**BouHybrid User’s Manual**

Dongbin He & Kezhao Fang

**The State Key Laboratory of Coastal and Offshore Engineering**

**Dalian University of Technology**

**Dalian, Liaoning, China**

**Contents**

[**1.** **Boussinesq Equations** 4](#_Toc460056193)

[**1.1** **Governing equations** 4](#_Toc460056194)

[**1.2** **Conservative form of the governing equations** 4](#_Toc460056195)

[**2.** **Numerical schemes** 6](#_Toc460056196)

[**2.1** **Spatial discretization** 6](#_Toc460056197)

[**2.2** **Time Stepping** 7](#_Toc460056198)

[**2.3** **Numerical treatment of wet-drying and wave breaking** 7](#_Toc460056199)

[**2.4** **Boundary conditions and wavemaker** 8](#_Toc460056200)

[**3.** **Code description** 10](#_Toc460056201)

[**3.1** **Model flow chart** 10](#_Toc460056202)

[**3.2** **Subroutine and function descriptions** 11](#_Toc460056203)

[**3.3** **Main global variables in the code** 11](#_Toc460056204)

[**4.** **Installation and compilation** 13](#_Toc460056205)

[**5.** **Input settings** 14](#_Toc460056206)

[**6.** **Examples** 19](#_Toc460056207)

[**7.** **References** 20](#_Toc460056208)

**Introduction**

BouHybrid is a two-dimensional shock-capturing wave model based on Boussinesq-type equations, which is suitable for simulating wave propagation and associated hydrodynamics in the nearshore region.

BouHybrid basically follows the FUNWAVE-TVD but some improved are made. The governing equations, in the form of depth-averaged velocities, are used. The numerical schemes used to get the numerical flux at the interface is MUSTA, which is Riemann problem free and easy for implementation. The treatment to wave breaking, bottom friction and moving shoreline are improved.

We welcome suggestions for improvement and criticism for mistakes.

Please contact us: [kfang@dlut.edu.cn](mailto:kfang@dlut.edu.cn); [hdb\_1992@mail.dlut.edu.cn](mailto:hdb_1992@mail.dlut.edu.cn).

1. **Boussinesq Equations**
   1. **Governing equations**

The governing equations used in this model is derived by Zou and Fang (2008) which is also a variant form of the enhanced GN equations. The governing equations include one continuity equation and two momentum equations in the horizontal directions, which can be written as





Where the subscript *t* denotes the partial derivative with respect to time,  is the two-dimensional partial differential operator,  is the depth averaged velocity, *η* is the free surface elevation, *h* is the still water depth, *d=h + η* is the total water depth, and *g* is the gravity acceleration constant. ***P*** is the dispersive term and defined as



Where  Following Madsen and Schäffer (1998), Zou and Fang (2008) added the following expressions to the left-hand side of Equation (2) to further enhance the dispersion and shoaling of the equations





Here the parameters  are used to optimize linear dispersion and shoaling, respectively. In the present study,  are used. If we set , the governing equations recover to the GN equations.

* 1. **Conservative form of the governing equations**

The governing equations are rewritten in the conservative form as



Where ***U*** is a vector variable, ***F*** and ***G*** are vector fluxes along *x* and *y* direction, respectively, and they are expressed as





with





where *p=du* and *q=dv*. The new variables water level  and bed elevation *zb* are introduced herein for the convenient implementation of the well-balanced scheme proposed by Wang, Liang, Kesserwani, and Hall (2012) *et al*. ***S*** in equation (6) is the source vector and can be grouped in three parts, namely, bottom slope term ***S****b*, bottom friction term ***S****f,* and dispersive term ***S****d*



where superscripts *x* and *y* denote the vector component in the *x* and *y* direction, respectively, and



where *f* in Equation (11) is the bottom friction coefficient. The dispersive terms are given by





1. **Numerical schemes**
   1. **Spatial discretization**

A combined cell-centered finite-volume (FV) and finite-difference (FD) method was applied to spatial discretization. If the computation domain is discretized into finite rectangular cells indexed as(*M* and *N* are the grid numbers in *x* and *y* directions respectively, whereasare the grid sizes), the resulting updating formula reads



where the superscript *n* denotes time level, is the time step, is redefined as the average variables of a cell volume. present that fluxes through the left- and right-cell interfaces, whileare the fluxes through south and north cell interfaces, respectively. *Si,j* is the numerical source for cell(*i,j*).

For the flux terms, a high-order FV method was adopted in the present model, while finite-difference method for source terms. FD method is straightforward and not repeated here.

Fig. 1 The sketch of MUSTA scheme (superscript k denotes the value after kth stage)

**

**





t



***s***

In the present model, the numerical fluxes are computed using Multi-stage (MUSTA) scheme In MUSTA approach, the Riemann problem(with the sample of 1D, along *x* direction) is transformed into an independent so-called MUSTA mesh in the *s–τ* plane as depicted in Fig. 1, where *s* denotes the spatial variable, associated with *x*, and *τ* denotes the temporal variable, associated with *t*. Note that cells *i* and *i*+1 correspond respectively to cells 0 and 1 on the MUSTA mesh, so that the inter-cell position on *i+*1/2 corresponds to the interface 1/2. In the MUSTA flux approach, we first evolve in time *τ* the datasets with increment , for a number of stages, utilizing a predictor flux function. In this manner, starting with , after *k* stages we have the evolved data. The sought numerical flux is then computed by using a corrector flux function as. By settingand using FORCE flux for both the predictor and the corrector steps, we have the following numerical algorithm to calculate the MUSTA flux.

**Step 1**: Flux evaluation



Stop if the prescribed number of states *K* has been reached set

**Step 2:** Opening of Riemann fan



**Step 3:** Go to Step 1.

Hereafter the one – (*K*=1), two – (*K*=2) and three – (*K*=3) stages of the MUSTA scheme will be referenced as MUSTA-1, MUSTA-2 and MUSTA-3. In theory, the larger the number of stages, the closer the resulting numerical flux is to the Godunov flux of the exact Riemann solver. Nevertheless, the choice *K*=1 is recommended for practical applications.

The above process only provides a first order numerical flux, which is generally not sufficiently accurate for approximating the convective flux embodied in Boussinesq formulations. Following, the fourth order MUSCL-TVD scheme is used to reconstruct the leftand rightstate at cell interface *i*+1/2, with the use of the Minmod limiter function. The MUSTA scheme is then initiated by replacing the constant statesandwith the reconstructed valuesand

* 1. **Time Stepping**

The third-order Runge\_Kutta scheme with strong stability preserving property was applied for time stepping. The scheme is given by



where superscripts (1) and (2) denote the corresponding quantities at intermediate in the Runge-Kutta integration.

For numerical stability requirement, the time increment  must be restricted by the CFL condition as



where  is the Courant number.

* 1. **Numerical treatment of wet-drying and wave breaking**

In the present model, we used the hydrostatic construction technique proposed by Wang *et al*. (2012) to treat wet-drying boundaries, which is summarized as follows.

**Step 1.** The first step is linear construction of Riemann states at each cell interface from independent variables (i.e. p, q, f, d) using a MUSCL method. The velocity components are then obtained:



where *L* and *R* denote the left- and right-state along cell interface, and (*i, j*) is the cell center index. The velocities are directly equal to zero if m (the threshold for defining a dry cell in the present study). The face values at the right-hand side of (*i*+1/2, *j*) are calculated in a similar way. The variables in a dry cell or a wet cell adjacent to dry cells are assumed to have a constant distribution.

**Step 2.** The second step is finding the single face value of bed elevation and nonnegative construction of water depth. This is implemented as



where , As a result of changing *zb*, the released quantities are redefined as



The above reconstruction guarantees the nonnegative water depth, and a similar procedure is applied to calculate the values on the right-hand side of (*i*+1/2, *j*).

**Step 3.** The third step is correcting the face value of bed elevation at wet–dry fronts to satisfy the extended *C*-property. This is achieved first by



and then by



The Riemann states obtained in Equations (20)-(23) are then employed to compute the interface flux using the MUSTA scheme. To keep consistency with the flux calculation, the bottom slope term is calculated by a central difference scheme at all cell centers. In the *x* direction, for instance



We proposed a combination of two criteria for triggering wave breaking modeling within our shock-capturing scheme, namely, the ration of wave height to water depth criterion(*e.g.*, Fang *et al*., 2013; Shi *et al*., 2012; Tonelli and Petti, 2011) and the local slope angle criterion(is the critical front face angle at the initiation of breaking; see Kazolea, Delis, and Synolakis, 2014). Local wave slope criterion is introduced herein to consider steady jumps, which cannot be captured by the first criterion since (Kazolea, Delis and Synolakis, 2014)). In the computation, the ratio of wave height to water depthand the local wave angle  are computed at each computation cell. If at least one of the criteria is satisfied, the cells are labeled as breaking cells, and dispersive terms are deactivated. Dispersive terms can only be reactivated oncereduces to below a certain threshold, *i.e.*, this ensures the stable computation(Tonelli and Petti, 2011) and is quite necessary for fully nonlinear Boussinesq equations, otherwise instability tends to occur due to sharply deactivating or reactivating dispersive terms.

* 1. **Boundary conditions and wavemaker**

In addition to the aforementioned moving shoreline, other type boundary conditions are imposed by setting another three ghost cells around the entire computation domain (Fang *et al*., 2013). For the reflective wall, tangential velocity and normal velocity on ghost cells are determined from the inner domain by imposing symmetric and antisymmetric conditions with respect to the solid wall. Besides, a sponge layer is also placed at the front of solid walls to absorb wave energy once necessary.

Wavemaker implemented in this study include Wei and Kirby’s (1999) internal wavemakers for regular waves and irregular waves. For the irregular wavemaker, an extension was made to incorporate an alongshore periodicity into wave generation, in order to eliminate a boundary effect on wave simulations. The technique exactly follows the strategy in Chen *et al*.(2003), who adjusted the distribution of wave directions in each frequency bin to obtain alongshore periodicity. This approach is effective in modeling of breaking wave-induced nearshore circulation such as alongshore currents and rip currents.

1. **Code description**

The code of BouHybrid is basically based on the FUNWAVE-TVD (Shi *et al*., 2012), reference to FUNWAVE-TVD is recommended.

* 1. **Model flow chart**

The Flow chart is show in Fig. 2.

Fig. 2 Flow chart of the main program

* 1. **Subroutine and function descriptions**

Ongoing…

* 1. **Main global variables in the code**

Depth(): still water depth h at element point

DepthNode(): still water depth h at node

DepthX(): still water depth h at x-interface

DepthY(): still water depth h at y-interface

Eta(): surface elevation, for dry point, Eta() = MinDepth - Depth(), MinDepth is specified in

input.txt.

Eta0(): η at previous time level

MASK(): 1 - wet, 0 - dry

MASK STRUC(): 0 - permanent dry point

MASK9: mask to switch from Boussinesq equation to SWE, 1 - Boussinesq, 0 - SWE

U(): depth-averaged u or u at the reference level (uα) at element

V(): depth-averaged v or v at the reference level (vα) at element

HU(): (h + η)u at element

HV(): (h + η)v at element

P(): (h + η)(u + U4) at x-interface

Q(): (h + η)(v + V4) at y-interface

Fx(): numerical flux F at x-interface

Fy(): numerical flux F at y-interface

Gx(): numerical flux G at x-interface

Gy(): numerical flux G at y-interface

Ubar(): U

Vbar(): V

EtaRxL(): η Left value at x-interface

EtaRxR(): η Right value at x-interface

EtaRyL(): η Left value at y-interface

EtaRyR(): η Right value at y-interface

HxL(): total depth Left value at x-interface

HxR(): total depth Right value at x-interface

HyL(): total depth Left value at y-interface

HyR(): total depth Right value at y-interface

HUxL(): (h + η)u Left value at x-interface

HUxR(): (h + η)u Right value at x-interface

HVyL(): (h + η)v Left value at y-interface

HVyR(): (h + η)v Right value at y-interface

PL(): (h + η)(u + U4), Left value at x-interface

PR(): (h + η)(u + U4), Right value at x-interface

QL(): (h + η)(v + V4), Left value at y-interface

QR(): (h + η)(v + V4), Right value at y-interface

FxL = HUxL\*UxL + 1/2\*g\*(EtaRxL2 + 2\*EtaRxL\*Depthx)

FxR = HUxR\*UxR + 1/2\*g\*(EtaRxR2 + 2\*EtaRxR\*Depthx)

FyL = HyL\*UyL\*VyL

FyR = HyR\*UyR\*VyR

GxL = HxL\*UxL\*VxL

GxR = HxR\*UxR\*VxR

GyL = HVyL\*VyL + 1/2\*g\*(EtaRyL2 + 2\*EtaRyL\*Depthy)

GyR = HVyR\*VyR + 1/2\*g\*(EtaRyR2 + 2\*EtaRyR\*Depthy)

1. **Installation and compilation**

BouHybrid is distributed in a compressed file. To install the programs, first, unzip the package. The source code was written using Fortran 90 with Compaq Visual Fortran 6.6. For users, running the executable file named BouHybrid.exe under the release folder is the simplest way( NOTE: BouHybrid was developed with Windows XP. which means that there may be compatibility issues if BouHybrid works in other operating systems including higher versions of Windows), and users need input files ready for their cases. Besides, we used Compaq Array Visualizer (CAV) in BouHybrid to view and analyze array data graphically, which requires the installation of CAV. Of course, users can choose to switch that function off. Both CVF and CAV are not distributed with BouHybrid, and users can get them from Internet.

We also provide users source codes (main.f90, mod\_global.f90, mod\_param.f90, mod\_util.f90) to compile and run. however, we still recommend users compile these files under CVF.

1. **Input settings**

Following are descriptions of parameters in input.txt (NOTE: all parameter names are capital sensitive)

**PROJECTNAME:** name of your project, used for log file and project folder.

**SPECIFICATION OF EQUATION**

EqID: governing equation

EqID=23: GN equation.

EqID=1: nonlinear shallow water equation.

**SPECIFICATION OF DIMENSION**

Mglob: global dimension indirection.

Nglob: global dimension indirection.

SPECIFICATION OF GRID SIZE

DX: grid size(m) in  direction.

DY: grid size(m) in  direction.

**SPECIFICATION OF TIME**

TOTAL\_TIME: simulation time in seconds

PLOT\_INTV\_SNAPSHOT: output interval in seconds (Note, output time is not exact because adaptive dt is used.)

PLOT\_INTV\_GAGE: time interval (s) of gauge output.

SCREEN\_INTV: time interval (s) of screen print.

**SPECIFICATION OF BATHYMETRY**

DEPTH\_TYPE: depth input type.

DEPTH\_TYPE=DATA: from a depth file (depth.txt).

The program includes several simple bathymetry configurations such as

DEPTH\_TYPE=FLAT: flat bottom, need DEPTH\_FLAT.

DEPTH\_TYPE=SLOPE: plane beach along  direction. It needs three parameters: slope, SLP, slope starting point, Xslp and depth of flat part, DEPTH\_FLAT.

DEPTH\_FILE: bathymetry file if DEPTH\_TYPE=DATA, the file name must be ‘depth.txt’.

1. For 1D case, users should provide the coordinates of control nodes in file ‘depth.txt’. As shown in Fig. 3, A~F is called control nodes

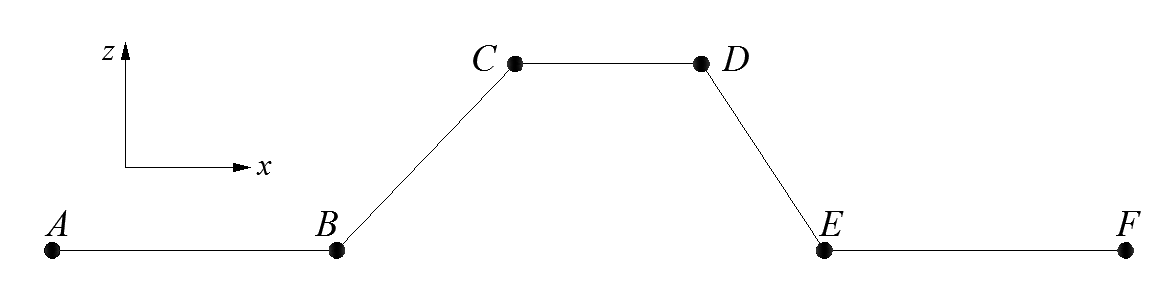


Fig 3. Example

2) For 2D case, file dimension should be  with the first point as the south-west corner. The read format in the code is shown below.

*DO J=1, Nglob*

*READ(1,\*) (Depth(I,J), I=1,Mglob)*

*ENDDO*

DEPTH\_FLAT: water depth of flat bottom if DEPTH\_TYPE=FLAT or DEPTH\_TYPE=SLOPE (flat part of plane beach).

SLP： slope if DEPTH\_TYPE=SLOPE

Xslp: starting (m) of a slope, if DEPTH\_TYPE=SLOPE

**SPECIFICATION OF INITIAL CONDITION**

INT\_UVZ: logical parameter for initial condition, default is FALSE.

INI\_U: logical parameter for initial , default is FALSE, the name of file must be U.txt, data format is the same as depth data.

INI\_V: logical parameter for initial , default is FALSE, the name of file must be V.txt, data format is the same as depth data.

INI\_Z: logical parameter for initial , default is FALSE, the name of file must be Z.txt, data format is the same as depth data.

**SPECIFICATION OF WAVEMAKER**

WAVEMAKER: wavemaker type.

WAVEMAKER=LEF\_SOL: left boundary solitary, need AMP, DEP and LAGTIME.

WAVEMAKER=INI\_SOL: initial solitary wave, WKN B solution, need AMP, DEP and XWAVEMAKER.

WAVEMAKER=INI\_REC: initial rectangular hump, need XC, YC and WID.

WAVEMAKER=WK\_REG: Wei and Kirby 1999 internal wave maker, need XC\_WK, Tperiod,, AMP\_WK, DEP\_WK, Theta\_WK, and Time\_ramp(factor of period).

WAVEMAKER=WK\_IRR: Wei and Kirby 1999 TMA spectrum wavemaker, need XC\_WK, DEP\_WK, Time\_ramp, Delta\_WK, FreqPeak, FreqMin, FreqMax, Hmo, GammaTMA, ThetaPeak

WAVEMAKER=WK\_TIME\_SERIES: fft time series to get each wave component and then use Wei and Kirby’s (1999) wavemaker. Need input WaveCompFile (including 3 columns: per,amp,pha) and NumWaveComp,PeakPeriod,DEP\_WK,Xc\_WK,Ywidth\_WK.

WAVEMAKER=WK\_SRC\_SERIES: only used once time series of source strength is given

AMP: amplitude (m) of initial , if WAVEMAKER = INI REC, WAVEMAKER = INI SOL,

WAVEMAKER = LEF SOL.

DEP: water depth at wavemaker location, if WAVEMAKER = INI SOL, WAVEMAKER = LEF SOL.

LAGTIME, time lag (s) for the solitary wave generated on the left boundary, e.g., WAVEMAKER= LEF SOL.

XWAVEMAKER:  (m) coordinate for WAVEMAKER = INI SOL.

Xc:  (m) coordinate of the center of a rectangular hump if WAVEMAKER = INI REC.

Yc:  (m) coordinate of the center of a rectangular hump if WAVEMAKER = INI REC.

WID: width (m) of a rectangular hump if WAVEMAKER = INI REC.

Time ramp: time ramp (s) for Wei and Kirby (1999) wavemaker.

Delta WK: width parameter δ for Wei and Kirby (1999) wavemaker. δ = 0.3 ∼ 0.6

DEP WK: water depth (m) for Wei and Kirby (1999) wavemaker.

Tperiod: period (s) of regular wave for Wei and Kirby (1999) wavemaker.

AMP WK: amplitude (m) of regular wave for Wei and Kirby (1999) wavemaker.

Theta WK: direction (degrees) of regular wave for Wei and Kirby (1999) wavemaker. Note: it

may be adjusted for a periodic boundary case by the program. A warning will be given if

adjustment is made.

FreqPeak: peak frequency (1/s) for Wei and Kirby (1999) irregular wavemaker.

FreqMin: low frequency cutoff (1/s) for Wei and Kirby (1999) irregular wavemaker.

FreqMax: high frequency cutoff (1/s) for Wei and Kirby (1999) irregular wavemaker.

Hmo: Hmo (m) for Wei and Kirby (1999) irregular wavemaker.

GammaTMA, TMA parameter *γ* for Wei and Kirby (1999) irregular wavemaker.

ThetaPeak: peak direction (degrees) for Wei and Kirby (1999) irregular wavemaker.

Sigma\_Theta: parameter of directional spectrum for Wei and Kirby (1999) irregular wavemaker.

**SPECIFICATION OF PERIODIC BOUNDARY CONDITION**

(Note: only south-north periodic condition was implemented)

PERIODIC: logical parameter for periodic boundary condition, T-periodic, F-wall boundary condition.

**SPECIFICATION OF SPONGE LAYER**

SPONGE\_ON: logical parameter, T – sponge layer, F – no sponge layer.

Sponge west width: width (m) of sponge layer at west boundary.

Sponge east width: width (m) of sponge layer at east boundary.

Sponge south width: width (m) of sponge layer at south boundary.

Sponge north width width (m) of sponge layer at north boundary

R sponge: decay rate in sponge layer. Its values are between 0.85 *∼* 0.95.

A sponge: maximum damping magnitude. The value is ∼ 5.0.

**SPECIFICATION OF PHYSICS**

DISPERSION: logical parameter for inclusion of dispersion terms. T – calculate dispersion, F-no dispersion terms

Gamma1: parameter for linear dispersive terms. 1.0 – inclusion of linear dispersive terms, 0.0 – no linear dispersive terms.

Gamma2: parameter for nonlinear dispersive terms. 1.0 – inclusion of nonlinear dispersive terms, 0.0 – no nonlinear dispersive terms.

Gamma1=1.0, Gamma2=1.0 for the fully nonlinear Boussinesq equations.

Gamma3: parameter for nonlinear shallow water equations(Gamma3=1.0).

Cd: bottom friction

**SPECIFICATION OF NUMERICS**

FLUX\_TYPE: options for numerical flux, HLL(c), MUSTA or CENTRAL

MUSCL\_ORDER: options for flux reconstruction, SECOND for the second-order, FOURTH for the fourth-order

LIMITTER\_YPE: options for flux limiter, MINMOD and VANLEE

Regularization: logical parameter, using regularization method to treat velocity (T) or not (F)

SHORELINE\_SKILL=HYDROSTATIC: only option for wet-drying treatment,

Time\_Iteration: stepping option, SSPRK2 for the second-order Strong Stability-Preserving(SSP) Runge-Kutta scheme, SSPRK3 for the third-order Strong Stability-Preserving(SSP) Runge-Kutta scheme

CFL: CFL number, CFL~0.25

**SPECIFICATION OF TREAT WAVE BREAKING**

SHOCK\_TYPE: options for wave breaking.

SHOCK\_TYPE=NONE: no wave breaking

SHOCK\_TYPE=RULE1: local hybrid epsilon() and slope()

SHOCK\_TYPE=RULE2: local hybrid  and slope()

SHOCK\_TYPE=RULE3: empirical eddy viscosity term is applied

GAMMA\_1: critical value for initiation of wave breaking

GAMMA\_2: critical value for stopping of wave breaking

ApplyFroude: logical parameter for consideration of Froude number, ApplyFroude=T, wave breaking is modeled by switching from Boussinesq to NSWE at cells where the Froude numbers exceeds a certain threshold, the default is F.

FroudeCap: cap for Froude number in velocity calculation for efficiency. The value could be 5~10.0

**SPECIFICATION OF WET\_DRY**

MinDepth: minimum water depth(m) for wetting and drying scheme. Suggestion: MinDepth=0.001 for lab scale and 0.01 for field scale

MinDepthFrc: minimum water depth(m) to limit bottom friction value. Suggestion: MinDepthFrc =0.01 for lab scale and 0.1 for field scale.

**SPECIFICATION OF MIXING**

MIXING: logical parameter for smagorinsky mixing, T of FALSE

T\_INTV\_mean: averaging time interval(s), the default is 20.0

C\_smg: smagorinsky coefficient.

**SPECIFICATION OF OUTPUT**

NumberStations: number of station for output. If NumberStations > 0, need input i, j in stat.txt

ETASCALE: scale factor for output *η*, only used in gauge output, the default is 1.0

USCALE: scale factor for output *u*, only used in gauge output, the default is 1.0

VSCALE: scale factor for output *v*, only used in gauge output, the default is 1.0

TSCALE: scale factor for output *t*, only used in gauge output, the default is 3600.0

TOFFSET: offset for output *t*, only used in gauge output, the default is 0.0

OUT\_RUNUP: logical parameter for output right-hand runup, only worked for 1D case. T of F.

CAL\_MEAN: logical parameter for output statistic wave height, only worked for 1D case. T or F.

TIMEBEG: start time of eta time series.

TIMEEND: end time of eta time series

HTYPE: options for output statistic wave height

HTYPE=1: average height

HTYPE=2: the difference between maximum and minimum of eta

HTYPE=3: significant wave height

DEPTH OUT: logical parameter for output depth. T or F.

U: logical parameter for output *u*. T or F.

V: logical parameter for output *v*. T or F.

ETA: logical parameter for output *η*. T or F.

BED: logical parameter for output bed elevation. T of F.

HMAX: logical parameter for output of recorded maximum surface elevation . T or F.

Umean: logical parameter for output mean *u*, T of F

Vmean: logical parameter for output mean *v*, T of F

ETAmean: logical parameter for output mean *η*, T of F

MASK: logical parameter for output wetting-drying MASK. T or F.

MASK9: logical parameter for output MASK9 (switch for Boussinesq/NSWE). T or F.

SXL: logical parameter for output of wave speed of left value at *x*-interface. T of F.

SXR: logical parameter for outoput of wave speed of right value at *x*-interface. T of F.

SYL: logical parameter for outoput of wave speed of left value at *y*-interface. T of F.

SYR: logical parameter for outoput of wave speed of right value at *y*-interface. T of F.

SourceX: logical parameter for output source terms in *x* direction. T or F.

SourceY: logical parameter for output source terms in *y* direction. T or F.

P: logical parameter for output of momentum flux in *x* direction. T or F.

Q: logical parameter for output of momentum flux in *y* direction. T or F.

Fx: logical parameter for output of numerical flux F in *x* direction. T or F.

Fy: logical parameter for output of numerical flux F in *y* direction. T or F.

Gx: logical parameter for output of numerical flux G in *x* direction. T or F.

Gy: logical parameter for output of numerical flux G in *y* direction. T or F.

1. **Examples**
   1. **Sinusoidal wave propagation over a submerged bar**

This case was provided by Profs. Gonzalo. The bathymetry configuration of the experiment is shown in Fig. 4. Sponge layers are placed at the left and right end to absorb the wave energy. Sinusoidal waves with amplitude 1.0*cm* and period 4*s* are incident at *x* = 15*m*. The grid size is 

Some important parameters in the input file, input.txt, are

! --------------------EQUATION------------------------------

EqID=23

DimID=1

! ------------------DIMENSION-------------------------------

Mglob = 700

Nglob = 1

! ----------------- GRID------------------------------------

DX = 0.1

DY = 0.1

! ----------------- TIME------------------------------------

TOTAL\_TIME = 1000000.0

PLOT\_INTV\_SNAPSHOT = 1.0

PLOT\_INTV\_GAGE = 0.01

SCREEN\_INTV = 0.1

! ----------------- BATHYMETRY------------------------------

DEPTH\_TYPE = DATA

DEP\_REF=0.35

! ----------------WAVEMAKER---------------------------------

WAVEMAKER = WK\_REG

DEP\_WK = 0.35

!=Wei and Kirby 1999

Time\_ramp = 2.0

Delta\_WK = 0.75 ! widthparameter 0.3-0.6

Xc\_WK = 15.0

Ywidth\_WK = 10000.0

!=Wei and Kirby regular wave

Tperiod = 4.0

AMP\_WK = 0.01

Theta\_WK = 0.0

!----------------SPONGE LAYER ------------------------------

SPONGE\_ON = T

Sponge\_west\_width = 15.0

Sponge\_east\_width = 15.0

Sponge\_south\_width = 0.0

Sponge\_north\_width = 0.0

R\_sponge = 0.95

A\_sponge = 5.0

!----------------PHYSICS------------------------------------

DISPERSION =T

Gamma1 = 1.0

Gamma2 = 1.0

Gamma3 = 1.0

Cd = 0.

!----------------NUMERICS-----------------------------------

FLUX\_TYPE=MUSTA

MUSCL\_ORDER = SECOND

LIMITTER\_YPE=MINMOD

Regularization=F

SHORELINE\_SKILL =HYDROSTATIC

Time\_Itegration=SSPRK2

CFL = 0.25

!----------------TREAT WAVE BREAKING------------------------

SHOCK\_TYPE= RULE1

GAMMA\_1=0.8

GAMMA\_2=0.3

!--------------WET-DRY--------------------------------------

MinDepth=0.0000001

MinDepthFrc = 0.0

!-----------------OUTPUT------------------------------------

DEPTH\_OUT = T

ETA =T

BED=T

1. **References**

Chen, Q., Kirby, J. T.*, et al.* (2003). Boussinesq modeling of longshore currents. *Journal of Geophysical Research Oceans, 108*(C11), 156-157.

Fang, K., Zou, Z.*, et al.* (2013). An efficient shock capturing algorithm to the extended Boussinesq wave equations. *Applied Ocean Research, 43*(5), 11-20.

Kazolea, M., Delis, A. I.*, et al.* (2014). Numerical treatment of wave breaking on unstructured finite volume approximations for extended Boussinesq-type equations. *Journal of Computational Physics, 271*(9), 281-305.

Kirby, J. T., Wei, G., Chen, Q., Kennedy, A. B. and Dalrymple, R. A., 1998, “FUNWAVE 1.0,  
Fully nonlinear Boussinesq wave model. Documentation and users manual”. Report CACR-  
98-06, Center for Applied Coastal Research, Department of Civil and Environmental Engineering, University of Delaware.

Madsen, P. A., & Schäffer, H. A. (1998). Higher-order Boussinesq-type equations for surface gravity waves: derivation and analysis. *Philosophical Transactions of the Royal Society A Mathematical Physical & Engineering Sciences, 356*(1749), 3123-3181.

Shi, F., Kirby, J. T.*, et al.* (2012). A high-order adaptive time-stepping TVD solver for Boussinesq modeling of breaking waves and coastal inundation. *Ocean Modelling, s 43–44*(2), 36-51.

Tonelli, M., & Petti, M. (2010). Finite volume scheme for the solution of 2D extended Boussinesq equations in the surf zone. *Ocean Engineering, 37*(7), 567-582.

Wang, Y., Liang, Q.*, et al.* (2012). A 2D shallow flow model for practical dam-break simulation. *Journal of Hydraulic Research, 50*(5), 544-545.

Zou, Z. L., & Fang, K. Z. (2008). Alternative forms of the higher-order Boussinesq equations: Derivations and validations. *Coastal Engineering, 55*(6), 506-521.