a vertical layer space in [m], which is defined as **delta_z**(k) = **z_level**(k) - **z_level**(k-1), where **z_level**(k) and **z_level**(k-1) are the vertical locations of the top and bottom faces of the *k*th layer, respectively (see Figure 3-7). One important thing that users must know is the surface layer's top-level, i.e., **z_level**(maxlayer), should be defined sufficiently above the **MSL** to capture the flooding event. If the top-level is defined close to the **MSL**, and if the predicted (simulated) water surface elevation is greater than the given top-level, GOM will be terminated.

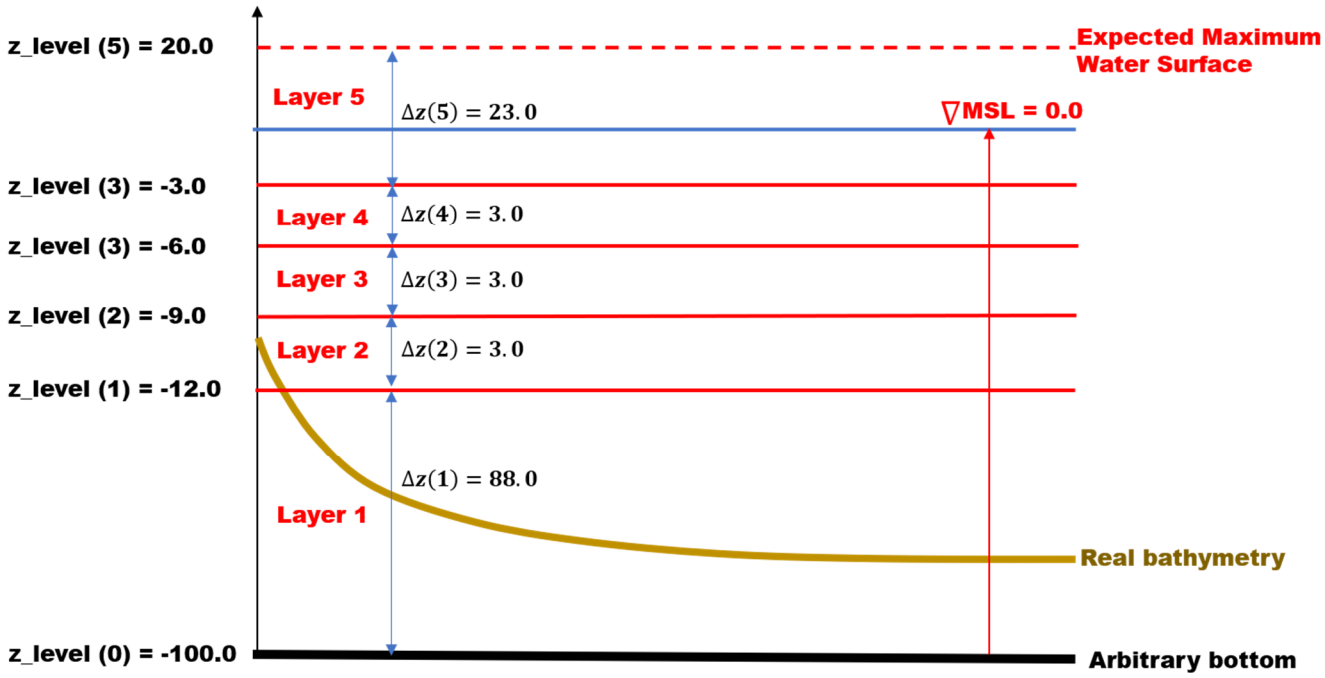

Figure 3-7. Schematic diagram of the z-grid system used in C2 & C2_1.

```
!========================================================================!
C3    Read time-related variables
!
!     dt                [r]   Time increment in [sec]
!     ndt               [i]   Total simulation step
!                             Total simulation time = dt * ndt [sec]
!     data_start_year   [i]   All given input data's start (reference) year (except hurricane_ser.inp)
!                             All data starts at data_start_year/January/1st 00:00:00
!     data_time_shift   [i]   Data start time shift in [day] from data_start_year/January/1st 00:00:00
!     start_year        [i]   Simulation start year. Should be 1900 <= start_year <= 2099
!     start_month       [i]   Simulation start month
!     start_day         [i]   Simulation start day
!     start_hour        [i]   Simulation start hour
!     start_minute      [i]   Simulation start minute
!     start_second      [i]   Simulation start second
!
C3    dt       ndt   data_start_year   data_time_shift   start_year  start_month start_day   start_hour  start_minute   start_secc
      600.0    8784  2010              212               2010        8           1           0           0              0
!========================================================================!
```

Figure 3-8. Card Image C3.

Card Image C3 (Figure 3-8) consists of a line with simulation time-related variables. It might be a bit confusing for a new user, however, this approach will give users much more flexibility when developing a model input data. Once model input data is generated, users can simulate GOM from any data time range. For example, if the provided input files (such as boundary elevation and river discharge data) start from January 1st, 2000 and a user wants to simulate from March 15, 2000, a user does not need to correct (or change) model input data but just run the model from March 15, 2000, with the following settings:

- **data_start_year** = 2000
- **data_time_shift** = 0
- **start_year** = 2000
- **start_month** = 3
- **start_day** = 15
- **start_hour**, **start_minute**, **start_second** = 0

It is also easy to accumulate model input data for a long period, and users just need to add more data at the end of the existing input file. The **data_time_shift** is also useful when there is a mistake in the generated input files; users can easily shift the data beginning time. Note that there is one more time shift option in each time series input file, and it will be explained later. The example in Figure 3-8 shows that the model simulation starts from 08/01/2010, and the provided data accumulated also from 08/01/2010 00:00:00 setting **data_start_year** = 2010 and **data_time_shift** = 212.

```
!========================================================================!
C4    Restart & screen show control
!
!     restart_in:       [i]   0  Cold start: normal case
!                             1  Hot start:  'restart.inp' is required.
!     restart_out:      [i]   0  Do not write restart file (restart_***.out)
!                            -1  Write restart file (restart_***.out) once at the end of run
!                             1  Write restart file at every restart_freq
!     restart_freq:     [i]   N  Write restart_***.out every restart_freq period (N * dt seconds)
!     ishow:            [i]   0  Do not show run-time results on screen
!                             1  Show run-time results on screen (full show)
!                             2  Show only beginning & ending on screen (skip showing run-time results)
!     ishow_frequency:  [i]   N  Screen show time interval (N * dt seconds)
!
C4    restart_in  restart_out restart_freq   ishow    ishow_frequency
      1           -1          4320           1        6                     ! every hr; dt = 600, every 30 day
!========================================================================!
```

Figure 3-9. Card Image C4.

Card Image C4 (Figure 3-9) consists of restart options and model simulation progress screen show control options. In this example, it shows that **restart_out** = -1, and that indicates:

- a user wants to generate restart file (*restart_***.out*) once at the end of the run; i.e., only one restart file will be generated at the last time step of the simulation (i.e., **ndt** in C3 or **dt** * **ndt** [s] = 600 * 8784 [s] = 61 [days]). Thus, GOM will produce only one restart file, *restart_0008784.out*

If a user select **restart_out** = 1 and **restart_freq** = 1440, for example, and that indicates:

- a user wants to generate restart file (*restart_***.out*) every 1440 time steps; i.e., restart files will be generated every 10 days of simulation (**dt** * **restart_freq** [s] = 600 * 1440 [s] = 10 [days]). Thus, GOM will produce *restart_0001440.out*, *restart_0002880.out*, …

Then, a user can use one of the restart files as an initial condition setting **restart_in** = 1 (note that a user should copy the restart file from *./output* to *./input* and change the file name to *restart.inp*) if a user wants to restart the model simulation from a sudden time.

```
!========================================================================!
C5_01 Momentum equation variables I - Propagation/Nonlinear advection
!     Note: The ELM method is used for the (1) nonlinear advection and (2) Coriolis terms
!
!     theta                [r]   Implicitness paramter for the propagation term. 0.5 <= theta <= 1.0
!     advection_flag       [i]   0/1   off/on, Turn off/on nonlinear advection: absolute criteria
!     adve_onoff_depth     [r]   [m]   Turn off nonlinear advection below this depth: conditional criteria
!                                      This is the additional off/on option for nonlinear advection
!     ELM_backtrace_flag   [i]   0  Constant ELM subdivisions, [ELM_sub_iter], will be used.
!                                1  The number of subdivisions, [ELM_sub_iter], will be automatically calculated in the code
!                                   based on the local velocity gradient.
!                                2~ Other integer numbers will make ELM unconditionally stable, but accuracy is not guaranteed.
!     ELM_sub_iter         [i]      Number of sub iteration for ELM
!                                   If ELM_backtrace_flag is off (==0), [ELM_sub_iter] will be used.
!     ELM_min_iter         [i]      Number of minimum iteration for ELM. When ELM_backtrace_flag == 1
!     ELM_max_iter         [i]      Number of maximum iteration for ELM. When ELM_backtrace_flag == 1
!                                   If ELM_backtrace_flag is on (==1), [ELM_sub_iter] will be calculated in the range:
!                                   ELM_min_iter < ELM_sub_iter < ELM_max_iter
!
C5_01 theta    advection_flag    adv_onoff_depth   ELM_backtrace_flag   ELM_sub_iter   ELM_min_iter   ELM_max_iter
      0.6      1                 0.01              1                    1              1              100
!========================================================================!
C5_02 Momentum equation variables II - Bottom friction
!     bf_flag           [i]   Bottom friction on/off option
!                             0 = No bottom fricition
!                             1 = Chezy-Manning equation
!                             2 = Quadratic friction law (log law)
!     bf_varying        [i]   Spatially varying or constant bottom friction?
!                             0 = spatially constant bottom friction; the constant 'manning' or 'bf_height' will be used.
!                             1 = spatially varying bottom friction; bottom_roughness.inp is required.
!     manning           [r]   Manning's n (for Chezy-Manning equation),
!                             This value will be used if bf_flag == 1 & bf_varying == 0
!     bf_height         [r]   Bottom roughness height, z0 (for log law) in [m],
!                             This value will be used if bf_flag == 2 & bf_varying == 0
!     von_Karman        [r]   von Karman constant (for log law); typical value = 0.41
!
C5_02 bf_flag  bf_varying  manning  bf_height   von_Karman
      1        0           0.018    0.00001      0.41
!========================================================================!
C5_03 Momentum equation variables III - Baroclinic gradient/wetting & drying/Coriolis
!
!     baroclinic_flag   [i]   0/1   off/on, baroclinic simulation
!                             0     Baroclinic density gradient effect will be ignored.
!                             1     Baroclinic density gradient effect will be included.
!     ana_density       [i]   0     analytical density is off; the full equation of state will be used.
!                             1     Use a simple linear equation of state (rho = 1000.0 + 0.7*salinity)
!                             2     Use this option for test (user must specify own equation in calculate_analytical_density.f90).
!     ref_salt          [r]   Reference salinity in [psu].
!                             If salinity transport is off (i.e., is_tran(1) == 0), this value will be used for the density calcul
!     ref_temp          [r]   Reference water temperature in [Celcius].
!                             If temperature transport is off (i.e., is_tran(2) == 0), this value will be used for the density cal
!     dry_depth         [r]   [m]   Should be greater than 0.0 [m] (0.01 m is recommended).
!                                   If depth <= dry_depth, it will be a dry element.
!     Coriolis_option   [i]   0  No Coriolis
!                             1  Constant Coriolis from given Coriolis (f-plane approximation)
!                                f = 2*omega*sin(mid_latitude); GOM will use f = Coriolis.
!                             2  Variable Coriolis (by beta-plane approximation); [coordinate_system] in C1 must be 2 (i.e., lon/l
!                                f = f0 + beta*y; GOM will calculate f at each face center.
!     Coriolis          [r]   Coriolis parameter (for f-plane approximation)
!
C5_03 baroclinic_flag   ana_density ref_salt ref_temp dry_depth   Coriolis_option   Coriolis
      1                 0           30.0     20.0     0.01        1                0.00003646  ! 30 degree
!========================================================================!
C5_04 Momentum equation variables IV - Horizontal & Vertical diffusion
!     In the case of momentum diffusion, they are called 'eddy viscosities'.
!     For scalar fields such as temperature or salinity, the coefficients are called 'eddy diffusivities'. They are on the order o
!
!     Ah_0                    [r]   Background horizontal eddy viscosity [m2/s]; for momentum diffusion.
!                                   This value will be added to the Smagorinsky model results.
!     Kh_0                    [r]   Background horizontal eddy diffusivity (Kh); for other diffusion.
!                                   This value will be added to the Smagorinsky model results.
!                                   Kh varies from near zero to order 10^4 [m2/s] (Cole et al, 2015); typically 10^2 ~ 10^4 [m2/s]
!     Smagorinsky_parameter   [r]   Smagorinsky parameter in Smagorinsky-Lilly model.
!                                   Usually it has the value: 0.0 ~ 0.25. See more details in 'solve_momentum_equation.f90'.
!                                   If this is set to 0.0, then the Smagorinsky-Lilly model will be turned off.
!     Av_0                    [r]   Background vertical eddy visocity [m2/s]; for momentum diffusion.
!                                   typical value = 1.e-2 [m2/sec]
!     Kv_0                    [r]   Background vertical eddy diffusivity [m2/s]; for other diffusion.
!                                   typical value = 1.e-4 [m2/sec]
C5_04 Ah_0  Kh_0  Smagorinsky_parameter   Av_0     Kv_0
      0.00  0.00  0.021                   1.e-2    1.e-4
!========================================================================!
```

Figure 3-10. Card Image C5_01 ~ C5_04.

Card Images C5_01 through C5_04 are the momentum equation-related variables, and each chard, unlike other sub cards, is independent of each other as previously described. The GOM is stable when the implicitness factor, θ, is in the range of $0.5 \leq \theta \leq 1.0$. The **advection_flag** is an absolute criterion to turn off/on the nonlinear advection term while the following **adv_onoff_depth** is a conditional criterion.

Users can either turn off or on **baroclinic_flag** while **transport_flag** in C7 is on. If **baroclinic_flag** is off while **transport_flag** is on, it means the transportive quantity (e.g., salinity or temperature) is treated as a tracer, and it doesn't affect the density flow. If, however, **baroclinic_flag** is on while **transport_flag** is on, it means the concentration of the salinity or temperature contributes to the water flow affecting water density.

GOM automatically checks whether an element is dry or wet comparing each element's depth with **dry_depth**. If the depth of an element is less than the given **dry_depth**, the element will be treated as a dry cell. Only elements greater than **dry_depth** will be treated as a wet element, and the momentum equation process will be conducted. Thus, this value must be greater than 0.0 m (the typical value is 0.01 m).

In GOM, a variable horizontal eddy viscosity (A_h) is calculated using the Smagorinsky subgrid-scale scheme (it is also known as the Smagorinsky-Lilly model) (Smagorinsky, 1963), which can be generally written in 2D Cartesian coordinates as (Ji, 2017):

$$A_h = C_s \Delta x \Delta y \left[\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 + \frac{1}{2} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)^2 \right]^{1/2} \tag{3-1}$$

where C_s is the Smagorinsky constant (and it is defined with the **smagorinsky_parameter** in C5_04), Δx and Δy are the model grid sizes in x and y direction, respectively. The Smagorinsky constant has typical values between 0.1 and 0.2. Then, a user can add additional values on top of the calculated eddy viscosity using the **Ah_0** in C5_04. Thus, the final eddy viscosity will be the sum of **Ah_0** and A_h which calculated from Equation (3-1).

```
!========================================================================!
C6    Select Matrix solver:
!     GOM provides two different matrix solvers, so select one of them.
!
!     matrix_solver_select [i]   1  Preconditioned Conjugate Gradient Method
!                                   Use this package with serial/OpenMP/MPI compiled executable.
!                                   If chosen, activate C6_1.
!                                2  PETSc:
!                                   Use this package with paralled compiled executable (with MPI).
!                                   If chosen, activate C6_2. Not yet implemented.
C6    matrix_solver_select
      1
!========================================================================!
C6_1  Jacobi Preconditioned Conjugate Gradient (PCG) method:
!     Activate if matrix_solver_select == 1
!
!     max_iteration_pcg [i]      Maximum number of iteration
!     error_tolerance   [r]      The stopping criterion or approximate relative accuracy desired in the final computed solution.
!     pcg_result_show   [i]      0/1   off/on, PCG solver results show
!                                If it is on, GOM will show the converging iteration number both on the screen and the log file, r
C6_1  max_iteration_pcg    error_tolerance   pcg_result_show
      1000                 5.e-6             0
!========================================================================!
C6_2  PETSc information:
!     Activate if matrix_solver_select == 2
!     Not yet included
!========================================================================!
```

Figure 3-11. Card Image C6.

Card Images C6 family (C6 through C6_2) provide a selection of matrix solvers used in GOM. Users can choose one of the two sparse matrix solvers used in GOM: (1) Preconditioned Conjugate Gradient Method with OpenMp written by myself, and PETSc. The PETSs is a sparse matrix solver package that comes with MPI; this is not yet implemented.

```
!========================================================================!
C7    off/on tranport equations & selection of the solver
!
!     transport_flag    [i]   0/1   off/on, global turn off/on flag for transport equation
!                               If transport_flag == 0, following variables will be automatically turned off.
!     transport_solver  [i]   1/2   Transport equation solver selection option
!                               1     TVD scheme
!                               2     Eulerian Lagrangian Method (ELM)
!
C7    transport_flag    transport_solver
      1                 2
!========================================================================!
C7_1  TVD solver
!     Activate if transport_flag == 1 & transport_solver == 1 in C7
!
!     trans_sub_iter    [i]   N     Number of sub iteration (sub-cycling) for the TVD transport eqaution
!                               0,1   No sub-iteration
!                               2~N   Number of sub-iterations
!     h_flux_limiter    [i]   0~3   Flux limiter functions for horizontal:
!                               0: No flux limiter (i.e., 1st-order upwind), 1: Minmod, 2: van Leer, 3: Superbee
!     v_flux_limiter    [i]   0~3   Flux limiter functions for vertical:
!                               0: No flux limiter (i.e., 1st-order upwind), 1: Minmod, 2: van Leer, 3: Superbee
!     rjk_option        [i]         Ratio of consecutive gradient, r, calculation method in the flux limite function
!                               1     Casulli's approach
!                               2     SCHISM's approach
C7_1  trans_sub_iter    h_flux_limiter    v_flux_limiter    rjk_option
!     30                3                 1                 1
!========================================================================!
C7_2  Turn off/on transport variables
!     Activate if transport_flag == 1
!
!     is_tran  [i]   0/1   off/on, transported scalar materials; note, currently GOM has two materials; it will be expanded step b
!                          is_tran[1]  = salinity
!                          is_tran[2]  = temperature
!     num_tran_ser   [i]   Total number of time series for each tranported material
!                          If > 0, should have '***_ser.inp'
!     tran_ser_shape [i]   There are two shapes of '***_ser.inp' depending on a user's preference
!                          1 = ***_ser1.inp (useful if each station has different time record)
!                          2 = ***_ser2.inp (useful if each station has identical time record)
!
C7_2  is_tran  num_tran_ser   tran_ser_shape
      1        3              1              ! salinity; salt_ser.inp
      0        3              1              ! temperature; temp_ser.inp
!========================================================================!
```

Figure 3-12. Card Image C7.

Card Images C7 family (C7 through C7_2) are related to the transport model. Users can either turn off or on the transport equation using the **transport_flag**; i.e., the barotropic or baroclinic simulation will be set by turning off or on **transport_flag** (i.e., 0 or 1), respectively. GOM provides two transport equation solving algorithms: (1) A high-resolution transport method with the 2nd-order semi-implicit Total Variation Diminishing (TVD) scheme with flux limiter functions, and (2) Eulerian Lagrangian Method (ELM); a user can select each algorithm with **transport_solver**. When a user selects the TVD scheme, a user can choose one of four transport models: (0) 1st-order upwind method, (1) Minmod limiter, (2) van Leer limiter, and (3) Superbee limiter. The stability criterion of this transport model is (Casulli and Zanolli, 2005):

$$\Delta t \leq \frac{P_i \Delta z_{i,k}^n}{2\sum_{j\in S_i^+}\left|Q_{j,k}^{n+\theta}\right| + 2\left|Q_{i,k+1/2}^{n+\theta}\right| + \sum_{j\in S_i^+\cup S_i^-} d_{j,k}^{n+\theta}} \tag{3-2}$$

However, sometimes this criterion is too severe comparing to the hydrodynamic model and this transport model part becomes a bottleneck. In this case, GOM allows users to use the sub-cycling (or sub-iteration) approach by

setting **trans_sub_iter** ≥ 2. If sub-cycling is used, the transport model will have a smaller time step than the hydrodynamic model (i.e., $dt_{transport} = dt/trans\_sub\_iter$), and GOM will become stable.

The current version of the GOM only has Salinity and Temperature (not yet implemented) transport models. If **transport_flag** is on, users must turn on C7_1, otherwise, turn off C7_1 either by deleting the line or putting "!" at the beginning of the line. Note that 0 or 1 for **trans_sub_iter** have the same effect since 0 means no sub-iteration, and 1 means 1 sub-iteration. If ELM is selected (i.e., **transport_solver = 2**), C7_1 must be turned off.

```
!=============================================================================!
C8    Model termination control
!
!     terminate_check_freq:   [i]   N  Check model termination criteria at every terminate_freq period (N * dt seconds)
!     eta_min_terminate:      [r]   Model termination criterion by minimum water surface elevation [m]
!     eta_max_terminate:      [r]   Model termination criterion by maximum water surface elevation [m]
!     uv_terminate:           [r]   Model termination criterion by maximum horizontal velocities [m/s]
!     salt_terminate:         [r]   Model termination criterion by maximum salinity [ppt]
!     temp_terminate:         [r]   Model termination criterion by maximum water temperature [Celcius]
!
C8    terminate_check_freq    eta_min_terminate    eta_max_terminate    uv_terminate   salt_terminate    temp_terminate
      9999999                 -3.0                 3.0                  3.0            40.0              40.0
!=============================================================================!
! The end of the first set.                                          !
! C20 ~ C29, Tidal & River Boundary Conditions                            !
'=============================================================================='
```

Figure 3-13. Card Image C8.

Card Image C8 consists of model termination control variables. Sometimes, the model may produce unrealistic results, but it keeps running. Therefore users may realize (or find) the faulty simulation at the end of the simulation. To avoid this effort, GOM provides these options. Using this card information, users can easily detect possible errors and terminate the model simulation. Card Image C8 is the end of the first set.

```
!========================================================================!
C20   Open boundary (tidal eta or tidal flow) information
!
!     num_ob_cell:      [i]   Total number of open boundary cells (elements) - for eta only
!                             [num_ob_cell] = [num_tide_bc] + [num_eta_bc]
!                             If > 0, either activate C20_1 or provide open boundary information in ob_info.inp
!                             If == 0, following variables will be ignored.
!     ob_info_option:   [i]   How do you want to provide the open boundary (eta) information?
!                             0  In the following sub-card; following sub-card must activated
!                             1  Provide the open boundary (eta) information in the additional ob_info.inp;
!                                if this is selected, do not activate the following sub-card.
!     num_tide_bc:      [i]   Number of tidal boundary (periodic forcing using tidal constitutents)
!     num_harmonic_ser: [i]   Total number of harmonic series
!                             If > 0, should have 'harmonic_ser.inp'
!     check_tide:       [i]   1 = Write harmonic tide checking file: check_tide.dat
!                                 Harmonic tide at each station: every 1 hour.
!                                 Not yet activated
!     num_eta_bc:       [i]   Number of surface elevation boundary (eta given, general purpose)
!     num_eta_ser:      [i]   Total number of eta time series
!                             If > 0, should have 'eta_ser.inp'
!     check_eta:        [i]   1 = Write eta series checking file: check_eta.dat
!                                 Interpolation result: default = every 30 minutes.
!                                 Not yet activated
!     eta_ser_shape:    [i]   There are two shapes of 'eta_ser.inp' depending on a user's preference
!                             1 = eta_ser1.inp (useful if each station has different time record)
!                             2 = eta_ser2.inp (useful if each station has identical time record)
C20
C20   num_ob_cell       ob_info_option
C20   num_tide_bc       num_harmonic_ser     check_tide
C20   num_eta_bc        num_eta_ser          check_eta      eta_ser_shape
      75                0
      0                 0                    0
      75                2                    0              2
!========================================================================!
C20_1 Eta boundary information
!     Activate if num_ob_cell > 0 .and. ob_info_option == 0.
!     Should have 'num_ob_cell' information (lines).
!
!     serial_num        [i]   Station number (dummy number)
!     ob_nodes(i,1)     [i]   First node of the station (or the element, in counterclockwise direction)
!     ob_nodes(i,2)     [i]   Second node of the station (or the element, in counterclockwise direction)
!     ob_eta_type       [i]   Open boundary surface elevation type:
!                            -1  Radiation bc
!                                No input is required; elevations will be computed as average of surrounding elevations.
!                             1  This boundary is forced by time history of elevation. eta_ser.inp is required
!                             2  This boundary is forced by tidal harmonic constituents; harmonic_ser.inp is required; default =
!     harmonic_ser_id   [i]   ID number of harmonic series at this station.
!                                If ob_eta_type == 2 or 3, it will be activated.
!                                It should be greater than 0.
!     eta_ser_id        [i]   ID number of surface elevation time series at this station.
!     salt_ser_id       [i]   ID number of salinity time series at this station.
!     temp_ser_id       [i]   ID number of temperature time series at this station.
!     ob_name           [c]   dummy variable. This will not be read.
!
C20_1 serial_num  ob_nodes(i,1)  ob_nodes(i,2)  ob_eta_type(i) harmonic_ser_id(:)   eta_ser_id(:)  salt_ser_id    temp_ser_id
      1           18718          18716          1              0                   1              1              1
      2           18716          18713          1              0                   1              1              1
      .              .              .           .              .                   .              .              .
      .              .              .           .              .                   .              .              .
      .              .              .           .              .                   .              .              .
     74           12402          12255          1              0                   2              2              2
     75           12255          12102          1              0                   2              2              2
!========================================================================!
```

Figure 3-14. Card Images C20 & C20_1.

Notice that the second card sets start from C20 not from C9, and there is a reason why I set this; this gap is room for possible modification in the future. Through Card Images C20 and C20_1 a user specifies tidal open boundaries. Users can set the tidal open boundary either by (1) using harmonic tidal constituents (setting **num_tide_bc** > 0, this requires *harmonic_ser.inp*) or by (2) providing measured water surface elevation data (setting **num_eta_bc** > 0, this requires *eta_ser.inp*). Thus, the total tidal open boundary elements (**num_ob_cell**) must equal the sum of **num_tide_bc** and **num_eta_bc**. When providing each open boundary element's node information in C20_1, the

first (**ob_nodes(i,1)**) and the second (**ob_nodes(i,2)**) nodes must be provided in the counterclockwise order in an element, otherwise, GOM will not be able to find the corresponding element. Each open boundary element can have boundary salinity and temperature, and they are provided by setting **salt_ser_id** and **temp_ser_id**. There is one more useful option **ob_info_option**. When **ob_info_option** is activated, users can provide the following sub_card C20_1 in a separate input file *ob_info.inp*, which has the exact form of C20_1. This option is quite useful when the open boundary consists of many elements; similarly, there is **Qbc_info_option** in C22.

```
!=========================================================================!
C21   Setup spinup function for tide
!
!     tide_spinup               [i]   0/1   off/on, spinup for tide
!     tide_spinup_period        [r]   [day]
!     baroclinic_spinup         [i]   0/1   off/on, spinup for baroclinic
!     baroclinic_spinup_period  [r]   [day]
!
C21   tide_spinup          tide_spinup_period
C21   baroclinic_spinup    baroclinic_spinup_period
      0                    0.5                    ! tide_spinup
      0                    0.5                    ! baroclinic_spinup
!=========================================================================!
```

Figure 3-15. Card Image C21.

```
!=========================================================================!
C22   Include river boundary
!
!     num_Q_bc:          [i]   Number of river or point source boundaries (Not node number but cell number) (Q given)
!                              If > 0, should activate the following sub-card & need 'q_ser.inp'
!     Qbc_info_option:   [i]   How do you want to provide the river boundary information?
!                              0  In the following sub-card; following sub-card must activated
!                              1  Provide the river boundary information in the additional Qbc_info.inp;
!                                 if this is selected, do not activate the following sub-card.
!     num_Q_ser:         [i]   Total number of Q time series
!     check_Q:           [i]   1 = Write Q series checking file: check_Q.dat
!                              Interpolation result: default = every 30 minutes.
!     Q_ser_shape:       [i]   There are two shapes of 'q_ser.inp' depending on a user's preference
!                              1 = q_ser1.inp (useful if each station has different time record)
!                              2 = q_ser2.inp (useful if each station has identical time record)
!
C22   num_Q_bc     Qbc_info_option   num_Q_ser      check_Q     Q_ser_shape
      1            0                 1             0           1
!=========================================================================!
C22_1 River or point source boundary (Q given) information
!     Activate if num_Q_bc > 0. num_Q_bc lines are required.
!
!     serial_num:     [i]   Station number (dummy number)
!     Q_nodes(i,1):   [i]   First node of the station (or the element, in counterclockwise direction)
!     Q_nodes(i,2):   [i]   Second node of the station (or the element, in counterclockwise direction)
!     Q_ser_id:       [i]   ID number of river (Q) time series
!     Q_portion:      [r]   Portion of Q for Q_ser_id
!     Q_salt_ser_id   [i]   ID number of salinity time series at this station.
!     Q_temp_ser_id   [i]   ID number of temperature time series at this station.
!     Q_name:         [c]   River or point source ID name (dummy variable)
!
C22_1 serial_num  Q_nodes(i,1)   Q_nodes(i,2)   Q_ser_id(:)    Q_portion   Q_salt_ser_id  Q_temp_ser_id  !Q_name
      1           151            141            1              2.5         3              3              ! Mobile River (cell# = 1
!=========================================================================!
```

Figure 3-16. Card Image C22 & C22_1.

```
!========================================================================!
C23   Include earth equilibrium tidal potential (or Solid Earth Tide)
!
!     no_Etide_species     [i]   Number of Earth equilibrium tidal potential species
!                                [no_Etide_species] should be less than [max_no_tidal_constituent]
!
!     Etide_cutoff_depth   [r]   [m], if the water depth is less than this, earth tide will not be added.
!                                See more details in [solve_momentum_equation.f90]
!
C23   no_Etide_species  Etide_cutoff_depth
      0                 40.0
!========================================================================!
C23_1 Activate if [no_Etide_species] > 0
!     [no_Etide_species] lines are required
!
!     Etide_species           [i]   1,2,3,...
!     Etide_amplitude         [r]   [m]
!     Etide_frequency         [r]   [0,1,2,...,n times]/day, or cycle per day (cpd)
!     Etide_nodal_factor      [r]   see more details in any reference
!     Etide_astro_arg_degree  [r]   [degree]
!
C23_1 Etide_species(i)  Etide_amplitude(i)   Etide_frequency(i)   Etide_nodal_factor(i)   Etide_astro_arg_degree(i)
!     0                 0                    0                    0                       0
!========================================================================!
! C23_1 is the end of the second set.                                    !
! C30 ~ C39, Air/Sea Boundary Conditions                                 !
!========================================================================!
```

Figure 3-17. Card Image C23 & C23_1.

Note that C23_1 is the end of the second card set.

```
!========================================================================!
C30   Wind & pressure data type #1: using simple measured data from wind station
!     General wind information from wind stations (not storm type winds)
!
!     wind_flag            [i]   0  There is no wind station data.
!                                1  Wind information in windp_ser.inp will be used.
!                                2  Analytical wind stress will be used in [ana_windp.f90].
!     airp_flag            [i]   0  There is no air pressure station data.
!                                1  Air pressure information in windp_ser.inp will be used.
!     num_windp_ser        [i]   Total number of wind and pressure series (stations) in windp_ser.inp
!                                If one of wind_flag or pressure_flag == 1, this variable will be activated.
!                                Currently, maximum 3 windp stations are allowed.
!                                If more than 3 stations, consider to use wind model data as in C32.
!     wind_formular        [i]   Select one of the following wind stress formulations:
!                                Default formular: Smith's (1980) equation (when you choose other than 1 ~ 3)
!                                1 = Garratt's (1977) formula: This method is also used in CH3D.
!                                2 = Smith's (1980) formula: This method is the default in GOM.
!                                3 = Wu's (1982) formula: This method is also used in SWAN.
!     wind_spinup          [i]   0/1   off/on,  Spinup for wind
!     wind_spinup_period   [r]   Wind spinup period in [day]
!
C30   wind_flag      airp_flag   num_windp_ser  wind_formular  wind_spinup    wind_spinup_period
      0              0           0              2              0              0.0
!========================================================================!
C31   Wind & pressure data type #2: using measured hurricane data or wind model data
!     Use hurricane data which includes wind and pressure
!     Data from National Hurricane Center (NOAA/NHC) or Hurricane Research Division (NOAA/HRD)
!
!     hurricane_flag          [i]   0  Wind and pressure (Hurricane type) will not be calculated,
!                                      Following varialbes will not be activated.
!                                   1  Wind and pressure will be calculated, hurricane (wind and pressure) data file hurricane_ser
!                                      Following variables will be activated.
!     hurricane_data_type     [i]   1  Wind and pressure data should be ascii format
!                                   2  wind and pressure data should be binary format
!     hurricane_interp_method [i]   1  time-only interpolate; use this when you only have wind model data
!                                   2  time & space interpolate: Inverse Distance Weighting (IDW) Interpolation
!                                      Hurricane center data file, hurricane_center.inp, should exist
!     hurricane_dt            [i]      Hurricane data interval in [sec]
!                                      Number of hurricane data set in hurricane_ser.inp should be greater than:
!                                         [hurricane_end_jday - hurricane_start_jday]*(86400/hurricane_dt) + 1
!                               188 ~ 193 julian day
C31
C31   hurricane_flag          hurricane_data_type     hurricane_interp_method    hurricane_dt
      1                       1                       1                          21600       ! 6 hourly data
!========================================================================!
C31_1 Hurricane start/end-year/month/day
!     Activate if hurricane_flag == 1
!
!     hurricane_start_year    [i]   Hurricane (or wind model data) start year
!     hurricane_start_month   [i]   Hurricane (or wind model data) start month
!     hurricane_start_day     [i]   Hurricane (or wind model data) start day
!     hurricane_end_year      [i]   Hurricane (or wind model data) end year
!     hurricane_end_month     [i]   Hurricane (or wind model data) end month
!     hurricane_end_day       [i]   Hurricane (or wind model data) end day
!
C31_1 hurricane_start_year    hurricane_start_month   hurricane_start_day
C31_1 hurricane_end_year      hurricane_end_month     hurricane_end_day
      2010                    8                       1
      2010                    9                       30
!========================================================================!
C32   Wind & pressure data type #3: using analytical hurricane data
!     (not yet finished)
!     Later, I will also include Willoughby et al. (2006)'s model
!     Use Holland storm surge model
!
!     holland_flag         [i]   0/1   off/on, Following variables will (or not) be activated.
!     holland_start_jday   [r]
!     holland_end_jday     [r]
!
C32   holland_flag      holland_start_jday   holland_end_jday
      0                 0.0                  0.0
!========================================================================!
! C32 is the end of the third set.                                       !
! C40 ~ C49: Output Control Options                                      !
!========================================================================!
```

Figure 3-18. Card Image C30 ~ C32.

Card Images C30 through C32 specify Air/Sea boundary conditions. Users can use three different approaches to include atmospheric (mostly wind) effects: (1) using wind station data, (2) using hurricane type data, and (3) using

analytical hurricane data. The first one, i.e., C30, is the most basic option, and users can use this when users want to use wind and air pressure station data directly. If this option is selected, GOM will interpolate atmospheric station data into a model grid nodal point using the Inverse Distance Weighting (IDW) interpolation method.

When users want to use either wind model data or want to conduct hurricane simulation, the second option, i.e., C31, can be used. The main conceptual difference of the data set between C30 and C31 is whether the data is provided at entire model grid nodes (C31, I defined this data set as a hurricane type) or not (C30). For example, a wind model data covers the entire model grid, thus users must choose the second data type. Then, GOM distinguishes the provided data by either wind model data or hurricane data. Users must select **hurricane_interp_method** = 1 for the general wind model. However, for the hurricane simulation, it is better to select **hurricane_interp_method** = 2 to use time and space interpolation. The main difference is that hurricane simulation requires *hurricane_center.inp* which is the hurricane center tracking (hurricane eye's x and y location at each time step) data. Users can also test **hurricane_interp_method** = 1 for the hurricane simulation, but it will give less accurate results than **hurricane_interp_method** = 2. When **hurricane_inter_method** = 2 is selected, the results will be better and accurate, but the simulation time will be increased by doing both time and space interpolation. When choosing wind model or hurricane data type, wind and pressure data at entire nodes must be provided with *hurricane_ser.inp*. The *hurricane_ser.inp* is independent from the data start time which is defined in C3 provides data start and end time for *hurricane_ser.inp*. There is a reason why I used this approach. The *hurricane_ser.inp* becomes quite big when the model grid is big, and that is why I selected this approach to reduce this file size.

The 3rd atmospheric data type is the analytical hurricane data type as defined at C32. Users can generate analytical wind and pressure data using Holland's storm surge model (Holland, 1980). The card image C32 is the end of the third set.

```
!====================================================================!
C40   Check grid
!
!     check_grid_2D        [i]   0/1   off/on, write a 2D grid checking file - origin shifted version (for the nice view)
!     check_grid_2DO       [i]   0/1   off/on, write a 2D grid checking file - original version
!     check_grid_3D        [i]   0/1   off/on, write a 3D grid checking file
!     check_grid_format    [i]   1/2/3 Tecplot/VTK/both file format
!     check_grid_info      [i]   0/1   off/on, write additional grid information - not yet activated
!     check_grid_unit_conv [r]   unit conversion factor for the coordinate
!                                1.0   =  [m], original unit in the grid information
!                                0.001 = [km], change the unit to kilometer
!
C40   check_grid_2D     check_grid_2DO    check_grid_3D     check_grid_format check_grid_info   check_grid_unit_conv
      1                 1                 1                 2                 0                 0.001
!====================================================================!
```

Figure 3-19. Card Image C40.

```
!====================================================================!
C41
!     tser_station_num     [i]   Total station for time series output files
!     tser_info_option     [i]   How do you want to provide the time series station information?
!                                0  In the following sub-card; following sub-card must activated
!                                1  Provide the time series station information in the additional tser_station_info.inp;
!                                   if this is selected, do not activate the following sub-card.
!     tser_hloc            [i]   Time series output horizontal location
!                                1 = cell center
!                                2 = node
!     tser_frequency       [i]   Time series output file (***_tser.dat) frequency
!     tser_time            [i]   output file time unit
!                                1 = second, 2 = minute, 3 = hour, 4 = day
!     tser_format          [i]   1/2/3 Tecplot/VTK/both file format
!
C41   tser_station_num  tser_info_option  tser_hloc   tser_time   tser_frequency tser_format
      5                 0                 1           4           6              2              ! every hr; dt = 600s
!====================================================================!
C41_1 Read individual time series station element number
!     Activate if tser_station_num > 0
!
!     serial_num        [i]   Station number (dummy number)
!     tser_station_cell [i]   Station element number: will be used only if [tser_loc == 1]
!     tser_station_node [i]   Station node number: will be used only if [tser_loc == 2]
!     tser_station_name [c]   Station name. Maximum 25 characters.
!
C41_1 serial_num  tser_station_cell tser_station_node    tser_station_name
      1           772               999                 'MSD'
      2           22197             999                 'DPI'
      3           14200             999                 'MBL'
      4           6975              999                 'BOS'
      5           1949              999                 'MEP'
!====================================================================!
C41_2 Activate/deactivate time series variables
!     Activate if tser_station_num > 0
!
!     tser_eta          [i]   0/1   off/on, write water surface elevation time series: eta_tser_from_msl.xxx,
!     tser_H            [i]   0/1   off/on, write total water depth: H_tser.xxx
!     tser_u            [i]   0/1   off/on, write vertically averaged velocity time series: u_tser.xxx
!     tser_v            [i]   0/1   off/on, write vertically averaged velocity time series: v_tser.xxx
!     tser_salt         [i]   0/1   off/on, write vertically averaged salinity time series: salt_tser.xxx
!     tser_temp         [i]   0/1   off/on, write vertically averaged temperature time series: temp_tser.xxx, not yet included
!     tser_pressure     [i]   0/1   off/on, write air pressure time series at surface: airp_tser.xxx, not yet included
!
C41_2 tser_eta tser_H   tser_u   tser_v   tser_salt   tser_temp   tser_pressure
      1        0        1       1       1           1           0
!====================================================================!
```

Figure 3-20. Card Image C41, C41_1, & C41_2.

```
!==========================================================================!
C42   2D output file (tec2D_###.dat or vtk2D_###.dat) control
!
!     IS2D_switch:    [i]   0/1   off/on option
!     IS2D_flood_map [i]    0/1   off/on 2D maximum flood map; write one time at the end of simulation.
!                           Works only when IS2D_switch is on
!     IS2D_format:    [i]   1/2/3 Tecplot/VTK/both file format
!     IS2D_binary:    [i]   0/1   Select file output format: ASCII or Binary
!                           0 = ASCII
!                           1 = Binary  ! not yet activated
!     IS2D_File_freq:[i]    2D out file generation frequency:
!                           A new file generation every [dt * IS2D_File_freq] times
!                           This option only works with Tecplot files
!     IS2D_time:      [i]   output file time unit
!                           1 = second, 2 = minute, 3 = hour, 4 = day
!     IS2D_frequency:[i]    2D contour plot frequency: display time = dt*IS2D_frequency [sec]
!                           Note: IS2D_File_frequency does not affect on VTK files.
!     IS2D_start:     [i]   2D data start time (simulation step)
!     IS2D_end:       [i]   2D data end time (simulation step)
!     IS2D_unit_conv:[r]    unit conversion factor for the coordinate
!                           1.0    = [m],  original unit in the grid information
!                           0.001 = [km], change the unit to kilometer
!
C42   IS2D_switch    IS2D_flood_map
C42   IS2D_format    IS2D_binary    IS2D_File_freq
C42   IS2D_time      IS2D_frequency IS2D_start     IS2D_end    IS2D_unit_conv
      0              0                                                  ! dt = 600
      2              0              144                                 ! every 1 day simulation
      4              6              1              999999      0.001    ! every 1 hr
!--------------------------------------------------------------------------!
C42_1 2D output file (tec2D_###.dat or vtk2D_###.dat) variable selection
!     It will be used only if (IS2D_switch == 1), so do no deactivate this card.
!     Note: vertically averaged values will be printed out.
!
!     variable:       [i]   dummy variable id (do not change order)
!     IS2D_variable: [i]    0/1   off/on option for each variable
!
C42_1 variable    IS2D_variable(:)
      eta         1
      u           1
      v           1
      salt        1
      temp        0
      rho         0
!==========================================================================!
```

Figure 3-21. Card Image C42 & C42_1.

```
!=========================================================================!
C43   3D output file (tec3D_###.out or vtk3D_###.dat) control
!     Currently, only for the surface elevation (eta)
!
!     IS3D_full_switch: [i]   0/1   off/on option for 'tec3D_full_***.dat' or 'vtk3D_full_***.dat'
!     IS3D_surf_switch: [i]   0/1   off/on option for 'tec3D_surf_***.dat' or 'vtk3D_surf_***.dat'
!     IS3D_format:      [i]   1/2/3 Tecplot/VTK/both file format
!     IS3D_binary:      [i]   0/1   Select file output format: ASCII or Binary
!                             0 = ASCII
!                             1 = Binary  ! not yet activated
!     IS3D_File_freq:   [i]   3D out file generation frequency
!                             A new file generation every [dt * IS2D_File_freq] times
!     IS3D_time:        [i]   output file time unit
!                             1 = second, 2 = minute, 3 = hour, 4 = day
!     IS3D_frequency:   [i]   3D contour plot frequency: display time = dt*IS3D_frequency [sec]
!                             Note: IS3D_File_freq does not affect on VTK files.
!     IS3D_start:       [i]   3D data start time (simulation step)
!     IS3D_end:         [i]   3D data end time (simulation step)
!     IS3D_unit_conv:   [r]   unit conversion factor for the coordinate
!                             1.0    = [m],  original unit in the grid information
!                             0.001 = [km], change the unit to kilometer
!
C43   IS3D_full_switch  IS3D_surf_switch
C43   IS3D_format       IS3D_binary       IS3D_File_freq
C43   IS3D_time         IS3D_frequency    IS3D_start     IS3D_end       IS3D_unit_conv
      1                 0                                                ! dt = 600
      2                 0                 144                            ! every 1 day simulation
      4                 6                 8640           999999          0.001 ! every 1 hr; from 60day
!-------------------------------------------------------------------------!
C43_1 3D output file (tec3D_###.dat or vtk3D_###.dat) variable selection
!     It will be used only if (IS3D_full_switch == 1 or IS3D_surf_switch == 1), so do not deactivate this card.
!
!     variable:      [i]   dummy variable id (do not change order)
!     IS3D_variable: [i]   0/1   off/on option for each variable
!
C43_1 variable    IS3D_variable(:)
      u           1
      v           1
      w           1
      salt        1
      temp        0
      rho         0
!=========================================================================!
```

Figure 3-22. Card Image C43 & C43_1.

```
!=========================================================================!
C44   2D Dump file: dump2D_****.dat
!     Vertically averaged values of [eta, u, v, salt, temp, rho] at nodes will be written
!
!     IS2D_dump_switch:    [i]   0/1   off/on option for 'dump2D_****.dat'
!     IS2D_dump_binary:    [i]   Select file output format: ASCII or Binary
!                                0 = ASCII
!                                1 = Binary
!     IS2D_dump_time:      [i]   output file time unit
!                                1 = second, 2 = minute, 3 = hour, 4 = day
!     IS2D_dump_File_freq: [i]   2D dump file generation frequency
!                                A new file generation every [dt * IS2D_dump_File_freq] times
!     IS2D_dump_frequency: [i]   Output dump frequency: printout time = dt*IS2D_dump_frequency [sec]
!     IS2D_dump_start:     [i]   2D dump start time (simulatioin step)
!     IS2D_dump_end:       [i]   2D dump end time (simulation step)
!
C44   IS2D_dump_switch        IS2D_dump_binary     IS2D_dump_time    IS2D_dump_File_freq
C44   IS2D_dump_frequency     IS2D_dump_start      IS2D_dump_end
      0                       0                    4                 48       ! generate a file every day
      2                       1                    999999                     ! dump every hr
!=========================================================================!
```

Figure 3-23. Card Image C44.

```
!========================================================================!
C45   3D Dump file: dump3D_****.dat
!     eta:              at node & surface only,
!     u, v, w:          at node & at vertical level [z0 ~ zn]
!     salt, temp, rho:  at node & at vertical layer [zl ~ zn]; mid-position
!
!     IS3D_dump_switch:    [i]   0/1   off/on option for 'dump3D_****.dat'
!     IS3D_dump_binary:    [i]   Select file output format: ASCII or Binary
!                                0 = ASCII
!                                1 = Binary
!     IS3D_dump_time:      [i]   output file time unit
!                                1 = second, 2 = minute, 3 = hour, 4 = day
!     IS3D_dump_File_freq: [i]   3D dump file generation frequency
!                                A new file generation every [dt * IS3D_dump_File_freq] times
!     IS3D_dump_frequency: [i]   Output dump frequency: printout time = dt*IS3D_dump_frequency [sec]
!     IS3D_dump_start:     [i]   3D dump start time (simulatioin step)
!     IS3D_dump_end:       [i]   3D dump end time (simulation step)
!
C45   IS3D_dump_switch        IS3D_dump_binary     IS3D_dump_time    IS3D_dump_File_freq
C45   IS3D_dump_frequency     IS3D_dump_start      IS3D_dump_end
      0                       0                    4                 144      ! generate a file every day
      6                       1                    999999                     ! dump every hr
!========================================================================!
```

Figure 3-24. Card Image C45.

```
!========================================================================!
C46   Write diagonostic files for important subroutines
!
!     dia_advection:       [i]   0/1   off/on for 'solve_nonlinear_advection.f90'
!     dia_momentum:        [i]   0/1   off/on for 'solve_momentum_equation.f90'
!     dia_freesurface      [i]   0/1   off/on for 'solve_freesurface_equation.f90'
!     dia_eta_at_ob        [i]   0/1   off/on in  'solve_freesurface_equation.f90'
!     dia_bottom_friction  [i]   0/1   off/on for 'calculate_bottom_friction.f90'
!     dia_face_velocity    [i]   0/1   off/on in  'calculate_horizontal_velocities.f90 & calculate_vertical_velocities.f90'
!     dia_node_velocity    [i]   0/1   off/on in  'calculate_velocity_at_node.f90'
!
C46   dia_advection        dia_momentum      dia_freesurface      dia_eta_at_ob
C46   dia_bottom_friction  dia_face_velocity dia_node_velocity
      0                    0                 0                    0
      0                    0                 0
!========================================================================!
! End of file ===========================================================!
```

Figure 3-25. Card Image C46.

Card Image C40 through the end covers output control options. The GOM provides two types of direct output data formats: (1) Tecplot, and (2) VTK file format. Tecplot is a commercial product, and it is easy to use. The Visualization Toolkit (VTK) can be used for several open visualization tools such as ParaView and VisIt. The variables named with *****_File_freq** (e.g., **IS2D_File_freq)** control a file separation into pieces of sub-files. One-dimensional time series files are usually not that big, thus the file does not need to be separated. However, 2D or 3D output files will be easily over several Gigabytes (GB), depending on the total number of elements. In this case, users can tell GOM how often generates sub-files, and a new file will be generated every, for example, [**dt**× **IS3D_File_freq**] second, and a new file will have **IS3D_File_freq** time data. For example, if **dt** = 600 sec and **IS3D_File_freq** = 1440 (as shown in C43), a new 3D output file will be generated every 10 days (= 600 sec× 1440 = 864000 sec = 10 days). A new file, e.g., *tec3d_****.dat*, where * denotes the file number: 1, 2, 3, ..., N, will be generated every 10 days, and it will have hourly data sets (i.e., 240 data sets) if **IS3D_frequency** = 6 (i.e., **dt**× **IS3D_frequency =** 600 sec× 6 = 3600 sec). Note that only the first file will have one more data set since it includes the initial condition. Since the Tecplot file formats are lighter than VTK file formats, the simulation speed will be

faster with the Tecplot file format options than the VTK file format options.

The GOM also provides binary output options, and in this case, users can easily construct any visualization tool-type data structure from the binary output file. Note that you will get the maximum simulation speed with the binary output options, thus these options will be benefitial especially for a large project.

The last Card Image C46 controls the diagnostic files for some important subroutines. The diagnostic files provide more detailed information at each time step.

4 Additional Input Files

The GOM uses several additional input files in addition to three essential input files which are explained in the previous section.

4.1 harmonic_ser.inp

```
!C harmonic_ser.inp
!C
!C    harmonic_ser_id_dummy:     [i]   ID number of tidal (harmonic) elevation time series. Dummy variable
!C    no_tidal_constituent:      [i]   Total number of tidal constituent in each series. This number should not need to be identic
!C    tidal_phase_shift:         [r]   Tidal phase shift in [degree].
!C                                     To use cosine wave: 0.0
!C                                     To use sine wave: 90.0
!C    tidal_period:              [r]   Tidal period in [hour] at current location (boundary node)
!C    tidal_amplitude:           [r]   Tidal amplitude at current location (boundary node)
!C    tidal_phase:               [r]   Tidal phase in [degree] at current location (boundary node)
!C    tidal_nodal_factor:        [r]   Nodal factor,f, at the reference time (model simulation begin time)
!C                                     This should be identical for same tidal constituent.
!C                                     Should be close to 1.0
!C    equilibrium_argument:      [r]   Earth equilibrium argument (phase at Greenwich) in [degree] at the reference time (model si
!C                                     This should be identical for same tidal constituent.
!C    forcing_name:              [c]   Forcing symbol (dummy variable, just for the reference, will not be read)
!C
!C    harmonic_ser_id_dummy   no_tidal_constituent    tidal_phase_shift
!C    tidal_period            tidal_amplitude   tidal_phase    tidal_nodal_factor   equilibrium_argument    forcing_name
      1                       1                       0.0
      12.42                   0.3048            0.0           1.0                  0.0                     ! M2
```

Figure 4-1. harmonic_ser.inp.

The input file *harmonic_ser.inp* specifies the harmonic tidal elevation variation at the specified boundary cells. Periodic tidal boundaries can be included setting **num_tide_bc** > 0 (in C20). If **num_tide_bc** is greater than 0, the user should specify how many harmonic tidal series are involved as setting **num_harmonic_ser** (in C20). If these two variables are specified (and of course **num_ob_cell** > 0), harmonic tidal series file, *harmonic_ser.inp*, should be provided. The *harmonic_ser.inp* requires the number of **num_harmonic_ser** harmonic tidal information sets. Note that most of the rest time-series input files have a similar structure.

4.2 eta_ser.inp

```
!C eta_ser.inp:
!C
!C    eta_ser_id_dummy     [i]   ID number of surface elevation (eta) time series. Dummy variable
!C    eta_ser_data_num     [i]   Number of time data lines at each time series
!C    eta_ser_time_conv    [r]   Time unit conversion factor to [sec]
!C    eta_ser_time_adjust  [r]   Additive adjustment to time values
!C    eta_ser_unit_conv    [r]   Surface elevation (eta) unit conversion factor to [m]
!C    eta_ser_adjust       [r]   Additive adjustment to eta values
!C    eta_ser_time:        [r]   Surface elevation data time
!C    eta_ser_eta:         [r]   eta (surface elevation) in [m]
!C
!C    eta_ser_id_dummy  eta_ser_data_num  eta_ser_time_conv    eta_ser_time_adjust  eta_ser_unit_conv    eta_ser_adjust
!C    eta_ser_time   eta_ser_eta: (1:eta_ser_data_num,1:num_ob_cell)
        1             3                3600.0              0.0                  1.0                  0.0
     0.00000   0.00
     1.00000   0.05
  9999.00000   0.05
        2             3                3600.0              0.0                  1.0                  0.0
     0.00000   0.00
     1.00000  -0.05
  9999.00000  -0.05
```

Figure 4-2. eta_ser.inp

The input file *eta_ser.inp* specifies the water surface elevation variation at the specified boundaries. If **num_eta_bc** (in C20) is greater than 0, *eta_ser.inp* is required. Like *harmonic_ser.inp*, *eta_ser.inp* requires the total number of **num_eta_ser** data sets, and each data set consists of a header line and **eta_ser_data_num** lines.

4.3 q_ser.inp

```
!C q_ser.inp:
!C
!C    q_ser_id_dummy    [i]   ID number of river (Q) time series. Dummy variable
!C    q_data_num        [i]   Total number data points in each series
!C    q_time_unit_conv  [r]   Time unit conversion factor to [sec]
!C    q_time_adjust     [r]   Additive adjustment to time values
!C    q_unit_conv       [r]   Q unit conversion factor to [m3/s]
!C    q_adjust          [r]   Additive adjustment to q values
!C    q_interp_method   [i]   Choose one of the following two interpolation methods
!C                              1  Linear Interpolation
!C                              2  Lagrange Interpolation
!C    q_ser_time:       [r]   Time. This number should start from 0 for correct interpolation.
!C    q_ser_Q:          [r]   Q (discharge rate)
!C
!C q_ser_id_dummy    q_data_num(:)     q_time_unit_conv     q_time_adjust  q_unit_conv    q_adjust    q_interp_method
!C    q_ser_time  q_ser_Q
   1                   2                86400.0             0.0           1.0            0.0         1
      0.0       10.0 10.0 10.0 10.0 10.0
 999999.0       10.0 10.0 10.0 10.0 10.0
   2                   2                86400.0             0.0           1.0            0.0         1
      0.0       -10.0 -10.0 -10.0 -10.0 -10.0
 999999.0       -10.0 -10.0 -10.0 -10.0 -10.0
```

Figure 4-3. q_ser.inp

The input file *q_ser.inp* specifies river inflow or withdraws from specified boundary cells at each vertical layer. The file structure and concepts of variables in *q_ser.inp* are identical to *eta_ser.inp*.

4.4 wind_ser.inp

```
!C windp_ser.inp:
!C
!C    windp_ser_id          [i]   ID number of wind time series. Dummy variable
!C    windp_ser_data_num    [i]   Total number of data points in each station
!C    windp_time_conv       [r]   Time unit conversion factor to [sec]
!C    windp_time_adjust     [r]   Additive adjustment to time values
!C    windp_station_node    [i]   Closest node number for this station
!C    windp_station_name    [s]   dummy variable
!C    windp_ser_time        [r]   Julian Time started from the beginning of the simulation start year.
!C    windp_u               [r]   Wind speed in x-direction [m/s]; note that this unit is fixed.
!C    windp_v               [r]   Wind speed in y-direction [m/s]; note that this unit is fixed.
!C    windp_p               [r]   Pressure in [Millibar]; note that this unit is fixed.
!C
!C windp_ser_id      windp_ser_data_num   windp_time_conv   windp_time_adjust    windp_station_node   windp_station_name(i)
!C    windp_ser_time    windp_u  windp_v  windp_p
   1                 2                    86400.0           0.0                  1                    ! example wind
   0.0      9.140     0.00      1013.0
 999.0      9.140     0.00      1013.0
```

Figure 4-4. wind_ser.inp.

The file *wind_ser.inp* specifies measured wind and atmospheric pressure information. The file structure and concepts of variables in *wind_ser.inp* are identical to *eta_ser.inp*.

4.5 hurricane_ser.inp

```
!C hurricane_ser.inp
!C Real-time hurricane data from NOAA/HRD
!C
!C    serial_num:             [i]   dummy number for the node number
!C    wind_u:                 [r]   Wind velocity component [m/s]: x-directioin
!C    wind_v:                 [r]   Wind velocity component [m/s]: y-direction
!C    air_p                   [r]   Atmospheric pressure in [millibar]
!C
!C hurricane_start_jday    hurricane_end_jday   hurricane_dt
!C serial_num  wind_u      wind_v       air_p
        1    0.06    6.51   1015.0647       ! the first time set start for maxnod (=126474)
        2    0.06    6.51   1015.0641
        3    0.05    6.51   1015.0635
        .     .        .     .
        .     .        .     .
        .     .        .     .
   126472   -1.09    6.94   1013.9925
   126473   -1.09    6.94   1013.9908
   126474   -1.09    6.94   1013.9916       ! end of the first time set
        1    0.98    6.10   1016.4757       ! the second time set start for maxnod (=126474)
        2    0.98    6.10   1016.4756
        3    0.98    6.10   1016.4754
        .     .        .     .
        .     .        .     .
        .     .        .     .
   126472    0.26    7.69   1015.6986
   126473    0.26    7.69   1015.6976
   126474    0.26    7.69   1015.6982       ! end of the second time set
```

Figure 4-5. hurricane_ser.inp.

The file hurricane_ser.inp may be a quite big file depending on the grid size. The data starts and end times are defined in C31_1, and the data interval between each data set is defined with **hurricane_dt**.

4.6 salt_init.inp & salt_ser.inp / temp_init.inp & temp_ser.inp

```
! salt_init.inp:
! initial salinity condition at each horizontal nodal and vertical center position at each layer
! If you want to add more comments, then put '!' at the first column of the line.
! node_id, (salt_node(i,k), k=1,maxlayer)
!
1   0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
2   0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
3   0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
.     .     .
.     .     .
.     .     .
61  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
62  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
63  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
```

Figure 4-6. salt_init.inp

```
!C salt_ser.inp:
!C
!C    salt_ser_id_dummy:   [i]   ID number of salinity [psu] time series. Dummy variable
!C    salt_data_num:       [i]   Total number data points in each series
!C    salt_time_unit_conv: [r]   Time unit conversion factor to [sec]
!C    salt_time_adjust     [r]   Additive adjustment to time values
!C    salt_unit_conv:      [r]   Salinity unit conversion factor to [kg/m3]
!C    salt_adjust:         [r]   Additive adjustment to salinity values
!C    salt_ser_time:       [r]   Time. This number should start from 0 for correct interpolation.
!C    salt_ser_salt:       [r]   Salinity
!C
!C salt_ser_id_dummy    salt_ser_data_num    salt_time_conv    salt_time_adjust  salt_unit_conv    salt_adjust
!C    salt_ser_time  salt_ser_salt
   1                 2                   1.0               0.0               1.0               0.0
      0.0        30.0
  999999.0        30.0
```

Figure 4-7. salt_ser.inp

Initial salinity and temperature data are provided with *salt.inp* and *temp.inp*, respectively, at the layer of each node. The boundary time series files are *salt_ser.inp* and *temp_ser.inp*, respectively.

4.7 Obc_info.inp & Qbc_info.inp

```
! Obc_info.inp; this is identical to C20_1
C20_1 Eta boundary information
!     Activate if num_ob_cell > 0.
!     Should have 'num_ob_cell' information (lines).
!
!     serial_num        [i]   Station number (dummy number)
!     ob_nodes(i,1)     [i]   First node of the station (or the element, in counterclockwise direction)
!     ob_nodes(i,2)     [i]   Second node of the station (or the element, in counterclockwise direction)
!     ob_eta_type       [i]   Open boundary surface elevation type:
!                            -1  Radiation bc
!                                No input is required; elevations will be computed as average of surrounding elevations.
!                             1  This boundary is forced by time history of elevation. eta_ser.inp is required
!                             2  This boundary is forced by tidal harmonic constituents; harmonic_ser.inp is required; default = c
!     harmonic_ser_id   [i]   ID number of harmonic series at this station.
!                                If ob_eta_type == 2 or 3, it will be activated.
!                                It should be greater than 0.
!     eta_ser_id        [i]   ID number of surface elevation time series at this station.
!     salt_ser_id       [i]   ID number of salinity time series at this station.
!     temp_ser_id       [i]   ID number of temperature time series at this station.
!     ob_name           [c]   dummy variable. This will not be read.
!
C20_1 serial_num  ob_nodes(i,1)  ob_nodes(i,2)  ob_eta_type(i) harmonic_ser_id(:)   eta_ser_id(:)  salt_ser_id temp_ser_id ob_name
      1           4295           4294           1              0                     1              1           0           ! Pass
      2           4294           4127           1              0                     1              1           0
      .             .              .            .              .                     .              .           .
      .             .              .            .              .                     .              .           .
      .             .              .            .              .                     .              .           .
     69            50             51            1              0                     2              2           0
     70            51             52            1              0                     2              2           0
```

Figure 4-8. Obc_info.inp

5 Compiling and Executing the Code

Source codes, which are all written in standard FORTRAN 90, are stored in the '*source*' directory. The three of the most important source codes that users must take a look at are (1) *mod_global_variables.f90*, (2) *mod_file_definition.f90*, and (3) *main.f90*. The global variables and parameters for GOM are defined in *mod_global_variables.f90*. If users want to take a look at input and output files and folder structure or want to change input and output file/folder names, *mod_file_definition.f90* is the source code users must take a look at. The straightforward simulation flow chart of the GOM can be found in the main program, *main.f90*.

5.1 Install Fortran Compiler

To compile the source code, users need a Fortran 90 compiler, and there are several different compilers. Among them, I specifically prefer to use GFortran from the GNU project, and it is a free compiler. If you use Linux/Unix, get it with (for example):

```
sudo ap-get install gfortran
```

Then, check if it is installed correctly with:

```
gfortran --version
```

If gfortran is successfully installed, you will see like this:

```
GNU Fortran (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3
Copyright (C) 2011 Free Software Foundation, Inc.

GNU Fortran comes with NO WARRANTY, to the extent permitted by law.
You may redistribute copies of GNU Fortran
under the terms of the GNU General Public License.
For more information about these matters, see the file named COPYING
```

Figure 5-1. gfortran version check result.

If you are a Windows user, I recommend you to download either (1) Minimalist GNU for Windows (MinGW) or (2) Cygwin (www.cygwin.com), which is a Unix-like environment and command-line interface for Microsoft Windows. After install either MinGW or Cygwin, check if it is installed correctly with the method I explained above.

5.2 Compile Source codes

To store source codes and object files in separate folders, the *makefile*, which describes the relationships among source codes and provides commands for updating each file, is located in '/source/release'. To compile source codes type '*make all*' on the command line, then the *makefile* will compile only newer source codes compared to the corresponding object files and link object files, and finally make an executable file as defined in the *makefile*, e.g., *run.exe*. If there are no object files in the folder, the command '*make all*' will compile all source codes and generate

an executable file. The command '*make clean*' will remove (or clean) all the object files and an executable file. Users should copy the executable file, e.g., *run.exe*, into the model simulation folder.

The executable file, *run.exe*, should be located under the project folder with '*input*' and '*output*' folder together:

- '/project_folder/input
- ./project_folder/output
- ./project_folder/run.exe

If the OpenMP option is selected in the *makefile*, you may need to set the total core numbers with the environmental variable, OMP_NUM_THREAD. Then, type '*./run.exe*' on the command window to run the GOM. The followings are the command I explained above:

```
make clean
make all
cp run.exe <your_project_folder>
export OMP_NUM_THREADS=8 (if it is compiled with OpenMP option, you want to use 8 cores.)
./run.exe
```

Now you will see the GOM simulation progress on the terminal window.

6 Source Codes and Subroutines

The current version of the GOM (GOM_v1.0) consists of more than one hundred standard FORTRAN source files including one main program and three modules. Names and brief descriptions of source codes are shown in Table 6.1.

Table 6.1. Subroutines in GOM. (not yet finished)

Subroutine	Description

7 References

Casulli, V., Walters, R.A., 2000. An unstructured grid, three-dimensional model based on the shallow water equations. International Journal for Numerical Methods in Fluids 32, 331–348. https://doi.org/10.1002/(SICI)1097-0363(20000215)32:3<331::AID-FLD941>3.0.CO;2-C

Casulli, V., Zanolli, P., 2005. High resolution methods for multidimensional advection–diffusion problems in free-surface hydrodynamics. Ocean Modelling, The Second International Workshop on Unstructured Mesh Numerical Modelling of Coastal, Shelf and Ocean Flows 10, 137–151. https://doi.org/10.1016/j.ocemod.2004.06.007

Holland, G.J., 1980. An Analytic Model of the Wind and Pressure Profiles in Hurricanes. Mon. Wea. Rev. 108, 1212–1218. https://doi.org/10.1175/1520-0493(1980)108<1212:AAMOTW>2.0.CO;2

Ji, Z.-G., 2017. Hydrodynamics and Water Quality: Modeling Rivers, Lakes, and Estuaries, 2nd edition. ed. Wiley, Hoboken, NJ.

Kincaid, D.R., Respess, J.R., Young, D.M., 2017. ITPACK 2C: A FORTRAN Package for Solving Large Sparse Linear Systems by Adaptive Accelerated Iterative Methods.

Kincaid, D.R., Respess, J.R., Young, D.M., Grimes, R.R., 1982. Algorithm 586: ITPACK 2C: A FORTRAN Package for Solving Large Sparse Linear Systems by Adaptive Accelerated Iterative Methods. ACM Trans. Math. Softw. 8, 302–322. https://doi.org/10.1145/356004.356009

Lee, Jun, Lee, Jungwoo, Yun, S.-L., Kim, S.-K., 2020. Three-Dimensional Unstructured Grid Finite-Volume Model for Coastal and Estuarine Circulation and Its Application. Water 12, 2752. https://doi.org/10.3390/w12102752

Smagorinsky, J., 1963. GENERAL CIRCULATION EXPERIMENTS WITH THE PRIMITIVE EQUATIONSI. THE BASIC EXPERIMENT. Mon. Wea. Rev. 91, 99–164. https://doi.org/10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2