Grid2Grid : HOS Wrapper Program

YoungMyung Choi, Maite Gouin, Guillaume Ducrozet,

Benjamin Bouscasse and Pierre Ferrant

17 November, 2017

Version 1.0

Ecole Centrale de Nantes

Contents

1 Introduction

Wave generation solvers using Higher Order Spectral Method (HOS) have been validated and developed for several years (Ducrozet et al. (2016), Ducrozet et al. (2007), Bonnefoy et al. (2011) and Ducrozet et al. (2012)). HOS solves nonlinear wave propagation in open-sea (HOS-Ocean) and also in numerical wave tank (HOS-NWT) with low computation time comparing with other nonlinear wave solvers because the theory is based on pseudo-spectral method (HOS (b) and HOS (a)). Those HOS wave solvers are released as open-source codes, which anyone can develop, use and distribute under the terms of GNU General Public Licence (GPLv3).

Nonlinear irregular wave generation in computational fluids dynamic (CFD) solvers becomes important recently in naval fields to better estimate the loads on offshore structure. The conventional linear superposition methods imply a long computational time for wave generation and the simulation is made almost impossible without having enough computational power. And if the method is based on linear wave theory, there is also question on occurence of nonlinear phenomenon and the interaction between waves as the simulation goes. Therefore dedicated nonlinear wave solvers with high computational speed are needed.

`Grid2Grid` is developed as a wrapper program of HOS to generate wave fields from the results of HOS computation. `Grid2Grid` reconstructs wave fields of HOS with inverse fast Fourier transforms (FFTs) and uses a quick spline module, the nonlinear wave fields can be fastly reconstructed for arbitrary simulation time and space. The nonlinear wave simulation is then possible for a particular position and time where specific non linear phenomenon occur.

`Grid2Grid` compiles a shared library (`libGrid2Grid.so`) which can be used for communication with other progamming language using the `ISO_C_BINDING` rule. It compiles also a post processing program of HOS.

2 Program Structure

The program structure of `Grid2Grid` is depicted in Fig. 2.1. The program is composed of several modules (`.f90`) which are denoted as `mod`. HOS wave fields are generated by `modSurf2Vol` from the HOS result file (`modes_HOS_SWENSE.dat`). Because the results files only contains modes information, the volumic wave fields is reconstructed on HOS grid by inverse FFTs (denoted as H_2 Operator). The volumic wave fields in `Grid2Grid` are called as snapshot of wave fields because it is re-constructed at certain simulation time and grid of HOS. `modVol2Vol` constructs the interpolation data structure from several snapshot of wave fields by using multidimensional spline module (Williams (2015)). It can directly communicate with other Fortran language. To communicate with other languages, `modGrid2Grid` can also be used as it receives input data and return wave fields data by using `ISO_C_BINDING`. `modPostGrid2Grid` is a post processing module of HOS generating wave fields in VTK format and also wave elevation time series.

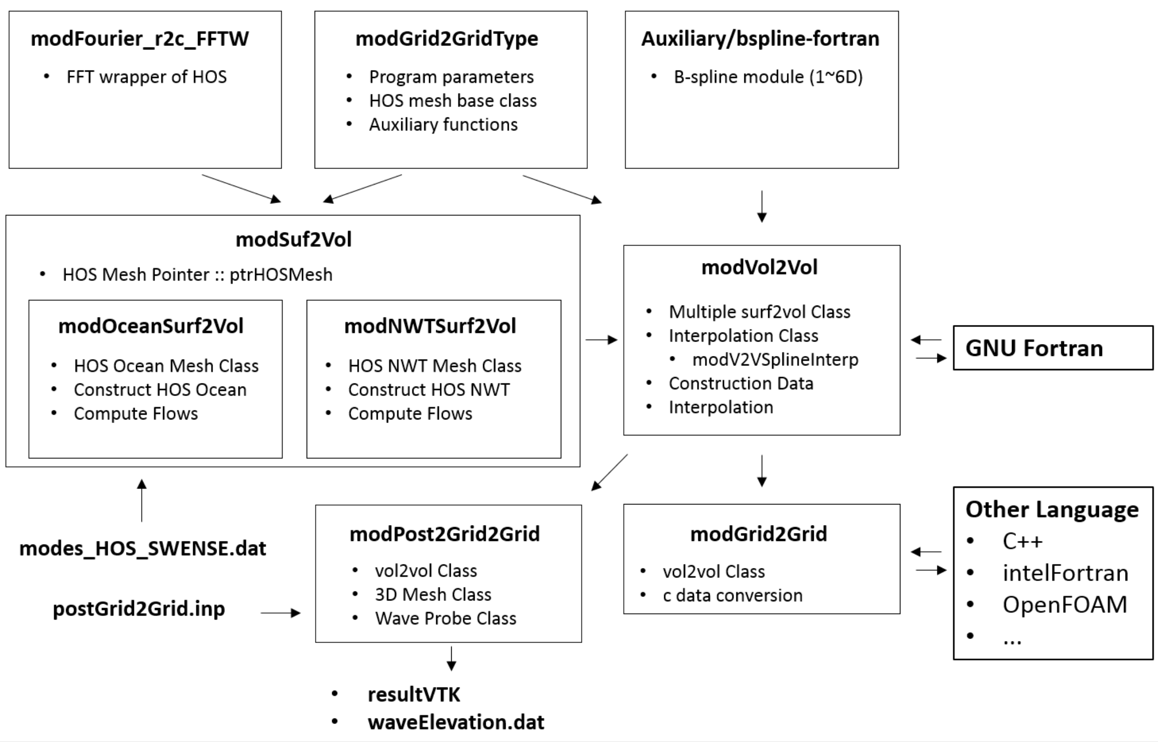


Figure 2.1. Program Structure of `Grid2Grid`

The main feature of each module is following.

- `modSurf2Vol` : Wrapper class of HOS-Ocean and HOS-NWT wave fields
- `modVol2Vol` : Interpolation of snapshot of wave fields (multiple Surf2Vol Class)
- `modGrid2Grid` : Communication with other language with `ISO_C_BINDING`
- `modPostGrid2Grid` : Post processing of HOS (wave fields and wave elevation)
- `modOceanSurf2Vol` : Re-construction HOS-Ocean wave fields
- `modNWTSurf2Vol` : Re-construction HOS-NWT wave fields
- `modFourier_r2c_FFTW` : Wrapper module of `FFTW` open library
- `modGrid2GridType` : Grid2Grid global variables and auxiliary functions
- `bspline-fortran` : Multidimensional b-spline module written in Fortran.

2.1 typSurf2Vol

2.1.1 Description

`Surf2Vol` is wrapper class to access HOS Ocean and NWT class. Functionality and data of both HOS classes are similar but the mathematical formulation and meshes are a little bit different. Therefore the flow information from each HOS grid is only taken by using HOS mesh base class and transfer it to `Vol2Vol` Class. It is reminded that the direct access on the HOS wave fields is only possible by HOS mesh pointer to prevent misuse of HOS wave data.

`Surf2Vol` class structure is depicted in Fig. 2.2. `Surf2Vol` class contains `HOSOceanSurf2Vol` and `HOSNWTSurf2Vol` classes. When the functionality is called, it calls the functionality of sub-class distiguished by HOS type. When the subroutine `correct` is called, the sub-class reads HOS modes from HOS result file and conducts H_2 operation to re-construct wave fields. Reconstructed wave field is saved on derived HOS grid class. And the HOS grid pointer at `Surf2Vol` points to derived HOS grid class to access wave fields.

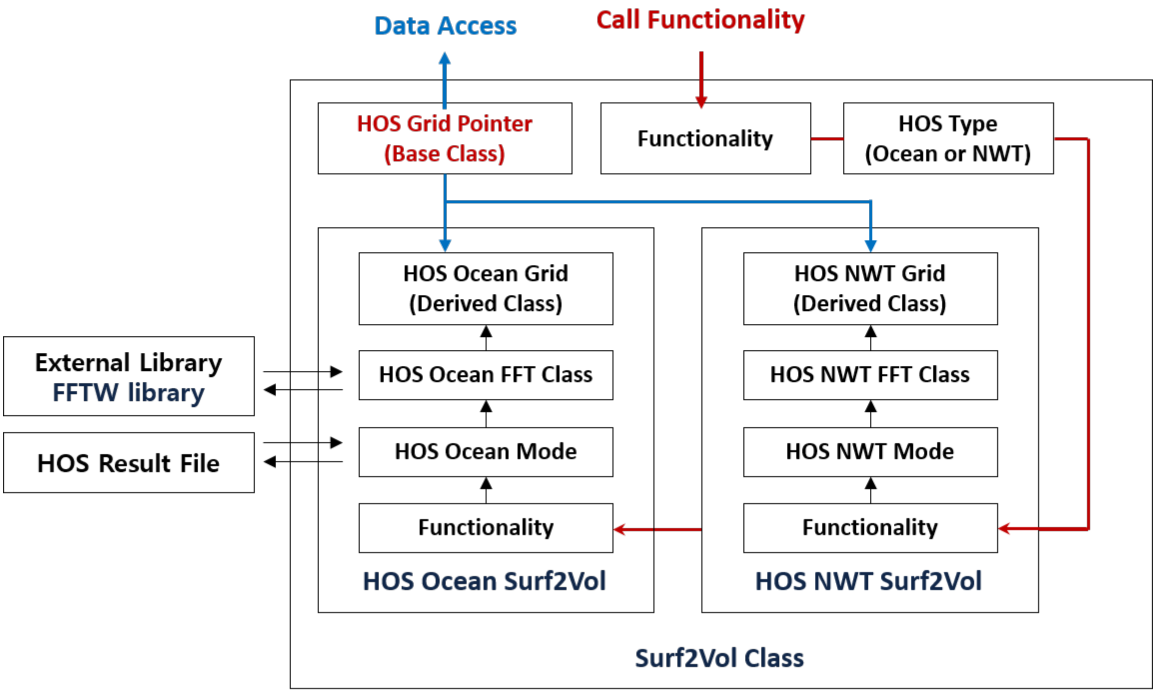


Figure 2.2. `Surf2Vol` class structure

`hosNWTSurf2Vol` and `hosOceanSurf2Vol` classes reconstruct HOS wave fields from HOS result file by using the H_2 operator. Wave reconstruction and HOS wave theory is well explained in Ducrozet et al. (2007) and Ducrozet et al. (2012). Each class constructs wave fields on derived HOS mesh class which can be pointed by HOS mesh base class.

HOS wave theory is based on the superposition of base modes which satisfy Laplace equation, sea bottom and free surface boundary condition basically, the behavior of the mode function has an exponential profile. This exponential profile could magnify local modes which has high wave number above $z = 0$. For high steepness waves, this property gives unnatural values closed to free surface, criterion to cut local mode is introduced as (2.1).

$$f_{mn}(z) = \begin{cases} \dfrac{\cosh k_{mn}(z+H)}{\cosh k_{mn} H} & \text{if} \quad f_{mn}(z) < C_{f(z)} \\ C_{f(z)} & \text{if} \quad f_{mn}(z) \geq C_{f(z)} \end{cases} \tag{2.1}$$

where

k_{mn} : HOS mode wave number $\quad z$: HOS z-coordinates $\quad H$: water depth

$C_{f(z)}$: $f_{mn}(z)$ function criterion value (default = 10)

The parameter $C_{f(z)}$ is set to the default value of 10 in `Grid2Grid`. If the local wave velocities generated by `Grid2Grid` are not sufficient, the parameter can be changed. The parameter is defined in `modGrid2GridType.f90` as a `FNZ_VALUE`.

HOS result file only contains mode amplitudes computed at each HOS simulation time, the volumic grid should then be reconstructed from those modes. HOS does not need vertical grid information, so z-directional grid information should be given when HOS `surf2vol` is initialised. Constructed HOS grid is used for interpolation grid.

To constuct the HOS grid, following information should be given.

- `zMin`, `zMax`
- `nZMin`, `nZMax`
- `nZMinRatio`, `nZMaxRatio` [Optional]

where `zMin` and `zMax` is used to set z-directional domain and it should have negative and positive value. `nZMin` and `nZMax` are the number of z-directional grid. It is recommended to have at least 50-100 points for the interpolation. If a sufficient number of grid points is not given, the interpolation scheme could give strange values due to exponetial profile of $f_{mn}(z)$. `nZMinRatio` and `nZMaxRatio` are the ratio of maximum and minumum height of grid ($\Delta z_{max}/\Delta z_{min}$). Minimum grid is located at $z = 0$. Those are optional values set to be 3 as default. The grid can be visualized by using ParaView. The VTK file is located at VTK/Grid2Grid/surf2volMesh.vtk.

2.1.2 Class (Type)

Class : `Surf2Vol`

- Data :
  - `hosNWTSurf2Vol_` : HOS-NWT Surf2Vol Class
  - `hosOceanSurf2Vol_` : HOS-Ocean Surf2Vol Class
  - `ptrHOSMesh_` : HOS Mesh Pointer

- Functionality :
  - `initialize` : Initialise HOS Surf2Vol class with HOS type, result file path, ...
  - `correct` : Update HOS wave fields
  - `destroy` : Class desroyer

Class : `HOSOceanSurf2Vol` and `HOSNWTSurf2Vol`

- Data :
  - `HOSfile` : HOS result file (`modes_HOS_SWENSE.dat`)
  - `HOSmode` : HOS Ocean or NWT modes
  - `HOSmesh` : HOS Ocean or NWT mesh (Two are different)
  - `HOSfftw` : HOS FFT class for H_2 operator

- Public Functionality :
  - `initialize` : Initialise HOS Ocean or NWT Surf2Vol with result file path, ...
  - `correct` : Update HOS Ocean or NWT wave fields
  - `destroy` : Class desroyer

- Private Functionality :
  - `init_read_mod` : Read HOS number of modes and allocate dynamic array
  - `read_mod` : Read HOS modes for given HOS time index
  - `buildGlobalMesh` : Build HOS mesh with given HOS construction parameter
  - `reconstuctionFFTs` : Reconstruct wave fields by inverse H_2 operator

2.1.3 How to use

– Initialise Surf2Vol

```
Call hosS2V%initialize(hosType, filePath, zMin, zMax, nZmin, nZmax,
    zMinRatio, zMaxRatio)

! hosS2V          : Surf2vol Class (Type)
! hosType         : HOS Type (Ocean or NWT)
! filePath        : HOS result file path (modes_HOS_SWENS.dat)
! zMin, zMax      : HOS grid z-minimum and z-maximul (vertical domain)
! nZmin, nZmax    : Number of z-directional Grid
!
! zMinRatio, zMaxRatio (Optional)
! : Ratio of maximum and minimum height of grid (default=3)
```

– Correct Surf2Vol

```
Call hosS2V%correct(hosIndex)

! hosS2V          : Surf2vol Class (Type)
! hosIndex        : HOS time index
```

– Data access on wave field of Surf2Vol

◦ Wave elevation

```
eta = hosS2V%ptrHOSMesh_%eta(ix, iy)

! hosS2V          : Surf2vol class (Type)
! ix, iy          : HOS grid index (x, y)
! eta             : Wave elevation
```

◦ Wave velocity

```
u = hosS2V%ptrHOSMesh_%eta(ix, iy, iz)
v = hosS2V%ptrHOSMesh_%eta(ix, iy, iz)
w = hosS2V%ptrHOSMesh_%eta(ix, iy, iz)

! hosS2V          : Surf2vol class (Type)
! ix, iy, iz      : HOS grid index (x, y, z)
! u, v, w         : Wave velocity (x, y, z)
```

◦ Dynamic pressure ($p_d = p - \rho g z$)

```
pd = hosS2V%ptrHOSMesh_%pd(ix, iy, iz)

! hosS2V          : Surf2Vol class (Type)
! ix, iy, iz      : HOS grid index (x, y, z)
! pd              : Dynamic pressure
```

– Destruct of `Surf2Vol`

```
Call hosS2V%destroy()

! hosS2V          : Surf2Vol class (Type)
```

2.2 typVol2Vol

2.2.1 Description

`Vol2Vol` is an interpolation class used to give interpolated wave information data from the reconstructed HOS wave field. It holds several `Surf2Vol` classes and interpolation class. The `Vol2Vol` class structure is described in Fig 3.1.

HOS result file holds modes amplitudes time series for the whole HOS simulation time. If we construct HOS wave fields and interpolation data structure for the whole HOS simulation time, not only the computation time is long but also a huge memory is demanded. `Grid2Grid` aims for construction of demanding wave fields for relatively short period and domain, it is not necessary to construct the whole HOS wave fields and interpolation data structure. Therefore revolving algorithm is applied just to update HOS wave fields adjucent to simulation time and constuct small interpolation data structure for efficient computation time and memory.

When `initialize` of `Vol2Vol` is called, it allocates `Surf2Vol` array, interpolation data and array related to revolving algorithm. And it initialises `Surf2Vol` classes and interpolation class and call its subroutine `correct` at $t = 0$.

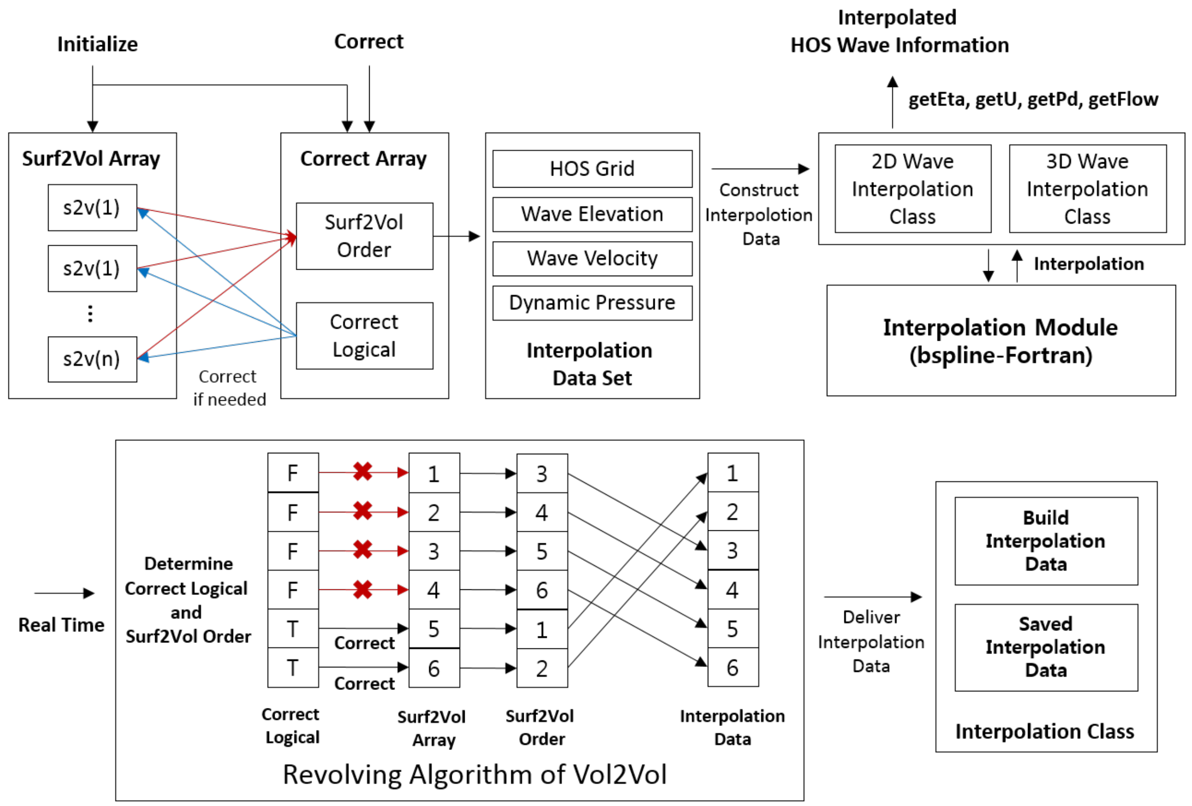


Figure 2.3. `Vol2Vol` class structure

The subroutine `correct` with input t first determine HOS `Surf2Vol` correction index and `Surf2Vol` order based on input t and previous HOS time index. By using correction index and order, only necessary HOS `Surf2Vol` is updated and re-ordered to constructed for interpolation data. After interpolation data is constructed, it is delivered to the interpolation class. The interpolation class communicates with the `bspline-Fortran` module and constructs the interpolation data structure.

The subroutine `getEta`, `getU`, `getPd` and `getFlow` return interpolated values for given space and time from the constructed interpolation data structure.

2.2.2 Class (Type)

Class : `Vol2Vol`

- Data :
  - `nInterp_` : Interpolation order (2 : Linear, 3 : Quadratic, 4 : Cubic, ...)
  - `nSaveT_` : Number of Surf2Vol wave fields (`nSaveT_` > `nInterp_`)
  - `HOSs2v_(:)` : Array of `Surf2Vol` class
  - `itp2D_` : Interpolation class for 2D waves
  - `itp3D_` : Interpolation class for 3D waves

- Functionality :
  - `initialize` : Initialise HOS Vol2Vol class with HOS type, result file path, ...
  - `correct` : Update HOS wave fields with real-time
  - `getEta` : Get interpolated wave elevation
  - `getU` : Get interpolated wave velocity
  - `getPd` : Get interpolated dynamic pressure
  - `getFlow` : Get flow information
  - `destroy` : Class destroyer

2.2.3 How to use

– Initialise `Vol2Vol`

```
Call hosV2V%initialize(hosType, filePath, zMin, zMax, nZmin, nZmax,
 zMinRatio, zMaxRatio, iflag)

! hosV2V          : Vol2vol Class (Type)
! hosType         : HOS Type (Ocean or NWT)
! filePath        : HOS result file path (modes_HOS_SWENS.dat)
! zMin, zMax      : HOS grid z-minimum and z-maximul (vertical domain)
! nZmin, nZmax    : Number of z-directional Grid
!
! zMinRatio, zMaxRatio  (Optional)
! : Ratio of maximum and minimum height of grid (default=3)
!
! iflag : Wriging option (iflag = 1, write Grid2Grid information)
```

– Correct `Vol2Vol`

```
Call hosV2V%correct(simulTime)

! hosV2V          : Vol2vol Class (Type)
! simulTime       : Simulation time (real time value)
```

– Get wave elevation from `Vol2Vol`

```
eta = hosV2V%getEta(x, y, simulTime, iflag)

! hosV2V          : Vol2vol Class (Type)
! x, y            : x and y position
! simulTime       : Simulation time (real time value)
!
! eta             : Wave elevation
!
! iflag (Optional)
! : if iflag = 1, nondimensional x and y can be given (default = 0).
```

– Get wave velocity from `Vol2Vol`

```
Call hosV2V%getU(x, y, z, simulTime, u, v, w, iflag)

! hosV2V        : Vol2vol Class (Type)
! x, y, z       : x, y and z position
! simulTime     : Simulation time (real time value)
!
! u, v, z       : Wave velocity ( x, y, z )
!
! iflag (Optional)
! : if iflag = 1, nondimensional x and y can be given (default = 0).
```

– Get dynamic pressure from `Vol2Vol`

```
pd = hosV2V%getPd(x, y, z, simulTime, iflag)

! hosV2V        : Vol2vol Class (Type)
! x, y, z       : x, y and z position
! simulTime     : Simulation time (real time value)
!
! pd            : Dynamic velocity (pd = p - rho * g * z)
!
! iflag (Optional)
! : if iflag = 1, nondimensional x and y can be given (default = 0).
```

– Get flow information from `Vol2Vol`

```
Call hosV2V%getFlow(x, y, z, simulTime, eta, u, v, w, pd, iflag)

! hosV2V        : Vol2vol Class (Type)
! x, y, z       : x, y and z position
! simulTime     : Simulation time (real time value)
!
! eta           : Wave elevation
! u, v, z       : Wave velocity ( x, y, z )
! pd            : Dynamic velocity (pd = p - rho * g * z)
!
! iflag (Optional)
! : if iflag = 1, nondimensional x and y can be given (default = 0).
```

– Destroy `Vol2Vol`

```
Call hosV2V%destroy()
```

2.3 modGrid2Grid

2.3.1 Description

`modGrid2Grid` is not a class but a Fortan module which contains an array of `Vol2Vol` classes and subroutines to communicate with other languages. The module structure is depicted in Fig. 2.4. It has two static data which are input data array and `Vol2Vol` array not to have and update multiple `Vol2Vol` classes for efficiency and to save memory. When `Grid2Grid` is initialized from a program written in another language, it firstly check the input variables and return `HOSIndex`. If the same input variables are given, it returns the same `HOSIndex` and does not allocate nor initialise `Vol2Vol`. Other languages can distiguish different HOS wave theory by using the `HOSIndex`. The subroutines `correct` and `get*` can be called with `HOSIndex`, position and time.

The array size of `modGrid2Grid` is 100 by default (It can deal with 100 different HOS waves) but it does not consume much memory because classes of `Grid2Grid` are programmed with dynamic arrays and consequently only a few of static data are used. If more HOS wave theories are needed, the variable `nMaxVol2Vol` in `src/modGrid2Grid.f90` can be changed to deal with over than 100 different HOS wave theory.

The functionality of `modGrid2Grid` is almost the same with `Vol2Vol`.

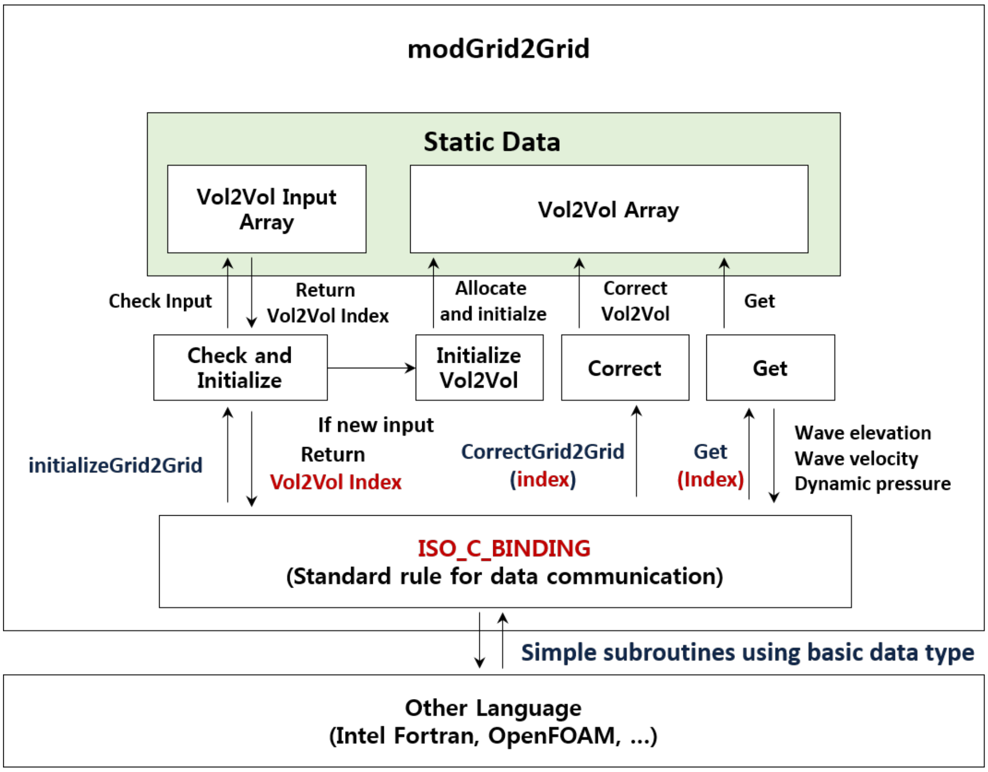


Figure 2.4. `Grid2Grid` Module structure

2.3.2 Subroutines

The interface between other languages with `modGrid2Grid` will be detailed in Chap. 4. Subroutines in `modGrid2Grid` follows `ISO_C_BINDING`.

Static subroutines :

– Initialise Grid2Grid

  ◦ `__modgrid2grid_MOD_initializegrid2grid`

– Correct Grid2Grid

  ◦ `__modgrid2grid_MOD_correctgrid2grid`

– Get wave elevation

  ◦ `__modgrid2grid_MOD_gethoseta`

– Get wave velocity

  ◦ `__modgrid2grid_MOD_gethosu`

– Get dynamic pressure

  ◦ `__modgrid2grid_MOD_gethospd`

– Get flow information

  ◦ `__modgrid2grid_MOD_gethosflow`

– Get HOS simulation end time

  ◦ `__modgrid2grid_MOD_gethosendtime`

– Get HOS water depth

  ◦ `__modgrid2grid_MOD_gethoswaterdepth`

– Get logical data indicating HOS wave theory is initialized

  ◦ `__modgrid2grid_MOD_isgrid2gridinitialized`

2.4 PostGrid2Grid

2.4.1 Description

`PostGrid2Grid` is a HOS post-processing class. It generates 3D VTK files of wave fields for visualization and wave elevation time series computed from `Vol2Vol` class. Wave fields at desired simulation time and spatial domain and wave elevation time series can be re-generated at some provided wave probes position.

`PostGrid2Grid` algorithm is depicted in Fig. 2.5. `PostGrid2Grid` is initialised with input file. The input file `postGrid2Grid.inp` contains HOS grid information and post processing information. `PostGrid2Grid` first reads and checks the input file and then build 3D visualization grid and wave probes. `Vol2Vol` class is also initialised. The subroutine `doPostProcessing` do time loop of `correct`. Subroutine `correct` first corrects the `Vol2Vol` class and gets the wave fields. If the grid option is set to no air mesh, 3D grid is fitted to wave elevation. It writes the results on files (3D VTK file and wave elevation time series).

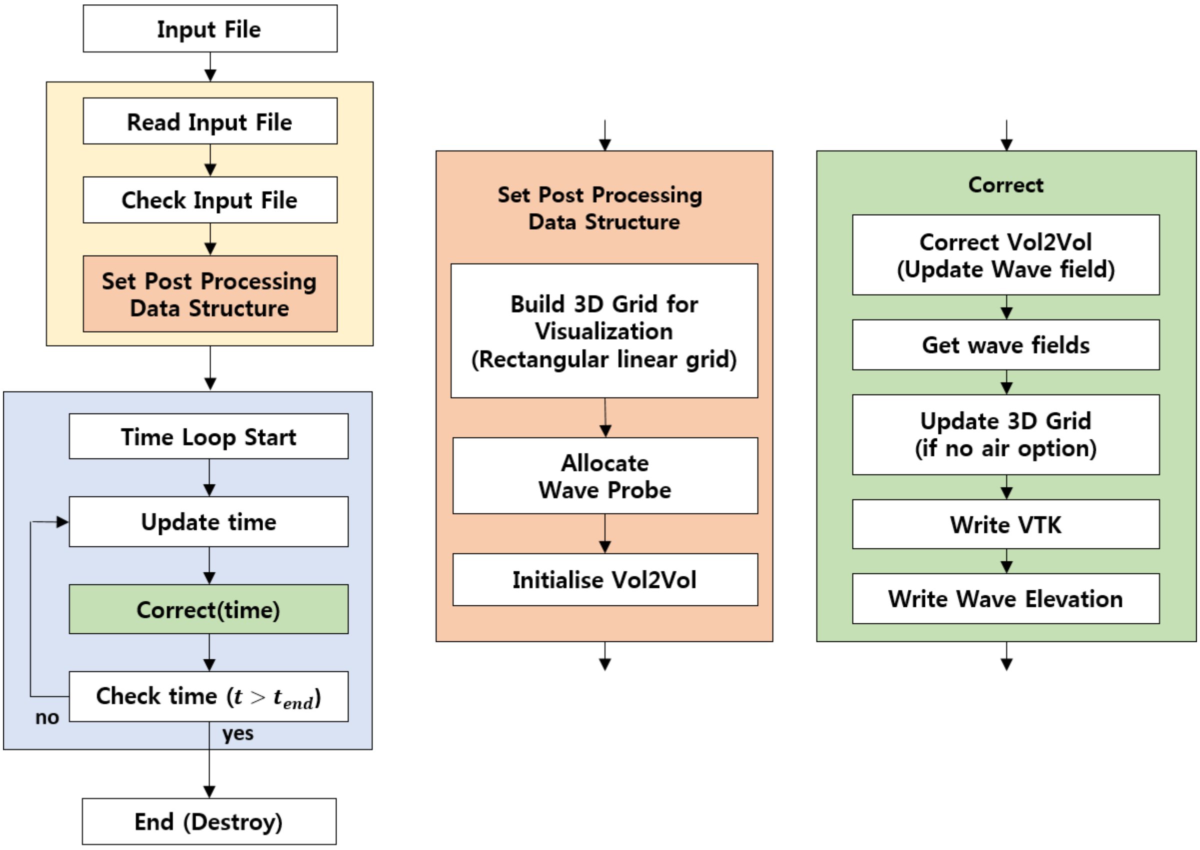


Figure 2.5. `PostGrid2Grid` Algorithm

2.4.2 Class(Type)

Class : `PostGrid2Grid`

- Data :
  - `hosVol2Vol_` : `Vol2Vol` class
  - `rectLGrid_` : Rectangular linear grid for 3D wave fields (VTK output)
  - `waveProbe_(:)` : Wave probe

- Functionality (Public) :
  - `initialize` : Initialise PostGrid2Grid class
  - `correct` : Correct `Vol2Vol`, `rectLGrid_` and `waveProbe_` and write output
  - `writeVTK` : Write 3D wave fields in VTK format
  - `doPostProcessing` : Do post processing
  - `destroy` : Destuctor of PostGrid2Grid class

- Functionality (Private) :
  - `readPostG2GInputFile` : Read `PostGrid2Grid` input file
  - `checkPostG2GParameter` : Check `PostGrid2Grid` input file
  - `writeVTKtotalASCII` : Write wave fields (Including air domain)
  - `writeVTKnoAirASCII` : Write wave fields (Grid is fitted to wave elevation)
  - `writeWaveProbe` : Write wave elevation time series

2.4.3 Input File of PostGrid2Grid

`PostGrid2Grid` needs input file. The input file name is `postGrid2Grid.inp`. The input is recognized by `keyword`. Input file has free format. `keyword` can be located at any line of file but `keyword` should be located as first word at line. If special character is added on character, it is recognized as comment. The input keyword is following.

– HOS Type (`solver`) and Result File (`hosFile`)

```
### Select Solver (Ocean or NWT) ---------------------------------- #

solver    Ocean

### hosFile Path --------------------------------------------------- #

hosFile   modes_HOS_SWENSE.dat
```

– Post Processing Time (`startTime`, `endTime`, `dt`)

```
### Post Processing Time ------------------------------------------- #

startTime   2712.0
endTime     2812.0
dt          0.1
```

– Write VTK Option (`writeVTK`) and Air Meshing Option (`airMesh`)

```
### Write 3D VTK File (true or false) ------------------------------ #

writeVTK    true

### Air Meshing (true or false) ------------------------------------ #

airMesh      true

# If airMesh is true or yes, grid will be constructed up to zMax.
#            is false or no, mesh will be constructed up to wave
#            elevation.
```

– 3D Output Domain Size (xMin, xMax, yMin, yMax, zMin, zMax)

```
### 3D Output Domain Size ------------------------------------------ #

xMin     1200
xMax     2000

yMin     1200
yMax     2000

zMin   -100.0
zMax     50.0

#   zMin and zMax are used to construct Surf2Vol HOS Grid.
```

– Number of Grid (nX, nY, nZmin, nZmax)

```
### Number of Mesh for 3D Output ----------------------------------- #

nX            500
nY            500

nZmin         100
nZmax          60

#   nZmin and nZmax are used to construct Surf2Vol HOS Grid.
```

– Vertical meshing scheme (zMesh)

```
### Z meshing scheme ----------------------------------------------- #

zMesh        meshRatio    3.0    3.0

# Meshing Scheme
#
# uniform    : uniform grid
# sine       : sine spaced grid (densed grid near to z=0)
# meshRatio : grid with constant geometric ratio. Minimum dz located
#              at z=0. (given two ratio is dz_max/dz_min)
#
# meshRatio needs two ratios (ratio for z <= 0  and ratio for z > 0)
#
```

– Wave probe write option (`writeWaveProbe`) and Output file path (`waveProbeFile`)

```
### Wave probe write option (true or false) ------------------------ #

writeWaveProbe  true

### Wave Probe File path ------------------------------------------- #

waveProbeFile   waveElevation.dat

# If waveProbeFile is not given in input file.
# Default iutput file name "waveElevation.dat" is used.
```

– Wave probes

```
### Wave probe Input Format ---------------------------------------- #
#
#   There should be no blank line after nWaveProbe and probe data.
#   If blank line is given, last wave probes will be discarded.
#
#   There should be no blank line.
#
#   ## Format
#
#     nWaveProbe  nProbes
#     probe1name(option)     xPos1    yPos1
#     probe2name             xPos2    yPos2
#     ...
#     probeNname             xPosN    yPosN
#
# ----------------------------------------------------------------- #

### Wave Probe Input ----------------------------------------------- #

nWaveProbe  5
wp1     0.0     0.0
wp2     1.0     2.0
wp3     2.0     3.0
wp4     4.0     5.0
6.0     7.0

# probeName    xPosition    yPosition
#
# probeName is optional.
```

2.4.4 How to use

Fortran subroutine for `postGrid2Grid` is given as example.

```
!! Subroutine to run PostGrid2Grid -----------------------------------
Subroutine runPostGrid2Grid(inputFileName)
!! -------------------------------------------------------------------
Use modPostGrid2Grid          !! Use PostGrid2Grid Module
!! -------------------------------------------------------------------
Implicit none
Character(Len = * ),intent(in) :: inputFileName    !! postGrid2Grid.inp
Type(typPostGrid2Grid)          :: postG2G       !! postGrid2Grid Class
!! -------------------------------------------------------------------

!! Initialize PostGrid2Grid with Input File
Call postG2G%initialize(inputFileName)

!! Do Post Processing
Call postG2G%doPostProcessing()

!! Destroy
Call postG2G%destroy

!! -------------------------------------------------------------------
End Subroutine
!! -------------------------------------------------------------------
```

3 Installation

3.1 Pre-Install

3.1.1 FFTW Install

`Grid2Grid` needs fast Fourier transform (FFT) library. `FFTW` is GNU licenced FFT library. The installation order is the following:

1. Download FFTW library

At the FFTW website (http://www.fftw.org/download.html), latest version of FFTW is available. Download `FFTW` library.

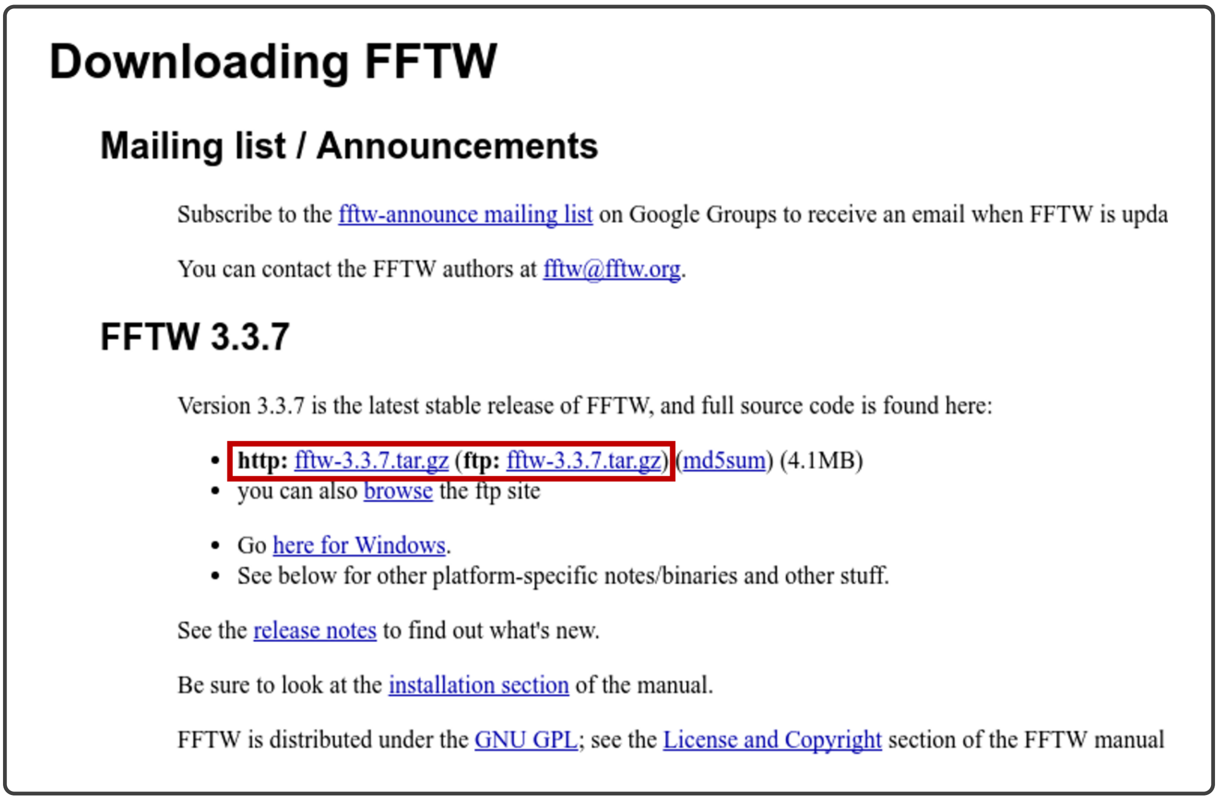


Figure 3.1. Download `FFTW library`

2. Extract FFTW library

```
$ tar -xvf fftw-3.3.7.tar.gz
```

3. Compile FFTW library

```
$ cd fftw-3.3.7/
$
$ export FFTW_PATH=$PWD
$
$ ./configure --prefix=$FFTW_PATH
$
$ make CFLAGS='-fPIC'
$
$ make install
```

4. Make soft link of FFTW library

If user has super user authority

```
$ sudo ln -s $FFTW_PATH/lib/libfftw3.a  /usr/local/lib/libfftw3.a
```

If user has no super user authority, third party path in `makefile` of `Grid2Grid` can be changed manually. If FFTW library (`libfftw3.a`) locates at `/home/lib/libfftw3.a`, then `makefile` of `Grid2Grid` can be changed as:

```
DTHRDLIB=/home/lib/
```

3.1.2 HOS Ocean and NWT (Optional)

`Grid2Grid` needs the result file of HOS solver. Installation of HOS is the following:

1. Install FFTW

See Chapter 3.1.1

2. Install makedepf90

```
$ sudo apt-get install makedepf90
```

3. Install LAPACK

```
$ sudo apt-get install liblapack-dev liblapack-doc-man liblapack-doc
 liblapack-pic liblapack3 liblapack-test liblapack3gf liblapacke
liblapacke-dev
```

4. Download HOS Ocean and NWT

```
$ # Path to desired installation path
$
$ cd $HOS_INSTALLATION_PATH
$
$ git clone https://github.com/LHEEA/HOS-ocean.git
$
$ git clone https://github.com/LHEEA/HOS-NWT.git
```

5. Change shared library path defined in makefile

Open `makefile` of HOS Ocean and NWT and change as following :

```
#LINKLIB = $(LIBDIR)libfftw3.a $(LIBDIR)liblapack.a $(LIBDIR)
librefblas.a

LINKLIB = $(LIBDIR)libfftw3.a -llapack
```

If FFTW s not installed at `$(LIBDIR)` in `makefile`, it can be changed as an example of previous chapter.

6. **Comple HOS NWT and HOS Ocean**

```
$ cd HOS-ocean/
$
$ make
$
$ cd ../HOS-NWT/
$
$ make
$
$ cd ..
```

7. **Check executable is generated**

```
$ # Check HOS NWT
$ ls HOS-NWT/bin/HOS-NWT
$
$ # Check HOS Ocean
$ ls HOS-ocean/bin/HOS-ocean
```

8. **Make soft link (optional)**

```
$ # Make Soft Link (Optional)
$
$ export HOS_PATH=$PWD
$
$ sudo ln -s $HOS_PATH/bin/HOS-NWT    /usr/bin/HOS-NWT
$
$ sudo ln -s $HOS_PATH/bin/HOS-ocean  /usr/bin/HOS-ocean
```

3.2 Grid2Grid Installation

3.2.1 Download Grid2Grid

Download Grid2Grid

```
$ # Path to desired installation path
$
$ cd $HOS_INSTALLATION_PATH
$
$ git clone https://github.com/LHEEA/Grid2Grid.git
```

Set FFTW Path (FFTW library is not installed on /usr/local/lib/

Change `DTHRDLIB` in `makefile` at FFTW library path.

```
DTHRDLIB=$FFTW_LIBRARY_PATH

# Example
# DTHRDLIB=/home/lib/      if "libfftw3.a" exists at /home/lib/
```

3.2.2 Compile PostGrid2Grid

```
$ cd Grid2Grid
$
$ make
```

3.2.3 Compile Grid2Grid Shared Library

Compile libGrid2Grid.so in Grid2Grid/obj/

```
$ make createlib
```

Compile libGrid2Grid.so in $FOAM_USER_LIBBIN

If OpenFOAM is installed, `libGrid2Grid.so` can be compiled at `$FOAM_USER_LIBBIN`. If OpenFOAM environment is called, following make rule can be used directly.

```
$ make createOFlib
```

4 Interface

4.1 GNU and Intel Fortran

An interface example fortran program is included in `interface/fortGrid2Grid`. To communicate with `Grid2Grid`, the fortran script `modCommG2G.f90` in `interface/fortGrid2Grid` is needed. It is a communication module with `libGrid2Grid.so`. The fortran interface example program is following :

```fortran
!! Program Start ---------------------------------------------------
Program Main
!! ---------------------------------------------------------------
use modCommG2G      !! Use Communication Module
!! ---------------------------------------------------------------
Implicit None
!! Variables -----------------------------------------------------
Integer,Parameter       :: nChar = 300      !! Default Character Length
Character(len = nChar) :: grid2gridPath    !! libGrid2Grid.so Path

integer                 :: hosIndex         !! HOS Index
Character(len = nChar) :: hosSolver        !! HOS Solver (Ocean or NWT)
Character(len = nChar) :: hosFileName      !! HOS Result File Path

Double precision        :: zMin, zMax       !! Surf2Vol Domain
integer                 :: nZmin, nZmax     !! Number of vertical grid
Double precision        :: zMinRatio, zMaxRatio   !! Grading ratio (=3)

Double precision        :: t, dt      !! Simulation Time, dt
Double precision        :: x, y, z    !! Computation Point
Double precision        :: eta, u, v, w, pd   !! HOS Wave Information
!! Dummy variables -------------------------------------------------
integer                 :: it         !! Dummy time loop integer

!! Program Body ----------------------------------------------------

!!!... Write Program Start
write(*,*) "Test program (Connect to Fortran) to use Grid2Grid shared
 library"

!!!... Set libGrid2Grid.so path.
!!!     It is recommended to use absolute path
! grid2gridPath = "/usr/lib/libGrid2Grid.so"  (if soft link is made)
grid2gridPath = "../../obj/libGrid2Grid.so"

!!!... Load libGrid2Grid.so and connect subroutines
Call callGrid2Grid(grid2gridPath)

!!!... Declare HOS Index
hosIndex = -1
```

```
!!!... Set HOS Type (Ocean or NWT)
hosSolver = "NWT"

!!!... Set HOS Result file Path
hosFileName = "modes_HOS_SWENSE.dat"

!!!... Set HOS Surf2Vol Domain and Vertical Grid
zMin = -0.6d0;          zMax =  0.6d0
nZmin = 50;             nZmax = 50
zMinRatio = 3.d0;       zMaxRatio = 3.d0

!!... Initialize Grid2Grid and Get HOS Index
Call initializeGrid2Grid(hosSolver, hosFileName, zMin, zMax, nZmin,
 nZmax, zMinRatio, zMaxRatio, hosIndex)

!! Time Information
t  = 0.0d0;     dt = 0.1d0

!! Given Point
x = 0.5d0;      y = 0.5d0;      z = -0.5d0

!! Time Loop
do it = 1,10

  !! Correct HOS Vol2VOl for given time
  Call correctGrid2Grid(hosIndex, t)

  !! Get Wave Elevation
  Call getHOSeta(hosIndex, x, y , t, eta)

  !! Get Flow Velocity
  Call getHOSU(hosIndex, x, y, z, t, u, v ,w)

  !! Get Dynamic Pressure
  Call getHOSPd(hosIndex, x, y, z, t, pd)

  !! Write Flow Information
  write(*,*) t, eta, u, v, w, pd

  !! Time Update
  t = t + dt
enddo

!! Write End of Program
write(*,*) "Test program (Connect to Fortran) is done ..."
!! -----------------------------------------------------------------
End Program
!! -----------------------------------------------------------------
```

4.2 OpenFOAM

An interface example OpenFOAM program is included in `interface/ofGrid2Grid`. The shared library `libGrid2Grid.so` should be compiled at `$FOAM_USER_LIBBIN`. To check `libGrid2Grid.so` exists at `$FOAM_USER_LIBBIN`, use following shell command :

```
$ ls $FOAM_USER_LIBBIN/libGrid2Grid.so
```

If `libGrid2Grid.so` not exists, refer to Chapter 3.2.

To call shared library `libGrid2Grid.so` in `$FOAM_USER_LIBBIN`, OpenFOAM compiling option is added at `Make/option`. Open `Make/option` and add following compiling option.

```
EXE_LIBS = \
          ...
          -lgfortran \
          -L$(FOAM_USER_LIBBIN) \
          -lGrid2Grid
```

OpenFOAM interface example program is given next page.

```
#include "fvCFD.H"

namespace Foam
{
  //- Grid2Grid Initial Character Length
  const int nCharGridGrid(300);

  //- Initialize Grid2Grid Class in Fortran
  //
  //  __modgrid2grid_MOD_initializegrid2grid
  //  (
  //      hosSolver,
  //      hosFileName,
  //      zMin,
  //      zMax,
  //      nZmin,
  //      nZmax,
  //      zMinRatio,
  //      zMaxRatio,
  //      hosIndex
  //  )
  //
  //    Input
  //      hosSolver              : "NWT" or "Ocean"
  //      hosFileName            : filePath of HOS mode result file
  //      zMin, zMax             : HOS grid zMin and zMax
  //      nZmin, nZmax           : HOS number of z grid
  //      zMinRatio, zMaxRatio : HOS z grid max/min ratio
  //
  //    Output
  //      hosIndex               : HOS Vol2Vol Index
  //
  extern "C" void __modgrid2grid_MOD_initializegrid2grid
  (
  const char[nCharGridGrid],
  const char[nCharGridGrid],
  const double*,
  const double*,
  const int*,
  const int*,
  const double*,
  const double*,
  int*
  );

  //- Correct Grid2Grid for given simulation Time
  //
  //  __modgrid2grid_MOD_correctgrid2grid(hosIndex, simulTime)
  //
  //    Input
```

```
//       hosIndex    : HOS Vol2Vol Index
//       simulTime   : Simulation Time
//
extern "C" void __modgrid2grid_MOD_correctgrid2grid
(
const int *,
const double *
);

//- Get HOS Wave Elevation
//
//   __modgrid2grid_MOD_gethoseta(hosIndex, x, y, t, eta)
//
//     Input
//       hosIndex : HOS Vol2Vol Index
//       x, y, t  : (x and y) position and simulation Time (t)
//
//     Output
//       eta      : wave elevation
//
extern "C" void __modgrid2grid_MOD_gethoseta
(
const int *,
const double *,
const double *,
const double *,
double *
);

//- Get HOS Flow Velocity
//
//   __modgrid2grid_MOD_gethosu(hosIndex, x, y, z, t, u, v, w)
//
//     Input
//       hosIndex     : HOS Vol2Vol Index
//       x, y, z, t   : (x, y, z) position and simulation Time (t)
//
//     Output
//       u, v, w      : (x, y, z) - directional flow velocity
//
extern "C" void __modgrid2grid_MOD_gethosu
(
const int *,
const double *,
const double *,
const double *,
const double *,
double *,
double *,
double *
);
```

```
    //- Get HOS Dynamic Pressure
    //
    //   __modgrid2grid_MOD_gethospd(hosIndex, x, y, z, t, pd)
    //
    //     Input
    //       hosIndex     : HOS Vol2Vol Index
    //       x, y, z, t   : (x, y, z) position and simulation Time (t)
    //
    //     Output
    //       pd       : Dynamic Pressure p = -rho*d(phi)/dt-0.5*rho*|U*U|
    //
    extern "C" void __modgrid2grid_MOD_gethospd
    (
    const int *,
    const double *,
    const double *,
    const double * ,
    const double *,
    double *
    );

    //- Get HOS Wave Elevation, Flow Velocity and Dynamic Pressure
    //
    //   __modgrid2grid_MOD_gethosflow(hosIndex, x, y, z, t, eta, u, v, w,
      pd)
    //
    //     Input
    //       hosIndex     : HOS Vol2Vol Index
    //       x, y, z, t   : (x, y, z) position and simulation Time (t)
    //
    //     Output
    //       eta          : wave elevation
    //       u, v, w      : (x, y, z) - directional flow velocity
    //       pd           : Dynamic Pressure p = -rho * d(phi)/dt - 0.5 *
      rho * |U * U|
    //
    extern "C" void __modgrid2grid_MOD_gethosflow
    (
    const int *,
    const double *,
    const double *,
    const double * ,
    const double *,
    double *,
    double *,
    double * ,
    double *,
    double *
    );

}
```

```
// Main OpenFOAM Program Start
int main(int argc, char *argv[])
{

  // Write Program Start
  Info << "OpenFOAM Program Example to Call Grid2Grid (HOS Wrapper) in
   OpenFOAM" << endl;

  // Set HOS Solver Type
  const word HOSsolver_("NWT");
  const word HOSFileName_("./modes_HOS_SWENSE.dat");

  // Set File Name
  string strHOSSolver = string(HOSsolver_);
  string strHOSFileName = string(HOSFileName_);

  // Set HOS Solver Type
  const char *HOSsolver = strHOSSolver.c_str();

  // Set HOS Mode Result File Path
  const char *HOSfileName = strHOSFileName.c_str();

  // Set HOS Z Grid Information
  int indexHOS(-1);

  double zMin(-0.6), zMax(0.6);
  int nZmin(50), nZMax(50);

  double zMinRatio(3.0), zMaxRatio(3.0);

  // Initialize Grid2Grid
  __modgrid2grid_MOD_initializegrid2grid(HOSsolver, HOSfileName,
  &zMin, &zMax,
  &nZmin, &nZMax,
  &zMinRatio, &zMaxRatio, &indexHOS);

  Info << "HOS Label : " << indexHOS << endl;

  // Set Position
  double x(0.5), y(0.0), z(-0.5);

  // Define Flow Quantities
  double eta, u, v, w, pd;

  // Set Simulation Time and Time Difference
  double simulTime(0.0);
  double dt(0.1);
```

```
  // Time Loop
  for (int it = 0; it < 11; it++)
  {
    // Correct Grid2Grid
    __modgrid2grid_MOD_correctgrid2grid(&indexHOS, &simulTime);

    // Get Wave Eta
    __modgrid2grid_MOD_gethoseta(&indexHOS, &x, &y, &simulTime, &eta);

    // Get Flow Velocity
    __modgrid2grid_MOD_gethosu(&indexHOS, &x, &y, &z, &simulTime, &u, &v
   , &w);

    // Get Dynamic Pressure
    __modgrid2grid_MOD_gethospd(&indexHOS, &x, &y, &z, &simulTime, &pd);

    Info << " sumulTime : " << simulTime << endl;
    Info << "   eta      : " << eta << endl;
    Info << "   u, v, w : " << u << " " << v << " " << w << endl;
    Info << "   pd       : " << pd << nl << endl;

    // Get whole Information
    __modgrid2grid_MOD_gethosflow(&indexHOS, &x, &y, &z, &simulTime, &
   eta, &u, &v, &w, &pd);
    Info << "   eta      : " << eta << endl;
    Info << "   u, v, w : " << u << " " << v << " " << w << endl;
    Info << "   pd       : " << pd << nl << endl;

    // Time Update
    simulTime+=dt;
  }

  return 0;
}
```

5 Validation

`foamStar` is used to validate `Grid2Grid`. `foamStar` is developed by Bureau Veritas and shared with Ecole Centrale de Nantes and it can simulate nonlinear waves, seakeeping problem and also hydro-elasticity problem. It solves multiphase problem with Reynolds Averaged Navier-Stokes equations (RANS) with Volume of fraction (VOF). It is based on standard multiphase solver in OpenFOAM (`interDymFoam`). To generate waves, `foamStar` uses explicit blending scheme which blends computed flow values to target values with weight function. The blending function is given as equation (5.1). Some details of `foamStar` are explained in Seng (2012) and Monroy et al. (2017).

$$\mathrm{U} = (1-w)\mathrm{U} + w\mathrm{U}^{target} \tag{5.1a}$$

$$\alpha = (1-w)\alpha + w\alpha^{target} \tag{5.1b}$$

where

$$w = 1 - (1 - w_{base})^{\chi} \qquad \chi = \frac{\Delta \mathrm{U} \Delta t}{\Delta x} \qquad w_{base} = -2\xi^3 + 3\xi^2$$

By using `Grid2Grid`, the target values at blending zone can be replaced by the wave components computed from HOS wave theory. For the validation, considered HOS simulation condition is given in Table 5.1.

Table 5.1. HOS Wave condition for validation

Wave Type	Value	HOS-Ocean		HOS-NWT	
		2D	3D	2D	3D
Regular Wave	T [s]	-	-	0.702	0.702
	H [m]	-	-	0.0431	0.0288
Irregular Waves	T_p [s]	0.702	1.0	1.0	0.702
	H_s[m]	0.0288	0.10	0.05	0.0384
	γ [-]	3.3	3.3	3.3	3.3
Extreme Waves	T_p [s]	-	-	17.5	-
	H_s[m]	-	-	15.5	-
	γ [-]	-	-	3.3	-

5.1 Simulation results

The wave elevation computed by using `foamStar` and HOS-Ocean are compared in Fig. 5.1. Small differences are observed but it is assumed to be caused by the resolution of the finite volume mesh which is not sufficient near the free surface. The VOF solver gives then those differences. Snapshots of HOS-Ocean wave fields are shown in Fig. 5.1.

The results of HOS-NWT for a two dimensional case is shown in Fig. 5.3. And a simulation snapshot is shown in Fig. 5.4. Three dimensional waves with HOS-NWT are also given in Figs. 5.5 and 5.6.

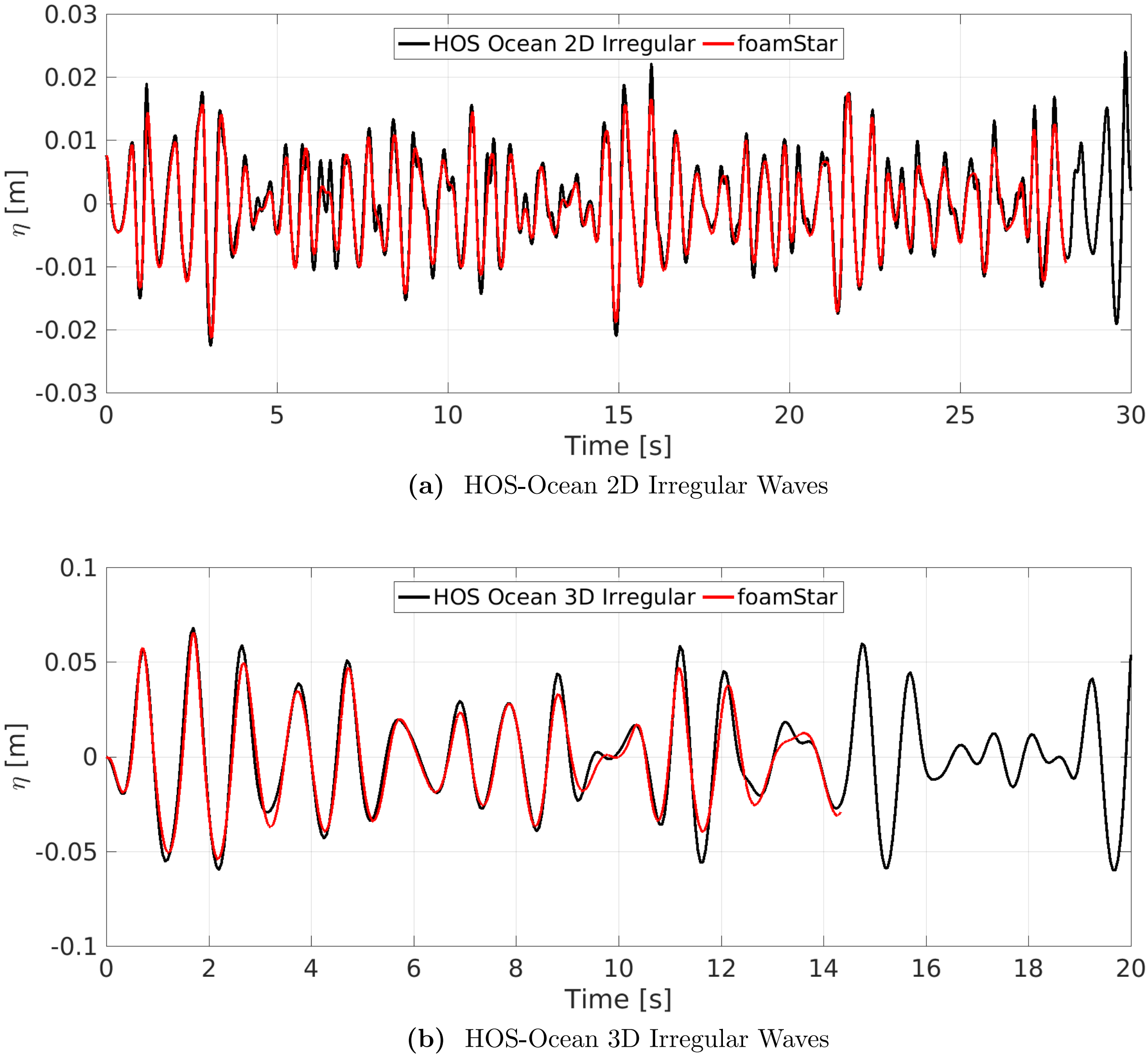
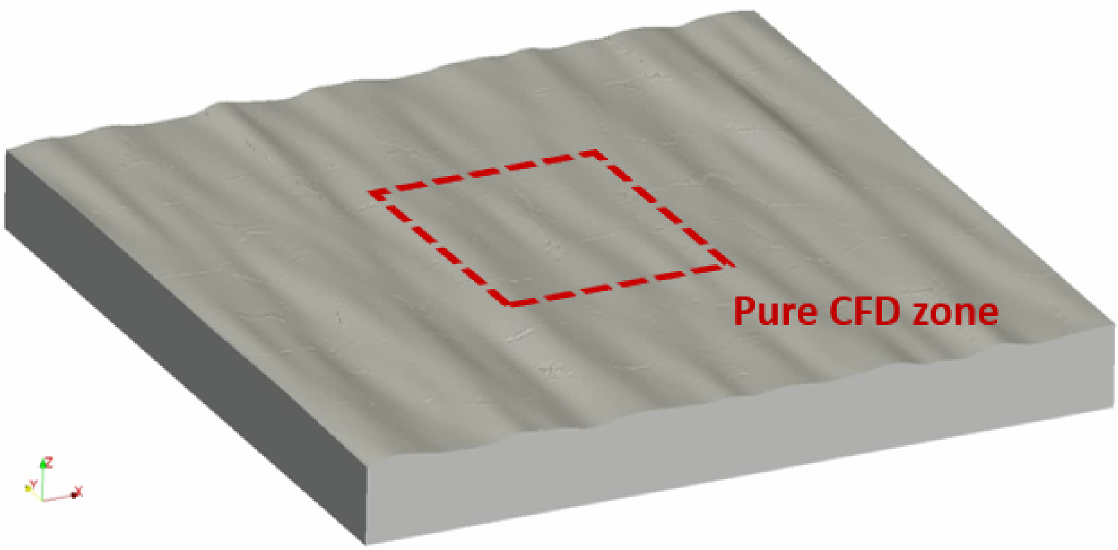


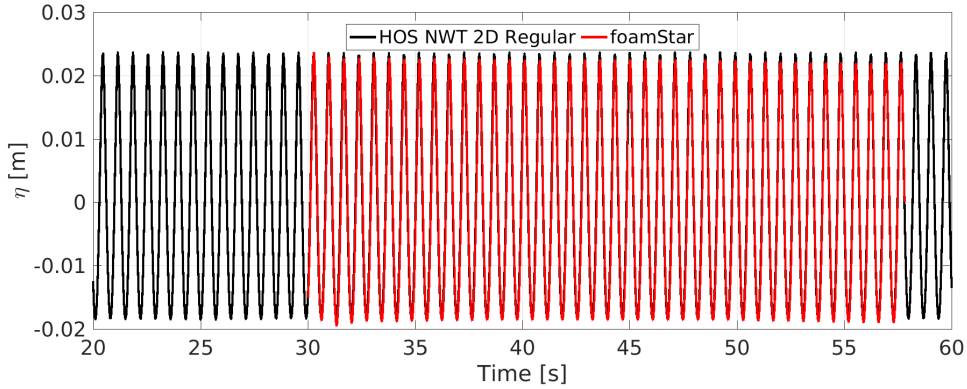
Figure 5.1. Comparison of HOS-Ocean wave elevation

(a) HOS-Ocean 2D Irregular Waves

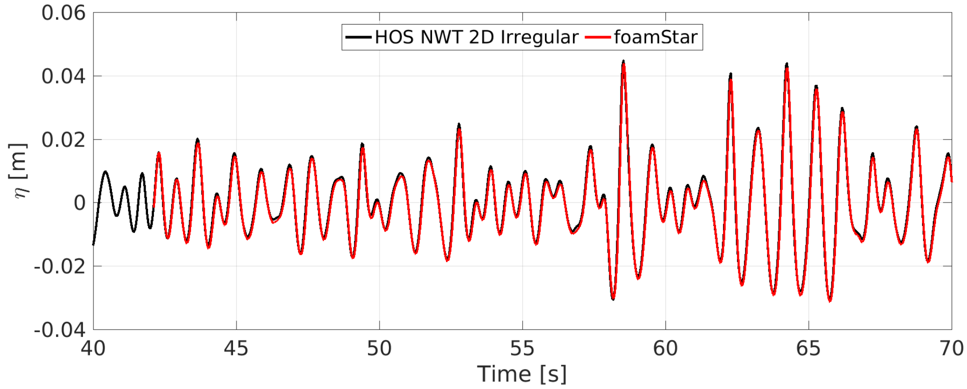


(b) HOS-Ocean 3D Irregular Waves

Figure 5.2. Snapshot of HOS-Ocean wave fields by `foamStar`



(a) HOS-NWT 2D Regular Waves



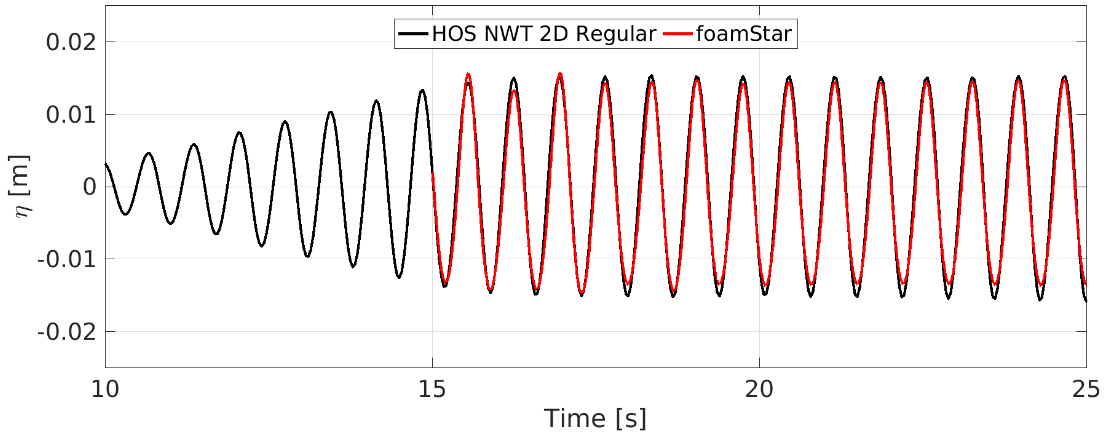
(b) HOS-NWT 2D Irregular Waves

Figure 5.3. Comparison of HOS-NWT 2D wave elevation

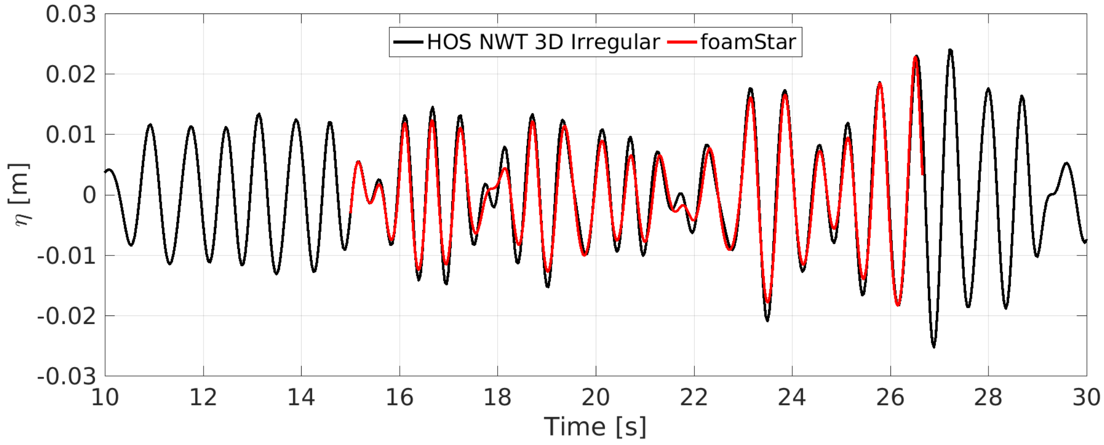
(a) HOS-NWT 2D Irregular Waves

(b) HOS-NWT 2D Irregular Waves

Figure 5.4. Snapshot of HOS-NWT 2D wave fields by `foamStar`

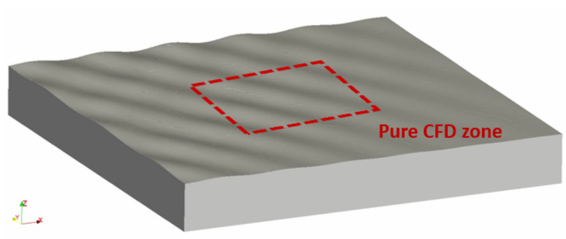


(a) HOS-NWT 3D Regular Waves

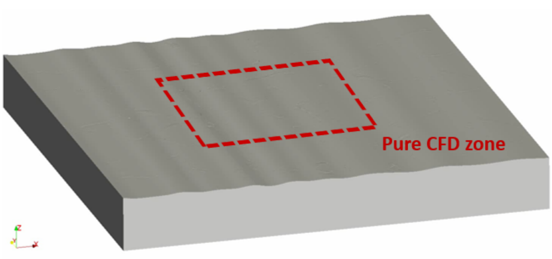


(b) HOS-NWT 3D Irregular Waves

Figure 5.5. Comparison of HOS-NWT 3D wave elevation



(a) HOS-NWT 3D Regular Waves



(b) HOS-NWT 3D Irregular Waves

Figure 5.6. Snapshot of HOS-NWT 3D wave fields by `foamStar`

Simulation of extreme waves (1000 year return period waves in Gulf of Mexico(GOM) (Condition : $H_s = 17.5m, T_p = 15.5s, \gamma = 3.3$) is simulated with HOS-NWT for 2D cases and used to generate waves in `foamStar`. The waves are compared with experiments performed in the wave basin of Ecole Centrale de Nantes (ECN). To simulate nonlinear breaking waves, the wave breaking model in HOS is utilized and allows also to capture when wave breaking occur. The expected wave breaking events are shown in Fig. 5.7. In the experiments, wave breaking is observed at the expected position and time by HOS wave theory. The breaking moment in the experiment is shown in Fig. 5.8. The time series of wave elevation measured in the experiments are compared with the results of simulation using `Grid2Grid` in Fig 5.9. The simulation snapshot when wave breaking occur is shown in Fig. 5.10. The small disturbance at the wave front is observed.

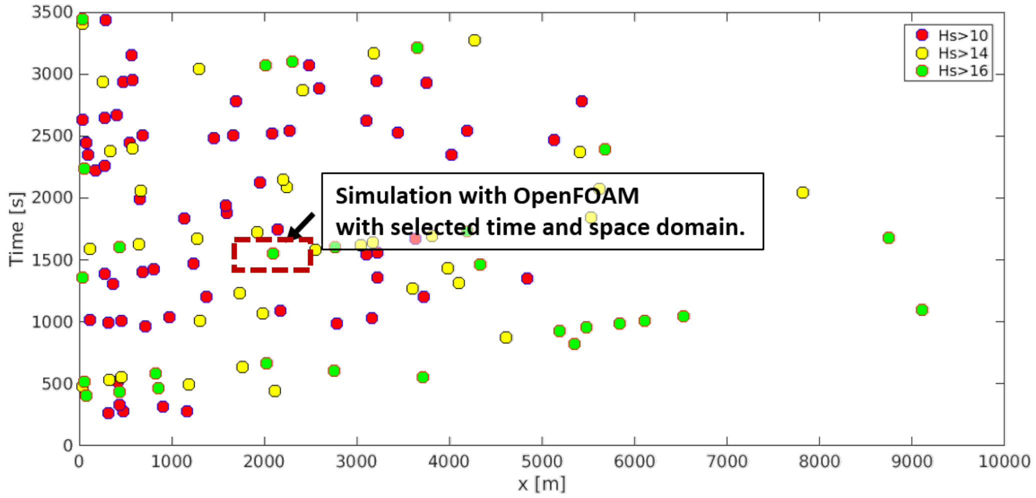


Figure 5.7. Expected wave breaking by HOS

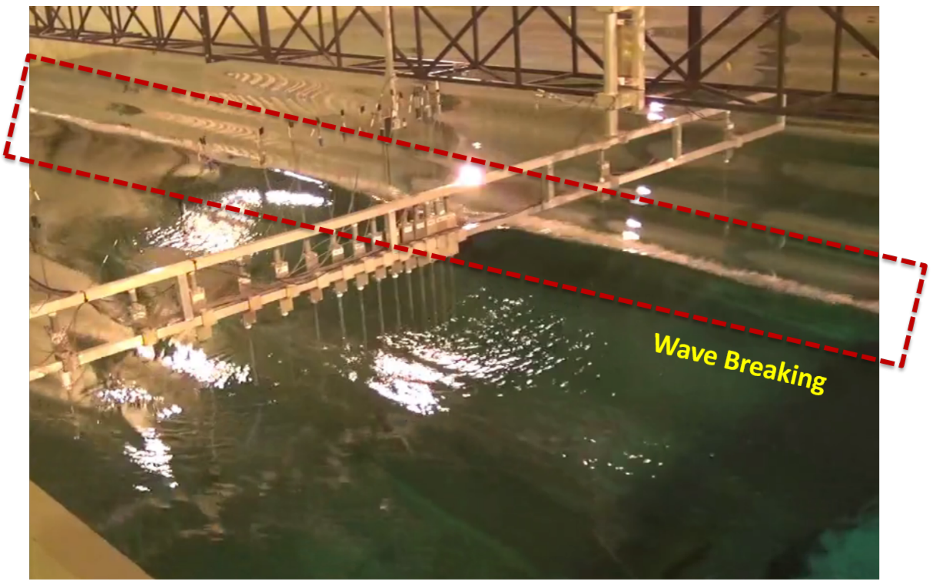
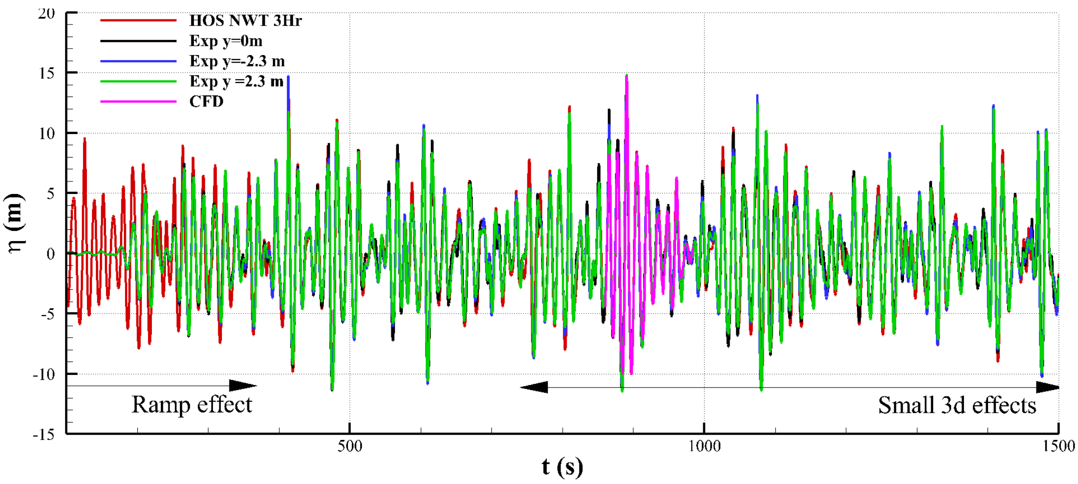
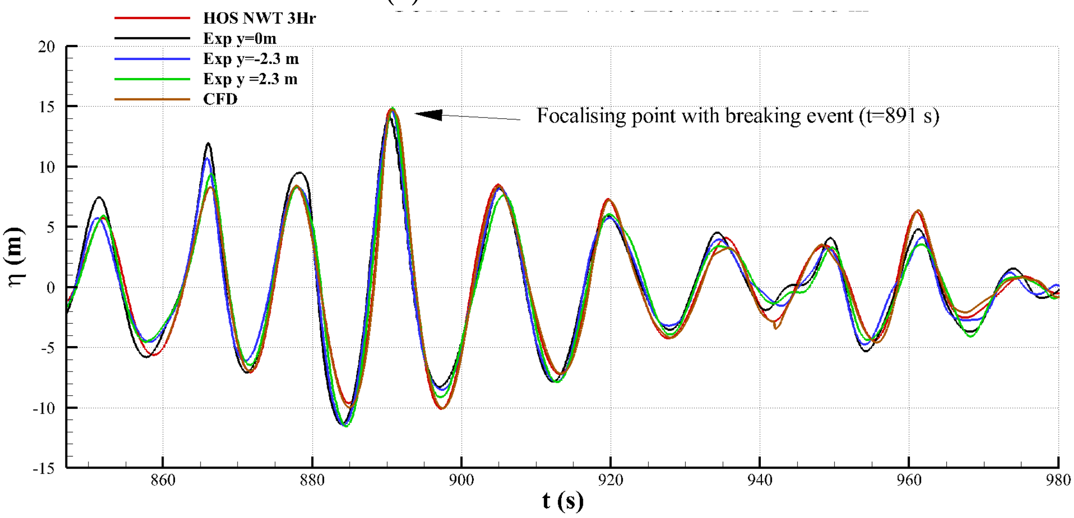


Figure 5.8. Observed wave breaking in experiment



(a) Overall wave time series



(b) Wave elevation at the wave breaking moment

Figure 5.9. Snapshot of HOS-NWT 3D wave fields by `foamStar`

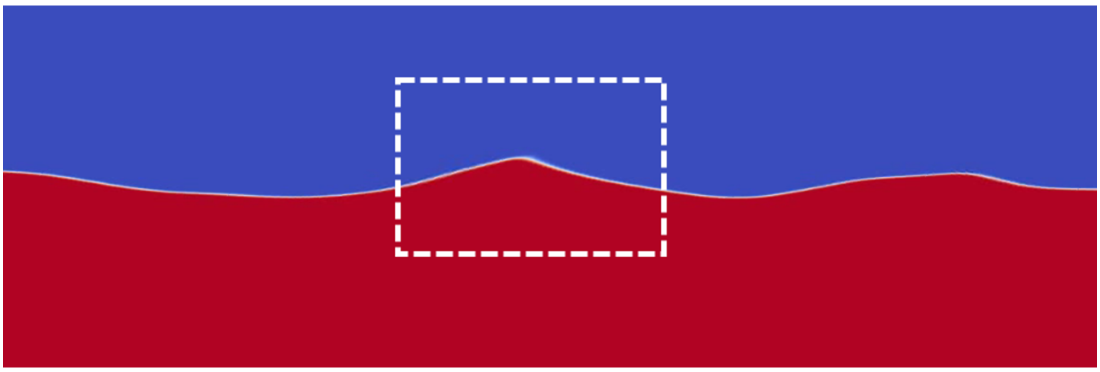


Figure 5.10. Simulation of wave breaking by `foamStar` and `Grid2Grid`

6 Summary

HOS wrapper program called `Grid2Grid` is developed for the nonlinear wave simulation of numerical solvers. Most of data and functionality is encapsuled as a class to be easily used and maintained. `Grid2Grid` generates dynamic linked library as an independent package to be called and easily used in other languages.

The post processing of HOS is also possible by using `Grid2Grid`. Included post processing program is called `postGrid2Grid`. The usage is explained in Chapter 2.4.

`Grid2Grid` is validated by using the code `foamStar` based on standard multiphase solver of OpenFOAM and also with an experiment. In the experiment, waves corresponding to 1000 year return period in Gulf of Mexico are generated. The wave elevation is measured at the wave breaking position expected by HOS wave theory and compared with the results of the simulation. Good agreement is shown between the measurement and the numerical solutions and the nonlinear wave phenomenon is observed both in experiment and in simulation.

In this document, the `Grid2Grid` program architecture, class and module structure, principle class data and functionality are explained to understand the feature of `Grid2Grid` and to be easily applied to numerical solvers. The interface examples with other programming languages are given as a source code and also in `Grid2Grid` package.

References

Open-source release of hos-nwt. `https://github.com/LHEEA/HOS-NWT`. Accessed: 2017-11-17.

Open-source release of hos-ocean. `https://github.com/LHEEA/HOS-ocean`. Accessed: 2017-11-17.

Bonnefoy, F., Ducrozet, G., Touzé, D. L., and Ferrant, P. (2011). *Time Domain Simulation of Nonlinear Water Waves using Spectral Methods*, pages 129–164. World Scientific.

Ducrozet, G., Bonnefoy, F., Le Touzé, D., and Ferrant, P. (2007). 3-d hos simulations of extreme waves in open seas. *Natural Hazards and Earth System Sciences*, 7(1):109–122.

Ducrozet, G., Bonnefoy, F., Touzé, D. L., and Ferrant, P. (2012). A modified high-order spectral method for wavemaker modeling in a numerical wave tank. *European Journal of Mechanics - B/Fluids*, 34(Supplement C):19 – 34.

Ducrozet, G., Bonnefoy, F., Touzé, D. L., and Ferrant, P. (2016). Hos-ocean: Open-source solver for nonlinear waves in open ocean based on high-order spectral method. *Computer Physics Communications*, 203(Supplement C):245 – 254.

Monroy, C., Seng, S., and Malenica, S. (2017). Développements et validation de l'outil CFD OpenFOAM pour le calcul de tenue à la mer. 15émes Journées de l'hydrodynamique. French.

Seng, S. (2012). *Slamming and Whipping Analysis of Ships*. PhD thesis, Technical University of Denmark, Lyngby, Denmark.

Williams, J. (2015). Open-source release of multidimensional b-spline written in fortran. `https://github.com/jacobwilliams/bspline-fortran`. Accessed: 2017-11-17.