User's Guide to the Nueces Delta Hydrodynamic Model v1.0

Andrea J. Ryan, M.S.
Ben R. Hodges, Ph.D.

Center for Research in Water Resources
The University of Texas at Austin

CRWR Online Report 11-8
Submitted to the Coastal Bend Bays and Estuaries Program
Nueces Delta Hydrodynamics Modeling, Contract #1001
October 10, 2011

Acknowledgments

This user manual is part of a project funded by the Coastal Bend Bays and Estuaries Program, with support of the City of Corpus Christi and the U.S. Army Corps of Engineers. Some work related to subgrid-scale roughness in developing the NDHM was developed under National Science Foundation Grant No. 0710901. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or any other sponsor.

Preface

This document provides guidance for version 1.0 of the Nueces Delta Hydrodynamic Model. As of this writing, the model remains under development. The second author is continuing to revise and improve both the underlying model algorithms and the user interface. Comments or questions from model users are welcome. Please contact the second author at hodges@mail.utexas.edu.

Table of Contents

This page intentionally left blank for two-sided printing.

1 Introduction

The Nueces Delta Hydrodynamic Model (NDHM) is a customized implementation of the Predictor-Corrector 2nd order hydrodynamic code (PC2) developed at the Center for Research in Water Resources at the University of Texas at Austin from 2005 to 2008. The numerical algorithms for the model are described in Hodges and Rueda (2008). The model application and testing for the Nueces Delta is described in Ryan and Hodges (2011). This User Manual was developed in support of the Nueces Delta Hydrodynamic Modeling project supported by the Coastal Bend Bays and Estuary Program, the City of Corpus Christi, and the U.S. Army Corps of Engineers.

The PC2 code is written in Matlab, which allows rapid model development and customization. The PC2 implementation for the NDHM is a two-dimensional (2D) solution of the hydrostatic Navier-Stokes equations with conservative mass transport. Running the model requires a computer with Matlab already installed. While the model is running, Matlab must remain open and operating. Generally, Matlab allows only a single model to be run on a computer; however, if the Matlab Parallel Toolbox is installed, up to 8 different copies of the model may be run simulateously on a single multi-core workstation through the Matlab "batch" command.

Installation and use of NDHM requires four steps, corresponding to the following sections of this User Manual.

Installation of PC2 code and NDHM files

Customization of input files

Model execution

Output handling.

This User Manual is based upon PC2 Code v6.0 20110603. Because the code is used and modified extensively for research, later versions of the PC2 Code may have different requirements. The NDHM installation described in this User Manual is NDHMv1.0

This page intentionally left blank for two-sided printing.

2 Installation

2.1 Copying folders and files

The user should create an installation folder on their computer where they want to store and run the model. In this User Manual, we will assume the user has created an installtion folder with the path */usr/NDHM*, although the user is free to install the path into some other folder. Into the installation folder, the user should copy all the folders and files from the installation disk. These should include the PC2 code folder *PC2 Matlab Code v06 2011 06 03* along with folders entitled *Nueces20xxx* and the following files:

```
20080414_salinity.mat
20090410_salinity.mat
20100403_salinity.mat
20100409_salinity.mat
bathymetry_baybathy15_channels.mat
manningsn_landcover_culvert.mat
```

2.2 Folder and file hierarchy

NDHMv1 has the Matlab code for the model stored in the folder *PC2 Matlab Code v06 2011 06 03*. The user should not need to modify any of this code. Installing a new model requires replacing this folder in its entirety.

At the user's installation folder (e.g. */usr/NDHM*), binary Matlab files for initial conditions that are used for multiple file runs are stored. It is planned that later model implementations will use a common subfolder instead of the installation folder.

Other subfolders of */usr/NDHM* provide the input/output data space for individual NDHM model runs. The NDHM model runs completed for the Nueces Delta Hydrodynamic Modeling project have folder names in the form

```
NuecesYYYY_FG_Nd_Rn_Mp
```

where *YYYY* provides the year being modeled, *FG* designates forcing conditions, N provides the number of simulation days, *R* provides the roughness conditions, and M provides the number of added pumps continuously turned on (i.e. pumping that was *not* actually done, but is simulated for testing purposes). Cases examined during the Nueces Delta Hydrodynamic Modeling project as of the date of this project are shown in Table 1. The different roughness cases examined in Ryan (2011) are listed in Table 2.

Table 1. Folder name key

Code	Values used	Notes
YYYY	2008/2009/2010	YEAR of simulation
F	W/S/2W	WIND: normal / still (no wind) / 2x normal wind
G	R/D/hR	RAIN: normal / dry (0 rain) / heavy rain
N	7/14	DAYS: number of days simulated
R	a/b/c/d/e	ROUGHNESS: key for model (see Table 2)
M	0/1/2/3	PUMPS: number of added pumps (0 uses actual pumped inflows)

Table 2. Roughness model key

Roughness ID	Description
a	Original roughness created from land cover matrix
b	10 x roughness *a*
c	100 x roughness *a*
d	2 x roughness *a* for cells with subgrid-scale elevation standard deviation > 20 cm
e	10 x roughness *a* for cells with subgrid-scale elevation standard deviation > 20 cm

In the folder for each model run (i.e. Nueces*YYYY*_*FG*_*N*d_*R*n_*M*p) there are two subfolders (*Input, User_Code*) and a single file *PC2_solver_main.m*. The file *PC2_solver_main.m* is the Matlab top-level script that will run the NDHM model. The folder *Input* contains Matlab functions that are customized for the initial conditions (*Input_xxxx_IC.m*) and boundary conditions (*Input_xxxx_BC.m*) for the model run. These files only need to be changed by the user to run simulations for different times or different initial or boundary *conditions.*

The subfolder *User*_Code has three subfolders (*RestartIn, RestartOut, UserOut*) and three files (*User_Output_Data_Saved.m, User_Output_Movies.m, User_Settings.m*). The *RestartIn* and *RestartOut* folders are used for setting up a restarted run (i.e. a run that follows from a previously completed run). The *UserOut* folder is where the model will write the output created during a model run. The *User_Output_xxxx.m* files are used to configure the data saved by the model. The *User_Settings.m* is used to configure the model run.

3 Customization of Input Files

3.1 Folders and files

Folders and files for a new simulation should follow the hierarchy described in § 2.2 above. Users with expertise in Matlab can customize the model path and folder hierarchy as desired, but careful attention must be paid to the path relationships between some of the input folders/files and common file locations.

3.2 Modifying File PC2_solver_main.m

The file *PC2_solver_main.m* is used to execute NDHM, and contains the information on where Matlab should look for other input and model files. The Matlab code in this file must be modified for a new simulation to provide the new directory names.

The directory wher the PC2 code is installed must be provided as *setting.Folder.Model* similar to:

```
setting.Folder.Model =...
     '/usr/NDHM/PC2 Matlab Code v06 2011 06 03'
```

The directory where the input and configuration data for the model can be found must provided as *setting.Folder.UserDirectory*, similar to:

```
setting.Folder.UserDirectory =
     '/usr/NDHM/Nueces2010_WR_14d_en_0p';
```

Failure to change these directories for a new simulation may cause the model to either re-run a prior simulation, or crash due to an inability to find the correct folder.

3.3 Modifying File User_Settings.m

The *User_Settings.m* file is the principal file for configuration of a model run. The model sets defaults for many standard configuration settings, so this User Manual will focus on settings that the user may need to alter.

3.3.1 Restart Write File Output Configuration:

A set of restart output files usually should be saved so that a simulation may be restarted at a later date. The *User_Settings.m* file should contain the following

```
setting.Restart.Write_Type  = 'runfile_keep';
setting.Restart.Write_Interval  = 100;
setting.Restart.Writefile_Folder  = 'User_Code/RestartOut';
setting.Restart.Writefile_Name  = 'restart';
```

The setting *runfile_keep* requires the model to keep a restart file every *Write_Interval* time steps in the folder *User_Code/RestartOut* with the output files named *restart###* where ### is the time step that the particular restart file was written.

3.3.2 Restart Read File Input Configuration

The restart read setting should generally be commented unless a restarted simulation is required (see § 4.3 below). These lines (commented out) should look like

```
% setting.Restart.Read_Type = 'run_file';
% setting.Restart.Readfile_Folder = 'User_Code/RestartIn';
% setting.Restart.Readfile_Name =
```

3.3.3 Time step size a number of steps

The model *dt* (number of seconds in a time step) for the simulation is defined by

```
setting.Time.dt = 300;
```

where 300 seconds has proven an acceptably robust value for the Nueces Delta. The number of time steps in a simulation is controlled by the setting *nstep*, defined by

```
setting.Time.nstep = 2016;
```

where 2016 time steps is a 7 day simulation for *dt* = 300 seconds. Although the model has the capability to provide for an adjustable time step (i.e. *dt* changing as a function of the flow conditions), this feature is not recommended for NDHM, so the following setting should be retained

```
setting.Time.adjustable_dt = ‘no’;
```

Future versions may include a more robust adjustable time step feature.

3.3.4 Input Subdirectory:

If all the boundary and initial condition input files are stored directly in the folder *Input,* then the *User_Settings.m* file should contain the line

```
setting.Folder.Input_Subdirectory = '';
```

However, there may be cases where the user would like to store input data in a subfolder of input, in which case that subfolder name should be provided in the single quotation marks. Note that the path name should *not* be provided, only the subfolder name.

3.3.5 Grid Geometry

The name of the input files for the bathymetry, bathymetry edges, initial conditions for the free surface, and the vertical layers are provided through

```
setting.Grid.File.Bathymetry          = 'Input_Bathymetry_Nueces1';
setting.Grid.File.BathymetryEdges     = 'Input_Bathymetry_Edges';
setting.Grid.File.Surface             = 'Input_Free_Surface_Nueces1';
setting.Grid.File.Layer               = 'Input_Layer_Nueces1';
```

The size (in meters) of each model grid cell in the *x* and *y* directions are defined by

```
setting.Grid.dx = 15;
setting.Grid.dy = 15;
```

where 15 m has proven practical for the NDHM. Note that if the grid *dx* or *dy* is changed, the entire bathymetry input file, bathymetry edge file and all the inflow and tidal boundary condition files must be modified.

3.3.6 Temporal Boundary Conditions:

File names for time-varying boundary conditions on inflows, tides, wind, rain and meteorology are provided using

```
setting.BC.File.Inflow = 'Input_Inflow_Boundary_Condition';
setting.BC.File.Tide = 'Input_Tidal_Boundary_Condition';
setting.BC.File.Wind = 'Input_Wind_BC';
setting.BC.File.Rain = 'Input_Rain_BC';
% setting.BC.File.Meteorology = 'Input_Meteorology_BC';
```

Note that meteorological modeling (temperature, evaporation) requires further development for NDHM and is commented out.

3.3.7 Solution Method

The PC2 model contains a variety of options controlling how it solves the Navier-Stokes and transport equations. Most of these options are associated with how the model handles extreme cases with thin layers or artificially high velocities. Many of these are selected by defaults with the code itself. A few of the settings are illustrated in the *User_Settings.m* file. It is suggested that the following settings be used for any NDHM model.

```
setting.NavierStokes.corrector_steps = 0;
setting.NavierStokes.DryCell.Model = 'Zlimit';
setting.NavierStokes.DryCell.Zlimit.Zinterval = 0.25;
setting.NavierStokes.MinimumDepth = 1e-6;
setting.NavierStokes.MomentumNonlinearLimiter.Model = 'CFL';

setting.Groundwater.Model = 'UplandConstantLossRate';
setting.Groundwater.UplandConstantLossRate.DepthLossRate = 1.0e-4;
setting.Groundwater.UplandConstantLossRate.UplandElevation = 1.5;

setting.Scalar.SubTimeStep.Method = 'FluxVolumeLocal';

setting.Wind.ThinLayer.Model = 'ExponentialDecay';
setting.Wind.ThinLayer.ExponentialDecay.DepthCutoff = 0.5;
setting.Wind.ThinLayer.ExponentialDecay.CwT = 5;

setting.Velocity.Limiter.Model = 'SourceCutoff';
setting.Velocity.Limiter.SourceCutoff.CFLx = 6;
setting.Velocity.Limiter.SourceCutoff.CFLy = 6;

setting.BottomDrag.ThinLayer.Model = 'LinearIncrease';
setting.BottomDrag.ThinLayer.LinearIncrease.ThinLayerDepth = 0.06;
```

3.3.8 Output formatting:

The default for NDHM is to provide several types of output: *Information, Status, Movie, Figure*, and *Dat*a. Each of these categories can be controlled from the *User_Settings.m* file.

The *Information* is output to the command line of Matlab that provides detail on model settings selected by the user (or through model defaults). Generally this *Information* is valuable to an expert in debugging a model that is working poorly. To provide *Information* output, use the setting

```
setting.Output.Information = 0;
```

To suppress information, use

```
setting.Output.Information = 1;
```

The *Status* output provides command line output that tells the user what the model is presently doing. Again, this is mostly useful to experts in model debugging. However, it also provides feedback that lets the user know the model is actually doing something. The general command to see *Status* output is

```
  setting.Output.Status= 1;
```
with "=0" used to suppress the status output.

Control for movies of model output are controlled by the *User_Output_Movies.m* file. To suppress movie output without changing the file, use

```
  setting.Output.SkipMovie = 1;
```
Skipping movies is generally a good idea for any simulation for a large number of time steps. The movies significantly slow the model and can cause Matlab to run out of memory.

Figures are output that the model provides that are helpful in debugging. These figures track data through the simulation and then graph it when the simulation is finished. For a large number of time steps, these figures can cause problems, so it is a good idea to skip them with

```
  setting.Output.SkipFigure = 1;
```
To see the figures at the end of a run use *=0.*

In unusual conditions a user might want to run a simulation without creating the data output defined in the *User_Output_Data_Saved.m* file. To suppress the data output, use

```
  setting.Output.SkipOutput = 1;
```
Note that the model run with *=1* will not produce any output, and won't be of much use. Generally use *=0*

There are several *Write_to_File* settings that provide control over the simulation output. At the present time, the only change that the user might want to make is to

```
  setting.Output.Write_To_File.FileName = 'Test';
```
which controls the first part of the file name on the output files. The user may provide any string in place of "Test," which will change the output file names.

3.3.9 Velocity Initial Conditions:

The initial conditions for velocity are defined in the *Input_Velocity_IC.m* file, using

```
  setting.Velocity.InitFunction = 'Input_Velocity_IC';
```
For NDHM, the velocity initial conditions are unknown, so the file *Input_Velocity_IC.m* sets the initial velocity to zero.

3.3.10 Bottom Drag Coefficient:

For NDHM, the method used to specify bottom drag is the traditionall to Manning's n value, which is specified by

```
  setting.BottomDrag.method = 'ManningsN';
```
A global default value for Manning's n is set with

```
  setting.BottomDrag.ManningsN = 0.025;
```
However this value is overwritten by the Manning's n values provided through the file specified by

```
  setting.BottomDrag.InitFunction = 'Input_BottomDrag_IC';
```
See § 3.6.8 for a information on the bottom drag input file

3.3.11 Cell Edges:

Because the Nueces Delta has narrow railroad dikes that cannot be easily represented by bathymetry of a 15m x 15 m grid, the PC2 cell edge feature is used to define the dikes as edges of model cells that are higher than the cell centers. There are several different ways of handling the mathematics for cell edges within the PC2 code. Form NDHM the recommended approach is to treat the edge value as a maximum that is independent of the cell center elevation. This feature is known as a *sill_max* approach, which is defined using

```
setting.CellEdge.type = 'sill_max';
```

3.3.12 Scalars:

Scalars for NDHM include salinity and any tracers that the user desires to create. In the existing data set a tracer called *Blue* is used to track the flow of water from the Calallen-Rincon pumping station. Each scalar must have a unique name that is provided in the setting.Scalar.fieldname list, similar to

```
setting.Scalar.fieldname = {'Salinity','Blue'};
```

New scalars can be added after *Blue* with each fieldname separated by a comma. The temporal boundary conditions for a scalar need to be provided in a file, with the file name specified by a command similar to

```
setting.Scalar.BCfunction.Blue   = {'Input_Blue_BC'};
```

Note that the field name of the scalar (in this case *Blue*) must be correctly represented on the left-hand-side of the command. Similarly, an initial condition function is provided by

```
setting.Scalar.InitFunction.Blue = {'Input_Blue_IC'};
setting.Scalar.InitArgument.Blue = {'array3D'};
```

where the *InitArgument* allows the user to design an initial spatial distribution (see file *Input_Velocity_IC.m* for an example). Maximum and minimums for scalars are recommended to allow the code to truncate spurious high values that sometimes occur in very thin layers. These are set with

```
setting.Scalar.Minimum.Blue = 0;
setting.Scalar.Maximum.Blue = 1;
```

In the present code, all scalars should be modeled using the scalar mass approach with the setting

```
setting.Scalar.UseConcentrationToZero.Blue = 'yes';
```

To prevent the code from spending time computing meaningless small concentrations, it is useful to set a lower bound on concentrations that the model will truncate to zero. This feature is invoked using

```
setting.Scalar.UseConcentrationToZero.Blue = 'yes';
setting.Scalar.ConcentrationToZero.Blue = 1e-6;
```

3.4 Modifying User_Output_Data_Saved.m

The *User_Output_Data_Saved.m* file is used to define the data that will be saved. This file is a Matlab function that provides the data name and the step interval for saving data, similar to

```
user.Output_Data_Name{ii} = 'Surface Elevation';
```

```
  user.Output_Data_Interval{ii} = 12;
  ii=ii+1;
```

Note that the *ii=ii+1* is a counter used to increment storage so that it is easy to comment out or add new data.

The *Data_Interval* setting is the number of time steps (not the number of seconds) between saving data sets. Thus, the time (in seconds) between data output is given by *dt x Data_Interval*.

A common error is including a *Data_Name* in this file that is not actually simulated. For example, if we us

```
  user.Output_Data_Name{ii} = 'Red';
  user.Output_Data_Interval{ii} = 1;
  ii=ii+1;
```

but do not include specifications for scalar *Red* in *User_Settings.m* (see § 3.3.12), then the model will crash when it tries to process the *User_Output_Data_Saved.m* file.

The depth, surface elevation, and three velocity (*U, V,* and *W*) variables can be saved for any simulation, as they are a part of every simulation even though they are not specificed in the *User_Settings.m* file. Note that velocities can either be saved on the cell faces (which is what the model uses) or interpolated to the cell centers (which provides *U V* and *W* velocities at the same location).

3.5 User_Output_Movies.m

This file defines what debugging movies will be created from the simulation. These movies are created directly after the model has finished running, and are mainly useful for rapid debugging by model developers. It is recommended that users develop their own scripts for displaying data and creating movies directly from the output data.

3.6 Input Files

3.6.1 Inflow Boundary Condition file

The *Input_Inflow_Boundary_Condition.m* file defines the locations of the inflows (other than tidal forcing) and the values of the inflows at different times. Each inflow is provided a unique identification (*ID*), a type (*zType*), an inflow side (*side*) and a set of ii, jj indexes (*ijPair*). The *dataName* for an inflow is always *Inflow*, to distinguish it from tidal or rainfall inflows. The *ijPair* identifies the indexes in the bathymetry for an inflow cell, and the *side* identifies the side of the cell the inflow comes through. In NDHM, the inflows (USGS gage and Calallen-Rincon pipeline) are treated as *zType=bottom* and *side=–z*, so the inflow is treated as rising from the bottom of the grid cell. This bottom inflow treatment is consistent with the model grid scale of 15x15 m grid, which cannot accurately capture the smaller scale features near the inflows that would be required for a treating the inflows as side boundaries.

The temporal data in the inflow file is similar to other boundary condition files, and is discussed in § 3.6.3 below.

3.6.2 Tidal Boundary Condition file

The *Input_Tidal_Boundary_Condition.m* file provides the *ID*, *dataName, side*, *zType* and *ijPair* similar to as discussed for inflows in § 3.6.1 above. The dataName for a tidal boundary is always *Tide*. For the NDHM, the tidal boundary is enforced along the *+y side* of the *ijPair* cells. The tidal boundary is enforced over the entire water column, so the *zType* is *column*. The *ID* for the tide in the current NDHM is *WhitePointTide*, which is the location of the tide gauge used for the boundary condition data. The *ijPair* for the tidal input are *ijPair(:,1) = 353:609* (rows 353 - 609) and *ijPair(:,2) = 970* (column 970), as shown in Figure 1.

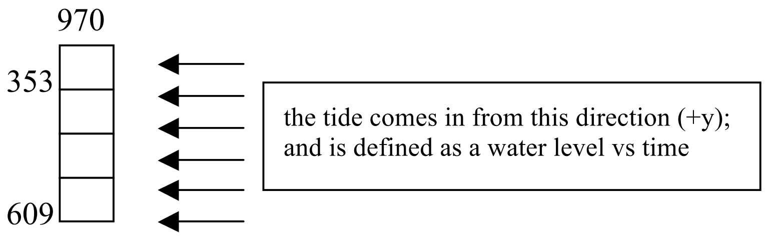
970
353
609
the tide comes in from this direction (+y); and is defined as a water level vs time

Figure 1. Tidal boundary condition enforced in column 970 along y+ boundary.

The temporal data in the tidal BC file is similar to other boundary condition files, and is discussed in § 3.6.3 below.

3.6.3 Temporal Boundary Conditions

Temporal boundary condition files (file names using the form *Input_xxxx_BC.m* or *Input_xxxx_Boundary_Condition.m*) require input for a value versus time. Values depend on the simulation models applied, but typically include rainfall rate, tidal elevation, inflow rate, salinity, scalar concentration, wind speed, and wind direction. The time must be provided as time in seconds (considering the starting point to be zero seconds). Future versions of the model will include year, month, day, clock formatting for time.

As a typical example, a tidal boundary condition with data provided every 30 minutes (1800 seconds) for 7 days might look like

```
tidal(1).data2D = …
[0    ,      0.069  ;…
1800 ,      0.061  ;…
3600 ,      0.056  ;…
.
.
.
604,800 ,   0.533];
```

where 332 lines of data have been truncated. The first column is the time (in seconds), the second column is the tidal elevation (in meters).

Note that thhe model will crash if insufficient BC data is provided. Each BC file should have time that covers *dt* x *nstep* (see § 3.3.3).

3.6.4 Tracer BC and IC files

The model has been tested with different "colors" of tracers associated with different inputs. These are

```
Input_Blue_BC.m
Input_Blue_IC.m
Input_Green_BC.m
Input_Green_IC.m
Input_Red_BC.m
Input_Red_IC.m
Input_Yellow_BC.m
Input_Yellow_BC.m
```

The BC files provide the concentration of the tracer for each of the inflow, tide, or rain inflow cell sets. The IC files provide the initial distribution of the tracer. Because the tracers can significantly increase the computational time, it is recommended that the user only set up the tracers of interest. For the runs conducted in the Nueces Delta Hydrodynamic Modeling project, only the Blue (pumped water) tracer was used.

The BC files must specify tracer concentrations (even if zero) at every tidal inflow, river, or rainfall boundary. The connection between the tracer BC file and the tide/flow/rain BC file is through the scalar.ID setting, typically appearing as

```
scalar(ii).ID = {'WhitePointTide'};
```

The above ID must match one of the ID's given in a tide/flow/rain BC file, in this case we find in *Input_Tidal_Boundary_Condition.m* an ID of

```
tidal(1).ID = 'WhitePointTide';
```

Mismatches in the ID between the scalar value at an inflow location and the name of the inflow location will cause the code to stop with an error flag. n

The temporal data in the tracer BC files is similar to other boundary condition files, and is discussed in § 3.6.3 above

3.6.5 Salinity BC and IC files

The salinity BC and IC files *Input_Salinity_BC.m* and *Input_Salinity_IC.m* are similar to the tracer files discussed in § 3.6.4. For the NDHM, zero salinity is used for all pumped water, river inflow, and rainfall. The salinity for the tide is set equal to the observed salinity measured at SALT03 in Nueces Bay from TCOON.

3.6.6 Bathymetry:

The file *Input_Bathymetry_Nueces1.m* is called to initialize the model bathymetry. Although the PC2 code is designed to read a Matlab function that describes the bathymetry, for NDHM it is convenient to pre-process the bathymetry into a Matlab file (**.mat*) that is faster to read and process. The bathymetry file used for NDHMv1.0 *bathymetry_baybathy15_channels.mat*, which is a common file stored at the top of the user installation directory. When the model calls *Input_Bathymetry_Nueces1.m* it is directed to read the file *bathymetry_baybathy15_channels.mat* that is located two folders above *Input_Bathymetry_Nueces1.m.* Note that if either file is moved or if the name is changed, the call in *Input_Bathymetry_Nueces1.m* will need to be re-written.

3.6.7 Bathymetry Edges file

The file *Input_Bathymetry_Edges.m* defines the locations and heights of the sills used to block flow across the railroad dikes in the Nueces Delta. All of the *ii* and *jj* index locations of these sills (as well as their heights) are identified in the file. If you choose to manipulate this file, be cautious of the definition of *ii* and *jj* indexes, as improper identification can lead to discontinuities in the railroad dike. For the present simulations and bathymetry, this file should not need alteration.

The sill locations are defined using *iiP* (the positive *ii* side of a cell) and *jjP* (the positive *jj* side of a cell) and are associated with the height of the sill at this location. The cell faces are oriented as depicted in Figure 2.

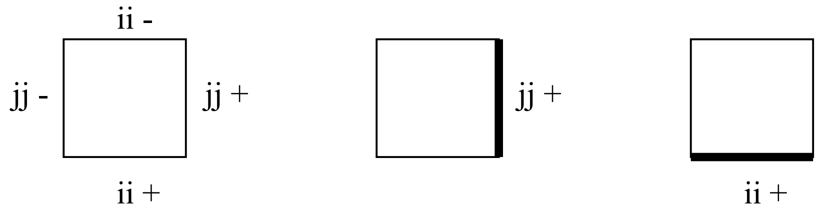

Figure 2. Definitions of the plus (+) and minus (-) faces of cells using the row (ii) and column (jj) index convention for Matlab

3.6.8 BottomDrag Initial Condition file

Input_BottomDrag_IC.m provides the settings for Manning's n used in the NDHM. This file calls the Manning's n Matlab binary file *manningsn_landcover_culvert.mat* that is common to all NDHM simulations. If the path relationship between the Manning's n file and the *Input_BottomDrag_IC.m* file is changed, then the latter file will need to be customized.

3.6.9 Free_Surface Initial Condition file

Input_Free_Surface_Nueces1.m sets the initial water level in the delta. This value should be equal to the initial tide level in the file *Input_Tidal_Boundary_Condition.m*. If there is a mismatch between the initial water level and the initial tide, the model may see a sharp shock at the first time step and crash.

3.6.10 Layer file

The PC2 code is designed for both 2D and 3D operation. In 3D operation, the vertical dimension is discretized into thin layers. For a 2D model, such as NDHM, there are always three layers: the operational layer (in the middel) and ghost layers above and below for enforcing boundary conditions. *Input_Layer_Nueces1.m* sets up the number of layers used in the model. The layer thickness in NDHM is required to cover all the terrain represented in the bathymetry, which ranges from approximately -4 m to 25 m in elevation. The model presently uses a thickness of 100m, although this could readily b reduced to 30 m without affecting model operation.

3.6.11 Velocity Initial Condition file

This file defines the initial condition for the water velocity in the system. In the NDHM, we set the velocity in all directions equal to zero.

3.6.12 Wind Boundary Condition File

Input_Wind_BC.m provides woth wind speed and direction using

```
  dataName = {'WindSpeed','WindDeirection'};
```

The *zType* is *surface*, *side* is *z+* and *ijPair* is the empty set as the wind is enforced over the entire free surface. Unlike the inflow and tidal boundary conditions, the wind file uses three columns

```
  [ time (in seconds) , wind speed (in m/s) , wind direction (in degrees) ; …
```

Note that the wind direction uses the conventional meteorological definition as the direction the wind is coming from.

3.6.13 Unused input files

The *Input_DragOutflow_BC.m* and *Input_Meteorology_BC.m* files are not used in NDHMv1.

*3.7 Common Initial Condition *.mat Files*

Because the 15 x 15 m grid applied to the Nueces Delta requires on the order of 500,000 grid cells, specifying the bathymetry and initial conditions in an ASCII text file is not recommended. Each (i,j) grid cell location requires a bathymetry elevation value, a Manning's n value, and an initial salinity. Data processing for the 15 x 15 m bathymetry, salinity and land cover is described by Ryan (2011). The data processing created salinity files for different dates (YYYYMMDD). The common files are

```
        20080414_salinity.mat
        20090410_salinity.mat
        20100403_salinity.mat
        20100409_salinity.mat
        bathymetry_baybathy15_channels.mat
        manningsn_landcover_culvert.mat
```

The initial conditions for salinity were created as a 3D matrix; the z-direction has identical values (since 2 of these layers are ghost layers). The x & y directions match up with the salinity in the delta, defined as being equal from north – south and varying with the salinities measured at the TCOON stations from east – west. The values at the TCOON stations were averaged through time for the day previous to the start of simulation, and then linearly interpolated between stations to create the initial condition matrix.

4 Model Execution

4.1 *Running a single simulation*

NDHM is run from the command line of Matlab. The user must be in the folder where the *PC2_solver_main.m* file is located for the particular model case. Because PC2_solver_main.m contains a customized path for a particular model case, the user must be careful to ensure that their Matlab command line is at the correct location before trying to run the model.

The model is executed by entering

```
PC2_solver_main
```

at the Matlab command line. The NDHM models run on a single processor of a 2.66GHz Intel Xeon multi-core chip have been running at 7 times faster than real time using a 300 second time step. It is recommended that model runs that are expected to last more than a few days should be broken up into pieces and run using the restart capability, see § 4.3 below.

4.2 *Running batch simulations*

With the Matlab Parallel Computing Toolbox, it is possible to run multiple NDHM models at the same time. Each model case uses a separate processor. The Matlab Parallel Toolbox presently supports up to 8 simultaneous jobs on a single workstation. However, the NDHM uses requires 4 to 5 GB of memory for each model, which may be more limiting. Matlab will allow 8 jobs to be started, but if there is not enough memory available, either Matlab may crash or the models be forced to use hard disk space as memory, which will significantly slow the operational time.

Parallel jobs are run using a Matlab *batch()* command. Each NDHM model requires its own folder hierarchy (see § 2.2) and its own *PC2_solver_main.m* file. To run the first model in parallel batch mode, change the Matlab command line to the appropriate model folder and type the command

```
job1 = batch('PC2_solver_main')
```

After changing to the folder to the location of the second model, at the command line type

```
job2 = batch('PC2_solver_main')
```

Similar commands are issued at the command line within the folders for other model runs.

Progress on the simulations can be monitored using the following commands within any folder

```
sched = findResource()
[JobPending JobQueued JobRunning JobCompleted = findJob(sched)
```

To cancel a simulation use:

```
cancel(JobRunning(##))
```

where ## is from the *findJob(sched)* output shown above. Here is a summary of the commands used in running batch simulations:

4.3 Restarting a Simulation

A simulation can either be restarted within its existing folder hierarchy, or a new hierarchy can be created. If an existing folder is used, existing output files may be overwritten.

A restart can only be done when a model has been run with the *Restart Write* option (see § 3.3.1). The restart files from the prior run were written into the *RestartOut* folder. Each file should have a number indicating the time step when the file was written.

Move or copy the desired restart file from the *RestartOut* folder into the *RestartIn* folder.

Open the *User_Setting.m* file, and uncomment or add the following lines of code

```
setting.Restart.Read_Type = 'run_file';
setting.Restart.Readfile_Folder = 'User_Code/RestartIn';
setting.Restart.Readfile_Name = 'restart_2500.mat';
```

where *restart_2500.mat* should be replaced with the name of the restart file you want to use.

Modify the *setting.Time.nstep* in *User_Setting.m* file for the number of time steps that you want to add to the restarted simulation.

5 Output Handling

The output of a simulation is located in the *User_Code/UserOut* folder.

If the *User_Settings.m* file includes the line

```
  setting.Output.Write_To_File.FileName = 'Test';
```

then the raw model output saved is located in *Test_##.mat* files, where ## is a number indicating to the time step. If the *Write_To_File.FileName* is given a different character string, then that string will be the leading string for the raw output files. For model efficiency, these raw output files store data in one-dimensional arrays, which are converted to 2D or 3D space at the end of the simulation

If the simulation was run was successfully without stopping or restarting through the specified *setting.Time.nstep,* then a file named *Test_FINAL_###.mat* will automatically be created, where ### is a date and time stamp when the file was created. This file contains the output data formulated into arrays that directly map to the bathymetry indexes. The string *Test* at the start of this filename will be replaced by whatever string is specified in the *User_Settings.m* command

```
  setting.Output.Write_To_File.FileName = 'Test';
```

If a run is restarted, the output *Test_FINAL _###.mat* file will contain only the data for the restarted run. If a run crashes, then the *Test_FINAL_###.mat* file will not be created. Additional Matlab scripts to recover data from a crashed run are under development.

The *Test_FINAL_###.mat* file can be loaded into Matlab and contains all the data output by the model in OutData2D and OutData3D arrays.

6 References

Hodges, B.R., and F.J. Rueda (2008). "Semi-implicit two-level predictor-corrector methods for non-linearly coupled, hydrostatic, barotropic/baroclinic flows." *International Journal of Computational Fluid Dynamics*, 22:9:593-607.

Ryan, A.J. and B.R. Hodges (2011), *Modeling Hydrodynamic Fluxes in the Nueces River Delta,* Technical Report, CRWR Online Report 11-7, Oct. 10, 2011. 86 pgs.