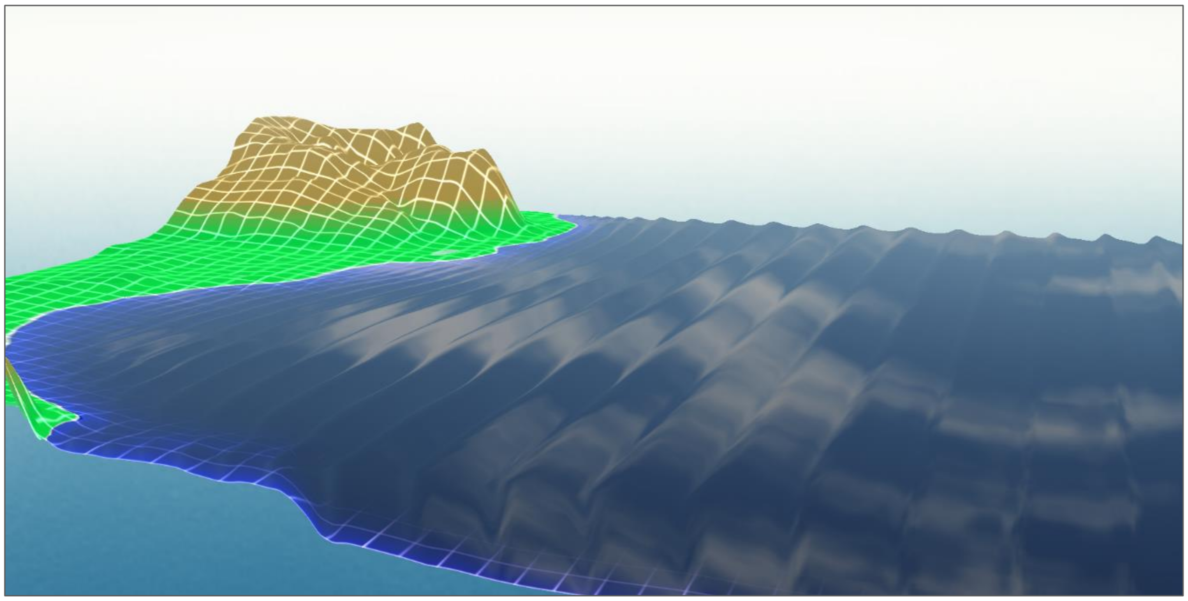
CELERIS ADVENT (v1.0)

USER’S MANUAL
Version 1.0



Prepared by
Sasan Tavakkol
Patrick Lynett, PhD

August 19, 2016

Introduction

This guide documents the use of wave simulation and visualization software, Celeris. This software solves the extended Boussinesq equations and is accelerated by the GPU. Celeris uses a hybrid finite volume – finite difference scheme to discretize the governing equations which are used in their conservative form. The tuple (w, P, Q) describes the flow parameters in a cell, where *w* is the water surface elevation from a fixed datum. *P* and *Q* are the integrated mass fluxes in *x* and *y* directions, respectively. Numerical and implementation details of the software can be found elsewhere and are out of scope of this writing. Celeris takes an XML file as the input setup for a specific experiment, but it also has an interactive GUI. It can also write data to ASCII formatted files. This document elaborates on the file formats and the GUI. Note that Celeris uses metric system.

Input files

As mentioned earlier we chose XML format as the input setup file for experiments, mostly because it is both human and machine readable. Any standard text editors, such as notepad++, can be used to edit these files. XML files consist of elements. An element generally begins with a start-tag and ends with the matching end-tag; any characters in-between are the contents of the element. For instance Fig. 1 shows an element which describes the name of an experiment.

```
<name>Conical Island</name>
```

Fig. 1- A sample XML element.

An element, can contain other elements, and in fact all the elements in an XML file must be located inside the "root" element. In addition to content, elements can also have attributes. For example Fig. 2 shows the element `<westBoundary>`. This element have three attributes, namely, *type*, *seaLevel*, and *widthNum*. It also contains another element, `<sineWave>`. This figure also shows a sample comment. Comments in XML begin with `<!--` and end with `-->`.

```
<!-- Settings for Boundaries-->
<westBoundary type = "SineWave" seaLevel = 0 widthNum = 2>
    <sineWave amplitude = .01 period = 2 theta = 0></sineWave>
</westBoundary>
```

Fig. 2- An XML element with attributes containing another element.

In order to distinguish Celeris input files from generic XML files, we set the extension of our input files to CML. These files, as any XML file, can be also easily edited with a standard text editor. Fig. 3 shows a complete CML for a sample experiment. We will walk through this file and explain its elements. The root element is `<Experiment>` and everything must be places inside this element. The name of the project is placed in the `<name>` element. This string value is shown on the title bar of Celeris.

Model Settings

The model settings are incorporated in the element `<model>` which have an attribute *type* and two elements `<parameters>` and `<friction>`, each with several attributes. The *type* attribute can be set to "BSNQ" or "NLSW" to switch between Boussinesq model and shallow water equations. The attribute *epsilon* is the value of ϵ in the governing equations which is used to avoid divisions by zero. The value of this parameter can have effect on the runup measurements and stability of the model. Larger values makes the code more stable, but increases the artifacts in simulating runups. Generally speaking, the simulation results in areas with water depth less than $\epsilon^{1/4}$ may have anomalies. The attribute *correctionStepsNum* sets the number of corrections steps and the attribute *timestep* is the size of timestep (d*t*) is seconds. The attribute *type* of the element friction can be switched between "Manning" and "Quadratic". Depending on the value of *type*, the attribute *coef* is either the Manning's roughness coefficient (*n*) in SI units, or quadratic friction coefficient (*f*).

```
<?xml version="1.0"?>
<Experiment>
    <name>Sample Experiment</name>
    <!-- Settings for Model -->
    <model type = "BSNQ">
        <parameters epsilon = 5e-12 correctionStepsNum = 2 timestep = 0.005></parameters>
        <friction type = "Manning" coef = 0.0> </friction>
    </model>
    <!-- Settings for Solution field -->
    <fieldDimensions width = 30 length = 30 stillWaterElevation = 0></fieldDimensions>
    <gridSize nx = 601 ny = 601></gridSize>
    <bathymetryFilePath> \resources\bathy.cbf </bathymetryFilePath>
    <!-- Settings for Initial Condition -->
    <hotStartFilePath> N/A </hotStartFilePath>
    <solitaryWave H = 0.05 theta = 0   xc = 5 yc = 15></solitaryWave>
    <solitaryWave H = 0.05 theta = -45 xc = 5 yc = 25></solitaryWave>
    <!-- Settings for Boundaries-->
    <westBoundary type = "SineWave" seaLevel = 0 widthNum = 2>
        <sineWave amplitude = .01 period = 2 theta = 0></sineWave>
    </westBoundary>
    <eastBoundary  type = "Sponge" seaLevel = 0 widthNum = 20></eastBoundary>
    <southBoundary type = "Solid"  seaLevel = 0 widthNum = 2></southBoundary>
    <northBoundary type = "Solid"  seaLevel = 0 widthNum = 2></northBoundary>
    <!-- Settings for Logging Data-->
    <logData doLog = true logStep = 20>
        <logPath>C:\conical_island\</logPath>
        <range filename = "island">
            <bottomLeft x = 228 y = 228></bottomLeft>
            <topRight x = 374 y = 374></topRight>
        </range>
        <gauges filename = "gauges">229,302,249,302,353,302,354,302</gauges>
    </logData>
</Experiment>
```

Fig. 3- A sample CML input file.

Field Parameters

Elements `<fieldDimensions>`, `<gridSize>`, and `<bathymetryFilePath>` set the properties of the solution field. The first element in this list has three attributes, *width*, *length*, and *stillWaterElevation* which all must be given in meters. Note that in Celeris width of the field is defined in *x* direction, and length of it is defined in *y* direction. Still water elevation is defined according to a fix datum, and most of the time is set to zero. The element `<gridSize>` has attributes *nx* and *ny* which define the number of cells is x and y direction, respectively. `<bathymetryFilePath>` is the relative or absolute path to a Celeris bathymetry file (*.cbf). If a relative path is used, Celeris will attempt to find the file relative to the location of the CML file itself.

Celeris bathymetry files are ASCII formatted text files, with the extension "cbf". Fig. 4 shows a sample cbf file. As can be seen the file starts with two tags `[nx]` and `[ny]` which determine the cell numbers in *x* and *y* directions, respectively. More descriptive tags will be added to this section in the next versions of Celeris. The properties section ends with a series of "=" signs. Afterwards the bathymetry matrix is given. A short MATLAP script called "write_bathy.m" is provided with Celeris to help creating bathymetry files. If the cell numbers in the bathymetry file (`[nx]` and `[ny]`) are different from the cell numbers (*nx* and *ny*) in the experiment, Celeris will use linear interpolation to generate the bathymetry.

```
[nx] 100
[ny] 50

====================================
-0.45000000 -0.40000000 -0.35000000 ...
-0.45000000 -0.40000000 ...
-0.45000000 ...
     ⋮
```

Fig. 4- A truncated sample Celeris bathymetry file.

Initial Conditions

Two type of initial conditions can be set in Celeris. Firstly, a Celeris hot-start file (*.chf) can be given to the software as the initial condition by setting the content of the element `<hotStartFilePath>` to the relative or absolute path of the file. This file must define the values of w, P, and Q on the whole domain and is formatted similar to Celeris output files. Celeris does not use interpolation for hot-start files and therefore, the given file must have the exact same cell numbers as the experiment in the both directions. Secondly, several solitary waves can be added to the field as the initial conditions. The properties of a solitary wave must be set as the attributes of the element `<solitaryWave>`. These attributes are the waveheight in meters (*H*), wave direction with respect to *x* axis in degrees (*theta*), and coordinates of the center of the wave (*xc* and *yc*). A CML file can have several solitary wave elements.

Boundary Conditions

Celeris uses two layers of ghost cells on each side of the field to implement the boundary conditions. This means that for an experiment with mesh size of *nx* X *ny*, Celeris defines a matrix of size (*nx*+4) X (*ny*+4) to store the flow parameters. Cell indices start from 0 in both direction and ends with *nx*+3 or *ny*+3 on each direction. The west and east boundaries are defined parallel to the *y* axis and are located on x_j = 2 and x_j = *nx*+1, respectively. Similarly, the south and north boundaries are defined parallel to the *x* axis and are located on y_i = 2 and y_i = ny+1, respectively. The boundary conditions for the four sides of the field must be all set in the CML file, as the attributes of the elements `<westBoundary>`, `<eastBoundary>`, `<southBoundary>`, and `<northBoundary>`. The *type* attribute of the boundary in Celeris Advent can be chosen among "Solid" for reflective solid walls, "Sponge" for sponge layers, and "SineWave" for a sinewave maker. The *seaLevel* attribute is the elevation of the still water on the boundary, which for most of coastal engineering experiments must have the same value as stillWaterElevation. The *widthNum* attribute defines the number of boundary cells. This value must be set to 2 for "Solid" and "SineWave" boundaries and to the appropriate number, corresponding the sponge layer length, for "Sponge" boundary.

If the boundary *type* is set to "SineWave", the boundary element must contain a `<sineWave>` element which describes the properties of the target sinewave. The attributes of this element are *amplitude* in meters, *period* in seconds, and *direction* with respect to *x* axis in degrees. If the type of the boundary is not "SineWave" this element will be ignored.

Output files

Celeris can save the flow parameters w, P, and Q to ASCII formatted files. Since writing is generally expensive in time, Celeris has the option to save data only in a given interval and given locations in the flied. The element `<logData>` is the right place to set output or as it is called in Celeris, logging properties and it contains several other elements. The attribute *doLog* must be set to *true* to enable, and to *false* to disable logging. *logStep* defines the logging interval as the number of timesteps. The content of element

`<logPath>` must set to the relative or absolute path of the folder which is intended to contain the log files. This folder must already exist, and Celeris will not create it.

Locations for logging data are defined by cell indices in the field. Two types of locations can be defined. The first kind is a range which is defined by two opposite points of a rectangle in the field. Ranges are defined in the element `<range>` where the attribute *filename* is the name of the file for the defined range. The range element must contain two other elements `<bottomLeft>` and `<topRight>` each with two attributes *x* and *y*. These elements define the corner of a rectangle with the minimum and maximum x and y indices, respectively. It is worth to emphasize that the *x* and *y* attributes are cell indices, and not distances in meter. Also the user must be aware to always consider the correct numbering of cells as described earlier in the boundary conditions section. A CML file can contains several ranges for logging data. For example if in an experiment the run-up on two islands are subjects of interest, it is recommended to define two ranges, each one covering one of the islands, rather than a large range covering both. As mentioned before, writing data to a file adversely affects the speed of simulation.

The data can be also logged on given gauges and saved to a file with the name *filename*. The gauge coordinates are given as (x,y) tuples inside the element `<gauges>` as shown in Fig. 3. The tuples are comma separated and similar to coordinates for ranges, their values are cell indices. For example in the experiment shown in Fig. 3, the flow parameters will be saved to the file "gauges.txt" located in the folder "C:\conical_island", on gauges (229, 302), (249, 302), (353, 302), and (354,302), every 20 timesteps. A CML file can only have one element of `<gauges>`.

Celeris output files are ASCII formatted text files with the common extension "txt". The flow parameters on computational cells are logged in this file as one tuple of (*j*, *i*, *w, P, Q*, *alpha*) on each line. *j* and *i* are the cell indices in *x* and *y* directions respectively. *w*, *P*, and *Q* are the flow properties. *alpha* is written to file for debugging purposes, is generally equal to zero, and must be ignored by users.

Graphical User Interface (GUI)

Celeris as the first interactive software for simulation of coastal processes, has an embedded GUI which can be used to change simulation and visualization parameters while the model is running. For example the user can change mesh size, friction coefficient, or the CFL number using the GUI. Boundary conditions can be changed and solitary wave can be superposed on the field as the model is running. Celeris has also several visualization options such as photorealistic and colormapped rendering. The GUI is better understood by running the software and experiencing it. It is also introduced in a short video available at https://youtu.be/pCcnPU7PCrg, and therefore we do not explain the details of GUI in this guide.

Running the Software

Celeris needs the recent version of DirectX to be properly installed on a Windows machine. This library is now integrated as part of the OS and therefore Celeris will run on a recent Windows machine with no preparation. However, depending on the last update of the OS, an update might be needed. The software is run by launching the file "Celeris.exe". If an error regarding a DirectX file is shown, please update your DirectX from here. The absolute path to the CML input file of an experiment can be given in a file called "setting.init", which is located in the same folder as "Celeris.exe". If a path is not given or if it is invalid, Celeris will automatically ask user to locate the CML file in file browser window. Afterwards the software will immediately start the experiment and will visualize the results as the simulation progresses. More than one instance of Celeris can be run simultaneously, however it should be noted that all the instances may share the computing power of GPU.

Tutorial Case

In this section we show, step by step, how to simulate the conical island benchmark in Celeris. This benchmark is based on experiments of Briggs et al. [19] for solitary wave interaction around a conical island and is frequently used to validate numerical models. To assist with writing bathymetry files and post-processing logged data, a few MATLAB scripts are distributed with Celeris as well. These files are not part of the software and are not required to use Celeris.

The first step in a simulation is to generate the bathymetry of the experiment. Fig. 5 shows the experimental setup of conical island benchmark. We simulate this benchmark in a 30 m x 30 m numerical domain. The circular conical island with a base diameter of 7.2m and side slope of ¼ is centered in the solution field. The process is done in the following steps:

Open the “Examples” folder and copy “template.cml” in to the “Tutorial Case” folder. This folder also contains a file named “conical_island_ready.cml” which is the ready-to-use input file for this tutorial case. We want to edit “template.cml” and make it look like “conical_island_raedy.cml”. This folder also has a CBF file which contains the bathymetry. A log folder with a few MATLAB files for post-processing must be also there.

Rename the “tutorial_case.cml” file to “conical_island.cml” in a standard text editor software such as Notepad++ (download). Follow the next steps as shown in the video file “Celeris Tutorial Case.mp4”. In the tutorial case we simulate the “Case C” with *H*/*d* = 0.18, where *H* is the solitary wave height and *d* is the depth. Sponge layers are imposed on the boundaries parallel to the soliton, and solid walls on the two other boundaries. The domain is discretized by 301x301 cells with a constant time step of 0.005 s. Bottom friction is neglected.

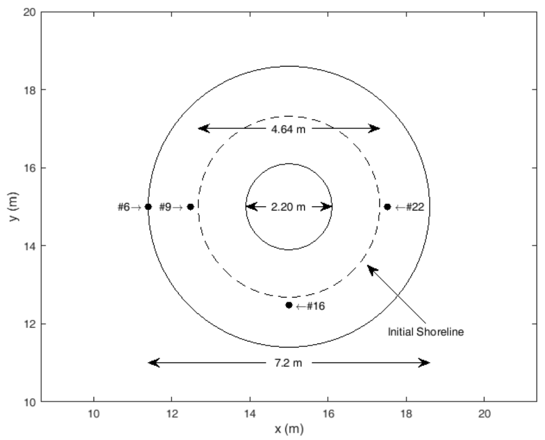


Fig. 5. Experimental setup of the conical island. The gauge locations are shown by dots and the wave approaches the island from the left.

The simulation results in an area surrounding the island and locations of gauges shown in Fig. 5 are selected to be saved on the disk on each 20 timesteps. After running the simulation, the logged results can be visualized in MATLAB using the provided scripts. The “gauges_read.m” script plots the simulation results and experimental ones on the 4 gauges shown in the figure. The “runup_island.m” script plots the maximum horizontal runup around the island for both experimental and numerical results