

Two-dimensional Runoff Inundation Toolkit for Operational Needs (TRITON)

Sudershan Gangrade ^{1,3}, Mario Morales-Hernández ^{2,3}, Md Bulbul Sharif ⁴, Tigstu T. Dullo ⁵,
Alfred Kalyanapu ⁵, Sheikh Ghafoor ⁴, Shih-Chieh Kao ^{1,3} and Katherine J. Evans ^{2,3}

¹ Climate Change Science Institute, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

² Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

³ Environmental Sciences Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

⁴ Department of Computer Science, Tennessee Technological University, Cookeville, TN 38505, USA

⁵ Department of Civil and Environmental Engineering, Tennessee Technological University, Cookeville, TN 38505, USA

User's Guide

TRITON

Current Version: April 03, 2020

Scheduled Release: May 2020

Table of Contents

1. Introduction.....	1
1.1 Overview.....	1
1.2 License and Availability	2
2. Obtaining the Source Code	3
3. Building the Executable.....	4
3.1 OLCF supercomputer – Summit.....	4
3.2 Non-OLCF machine.....	4
4. Running the Executable	6
4.1 OLCF supercomputer – Summit.....	6
4.2 Non-OLCF machine.....	6
5. Directory Structure and Features	7
5.1 Directory Structure.....	7
6. TRITON Configuration.....	8
6.1 Topography.....	8
6.2 Surface Roughness (Manning’s n Coefficient).....	8
6.3 Hydrologic Forcing.....	9
6.4 External Boundary	11
6.5 Simulation Control.....	12
6.6 Output Control	13
6.7 Input Output File Settings.....	14
6.8 Other Miscellaneous parameters.....	14
7. Test Cases Overview.....	16
8. Other Tools	18
8.1 Configuration File Generator	18
8.2 Ascii to Binary Conversion.....	18
8.3 Binary to Ascii Conversion.....	19
8.4 Discharge to Velocity Conversion	19
8.5 Join Utility	19
8.6 ASCII/Binary to NetCDF Conversion	20
9. References.....	21

1. Introduction

This document serves as a User's Guide for Two-dimensional Runoff Inundation Toolkit for Operational Needs (TRITON), which is a two-dimensional (2D) hydrodynamic model that simulates flood wave propagation and surface inundation based on the full shallow water equations. The purpose of this document is to provide new users with key information about the model and help them navigate through various topics including how to download, install and run the code. This guide also provides information about input and output files, model features and provides step by step instruction on simulating the pre-designed test cases or configure users' own simulation. Users are recommended to refer to technical reference manual for more advanced and detailed information.

1.1 Overview

The Two-dimensional Runoff Inundation Toolkit for Operational Needs (TRITON; Morales-Hernández et al., in preparation) is a 2D open source flood simulation tool designed for modern high performance computing (HPC). The core of the tool is a computationally efficient, physics-based hydraulic model that operates on a regular/structured grid and solves the full 2D shallow water equations. The key features of TRITON are:

- It can operate on multiple computer platforms and utilize modern HPC environments. The users can take advantage of:
 - Implementation with a single central processing unit (CPU) or multiple CPUs (using OpenMP+MPI)
 - Implementation with a single graphics processing unit (GPU) or multiple GPUs (using CUDA+MPI)

Highest TRITON computational efficiency can be achieved by using GPU implementation.

- TRITON utilizes topographical data (e.g., digital elevation model [DEM], light detection and ranging [LIDAR]), as its base input, in a uniform (Cartesian) grid structure. The model can be driven by streamflow hydrographs at specified locations or gridded runoff hydrographs, or both which serves as the model's hydrological forcing. The primary TRITON output includes water depth and 2D unit discharge maps at user defined time intervals. Other variables such as unit discharge values can be outputted. TRITON can also output timeseries of simulated results as user-defined point locations.
- TRITON is developed on Linux/Unix platform. The input/output files can be either in ascii or binary formats. A set of tools and instruction are provided for format conversion.
- The model utilizes International System of Units (SI). Users who are more familiar with United States (US) customary units need to perform proper unit conversion themselves.

1.2 License and Availability

- Name of the software: TRITON (Two-dimensional Runoff Inundation Toolkit for Operational Needs)
- Contact address: Oak Ridge National Laboratory, 1 Bethel Valley Road, TN 37830, USA / Tennessee Technological University, 1 William L Jones Dr, Cookeville, TN 38505, USA
- Email: Mario Morales-Hernández (moraleshernm@ornl.gov), Alfred Kalyanapu (akalyanapu@tntech.edu)
- Language: CUDA, C++
- Hardware: Desktop/Laptop or clusters of CPUs/GPUs
- Software: NVIDIA CUDA Toolkit, NETCDF (optional)
- Availability: <https://code.ornl.gov/hydro/triton>
- Year first available: 2020
- Copyright (c) 2019, UT-Battelle, LLC and Tennessee Technological University. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

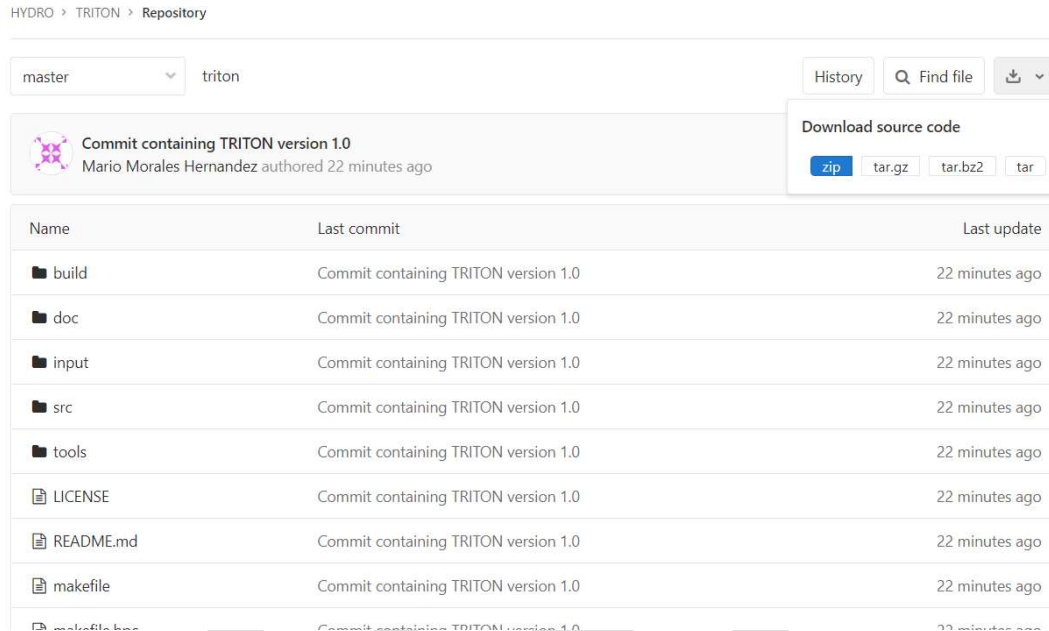
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

2. Obtaining the Source Code

The source code is available via several mechanisms. Users can obtain the code and pre-designed test cases using one of the following approaches.

- a. The latest version of code is maintained on the git repository. The packed archive file of the code and test cases is available to download from <https://code.ornl.gov/hydro/triton>. A user can download a tar.gz file from the website (as shown in the figure below) or in any other format as appropriate.



Assuming the downloaded file is named `triton-master.tar.gz`, use the following commands in terminal.

Use the following commands to unpack the file

```
$> tar -xzf triton-master.tar.gz
$> cd triton-master
```

Hereafter, within this document, this directory will be referred as `$TRITON`.

- b. Alternatively, users can directly clone the repository using the following commands

```
$> mkdir $TRITON
$> cd $TRITON
$> git clone https://code.ornl.gov/hydro/triton.git
$> cd triton
```

Hereafter, within this document, this directory will be referred as `$TRITON`.

To explore the directory structure, refer to Section 5.

3. Building the Executable

3.1 OLCF supercomputer – Summit

For GPU implementation

To create required environment, use the following commands

```
$> cd $TRITON
$> module load cuda
$> make clean && make summit_gpu
```

If successful, the executable `triton` will be located in the `$TRITON/build` folder.

For CPU (OpenMP) implementation

To create required environment, use the following commands

```
$> cd $TRITON
$> module load gcc
$> make clean && make summit_omp
```

If successful, the executable `triton` will be located in the `$TRITON/build` folder.

3.2 Non-OLCF machine

If you are using a Linux based machine, changes are required to the `makefile.hpc` to compiler flags in the file. The GPU based implementation requires `nvcc` compiler, while openMP requires `gcc` compiler.

The `makefile.hpc` contains following information:

```
ifdef ACTIVE_GPU
    CC := nvcc
    INC_DIRS :=
    FLAGS := --compiler-bindir `which mpic++` -arch=sm_35 -x cu
    LIBRARIES :=
    DFLAGS := -DACTIVE_GPU=1
else
    CC := mpic++
    INC_DIRS :=
    FLAGS := -Wall -fopenmp
    LIBRARIES :=
    DFLAGS := -DACTIVE_OMP=1
endif

triton: src/main.cpp
    if [ ! -d "build" ]; then mkdir build; fi
    @echo 'Compiling file: $<'
```

```
$(CC) $(INC_DIRS:%=-I%) $(FLAGS) $(DFLAGS) -O3 $(LIBRARIES) -o  
"build/$@" "$<" --std=c++11  
@echo 'Building finished: $@'
```

For GPU implementation

Use the following commands

```
$> cd $TRITON  
$> make clean && make hpc_gpu
```

If successful, the executable `triton` will be located in the `$TRITON/build` folder.

For CPU (OpenMP) implementation

Use the following commands

```
$> cd $TRITON  
$> make clean && make hpc_omp
```

If successful, the executable `triton` will be located in the `$TRITON/build` folder.

4. Running the Executable

The basic command to run the simulation is:

```
$> cd $TRITON
$> ./build/triton $CFG_NAME
```

where,

`$CFG_NAME` = Config file name

4.1 OLCF supercomputer – Summit

```
$> cd $TRITON
$> jsrun -n $N -r $R -a $A -g $G -c $C --smpiargs="-gpu" \
    ./build/triton $CFG_NAME
```

where,

`$N` = Number of total resources

`$R` = Number of resources per node

`$A` = Number of MPI process per resource

`$G` = Number of GPU per resource

`$C` = Number of CPU per resource

`$CFG_NAME` = Config file name

4.2 Non-OLCF machine

```
$> cd $TRITON
$> ./build/triton $CFG_NAME
```


5. Directory Structure and Features

5.1 Directory Structure

The main directory structure and contents are described in this section.

```
$TRITON      : top level base directory
|--build     : contains the executable
|--doc       : documentation (User guide, Reference Manual)
|--input     : input data, parameters and forcing directory
  |--cfg     : configuration file directory
  |--dem     : elevation data directory
  |--extbc   : external boundary conditions directory
  |--initc   : initial depths and unit discharge conditions
    |--inith
    |--initqx
    |--initqy
  |--mann    : manning's roughness coefficient parameters
  |--stageloc : user defined locations at which timeseries water
                elevation output is desired
  |--runoff  : runoff map and timeseries forcing
  |--strmflow : streamflow hydrograph and locations
|--output    : output data is generated here
|--src       : contains the source code
|--tools     : contains miscellaneous utilities for pre and
                post processing data for TRITRON
```

6. TRITON Configuration

The TRITON simulation can be configured and controlled by the configuration file which is located in the `$TRITON/input/cfg` subdirectory. The configuration file is a text file used to control various simulation options such as duration, parameters, paths to input and output, etc.

As a new user we recommend you explore one of the pre-existing configuration files to get acquainted. To configure your own test case, use one of these files as a template and make appropriate changes as necessary. A quick walk-through of main contents of the configuration file is provided.

6.1 Topography

- `dem_filename`: Path where DEM is located.

Example Usage:

```
dem_filename="input/dem/asc/dem_file.dem"
```

TRITON accepts topographical data in [Esri ASCII raster format](#). This file contains first six rows as header information, followed by a space delimited matrix of `nrows` x `ncols` (an example snapshot of the file is shown below). This serves as the base computational domain for the hydraulic simulations where the number of rows and columns defines the size of the computational domain.

```
1 ncols.....3159
2 nrows.....2394
3 xllcorner....218995.2218
4 yllcorner....3278483.5008
5 cellsize....29.4241
6 NODATA_value..-9999
7 98.91084 98.72333 98.53819 98.21210 98.00509 97.88680 97.60
8 98.63085 98.64519 98.34378 98.12985 97.84086 97.65837 97.30
9 98.34380 98.18010 98.16917 97.87180 97.67137 97.55568 97.30
10 98.29378 98.05849 97.70802 97.66949 97.47814 97.19872 97.10
11 98.09411 97.94952 97.62103 97.38382 97.18198 96.92780 96.90
12 97.88176 97.75722 97.51286 97.18246 97.12796 96.70582 96.40
13 97.70827 97.41485 97.18679 96.94337 96.75169 96.55964 96.70
14 97.69495 97.18163 97.06666 96.70103 96.64828 96.36138 96.40
15 97.32288 97.05244 96.67410 96.47199 96.24435 96.21249 96.10
16 97.00000 96.00000 96.00000 96.00000 96.00000 96.00000 96.00
```

Refer to `$TRITON/input/dem` to explore provided DEM examples. When designing a simulation, users must exercise caution to avoid having negative NODATA values (e.g. `NODATA_value = -9999`) in the topographical data, as they will act as large depressions during the hydraulic simulations.

6.2 Surface Roughness (Manning's n Coefficient)

- `n_infile`: location of manning's roughness file

Example Usage:

```
n_infile="input/mann/asc/roughness_file.mann"
```

TRITON accepts spatially varying Manning's n coefficient to account for surface roughness. The roughness should be defined explicitly for each grid cell of the computational domain. Therefore, this

file follows the same template as of the DEM file (explained above) *excluding the header* (i.e., ESRI ASCII raster format, but *without the header*).

- `const_mann`=user defined constant Manning's n value

Example Usage:

```
const_mann=0.035
```

Alternatively, user can also decide to provide a constant Manning's n value to the entire computational domain.

6.3 Hydrologic Forcing

TRITON can be driven by either providing streamflow hydrographs at specified locations or gridded runoff hydrograph, or both which serve as the model's hydrological forcings.

Streamflow Hydrograph

- `hydrograph_filename`: file location of streamflow hydrograph

Example usage

```
hydrograph_filename="input/strmflow/streamflow.hyg"
```

This file contains the streamflow hydrograph information. This data is in form of a table including the time (in hours), as first column. This is followed by columns of inflow sources (equal to number of inflow locations), containing the discharge for each source. *The units must be in cubic meters per second (m^3/s).*

- `num_sources`: number of inflow points/locations

Example usage

```
num_sources=2
```

- `src_loc_file`: file with cartesian coordinates of inflow locations.

Example usage

```
src_loc_file="input/strmflow/location.txt"
```

This file contains cartesian coordinates of all the inflow locations, in the order they appear in the `hydrograph_filename`.

Runoff Hydrograph

Users can also decide to provide runoff as an alternative or along with streamflow as an input hydrologic forcing to the TRITON.

- `runoff_filename`: file location with runoff hydrograph

Example usage

```
runoff_filename="input/runoff/runoff.hyg"
```

This file contains the runoff hydrograph information. This file contains data in form of a table including the time (in hours), as first column. This is followed by columns of runoff sources (equal to number of inflow locations), containing the discharge evolution for each source. *The units should be in mm per hour (mm/hour).*

- `num_runoffs`: number of distinct runoff areas

Example usage

```
num_runoffs=156
```

- `runoff_map`: matrix raster map of distinct areas

Example usage

```
runoff_map="input/runoff/runoff.rmap"
```

This file follows the same template as of the DEM file (explained above) *excluding the header* (i.e., ESRI ASCII raster format, but *without the header*). The file is a spatial map of distinct runoff areas defined by a non-negative integer number for each grid cell in the computational domain.

A sketch of the rainfall/runoff input files is depicted below, for a more detailed explanation please refer to Morales-Hernández et al., XX or explore the input files of test case #3.

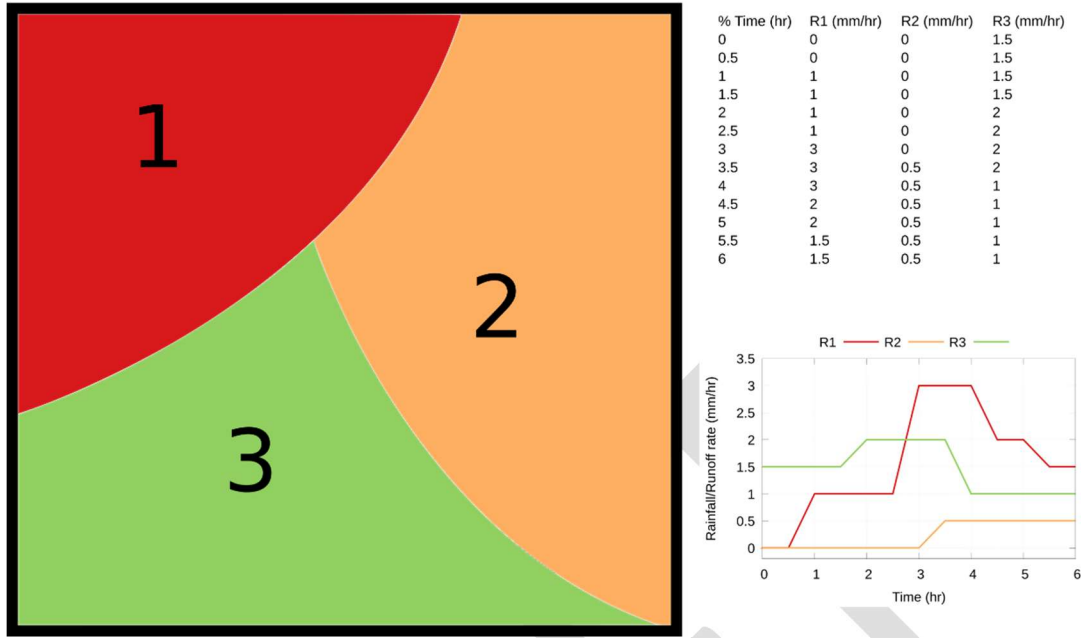


Figure 3 from Morales-Hernández et al., XX

- `runoff_row_size`: number of rows in the `runoff_filename` [Will be removed in the final release]

6.4 External Boundary

- `num_extbc`: number of external boundary conditions

Example usage

```
num_extbc=0
```

If `num_extbc = 0`, by default, boundaries of the computational domain (north, east, south and west) are assumed closed (i.e., water flow is prevented to cross out the domain)

If `num_extbc = 1 or greater`, the boundary conditions are controlled by `extbc_file`

- `extbc_file`: file to control the external boundary conditions

Example Usage

```
extbc_file="input/extbc/boundary.extbc"
```

In this file, the first column represents boundary condition type. Currently, boundary can acquire one of the following values `0, 1, 2 or 3`. A brief description of each type is given below

`0`: free flow (supercritical). No additional files are required.

`1`: level versus time. An additional file containing a table with the time and the water level is mandatory.

2: normal slope. The desired slope value is required in the last column.

3: Froude number. The Froude number to be imposed across the external boundary is required in the last column.

The second through fifth column represents X and Y coordinates of the two points between which a boundary is to be implemented.

An example of `extbc_file` is given below:

```
1 % BC Type, X1, Y1, X2, Y2, BC
2 1, 219010, 3314425, 219010, 3308452, "level.txt"
3 0, 311934, 3348880, 311934, 3305245
4 3, 311934, 3286946, 311934, 3278500, 0.25
5 2, 305870, 3278500, 310784, 3278500, 0.001
6 |
```

6.5 Simulation Control

- `sim_start_time`: start time of simulation in seconds

Example Usage

```
sim_start_time=0
```

- `sim_duration`: duration of simulation in seconds

Example Usage

```
sim_start_time=43200
```

- `checkpoint_id`=checkpoint id.

The checkpoint id can be used to restart a simulation. The checkpoint id is a counter written in the file `$TRITON/cid` once the simulation starts. The counter increases after every `print interval`. This is especially helpful if the simulation was not complete for any reason, and user would like to start from a particular point to avoid rerunning the entire simulation. The default value is zero, assuming a clean start.

To restart the simulation from a checkpoint, user must output all `huv` maps in the `print_option` as described in the Output section

Example Usage

```
checkpoint_id = 24
```

- `time_increment_fixed`= switch to control the time step.

Example Usage

```
time_increment_fixed=0 for variable dt  
time_increment_fixed=1 for constant dt
```

If `time_increment_fixed=1`

Then use `time_step` variable to control the timestep size. The units are in seconds.

Example Usage

```
time_step=0.01
```

- If a user is interested to start the simulation using initial condition, the following maps can be provided.

```
h_infile = initial water depth file  
qx_infile = initial x direction unit discharge file  
qy_infile = initial y direction unit discharge file
```

Example Usage

```
h_infile="input/inith/asc/case5.inith"  
qx_infile="input/initqx/asc/case5.initqx"  
qy_infile="input/initqy/asc/case5.initqy"
```

6.6 Output Control

- `print_option`= switch to govern the type of output

Example Usage

```
print_option=h % to output water elevation maps only  
print_option=huv % to output water
```

A map for the water depth (`h`) and unit discharges (`uv`) is written in the form of an ESRI ASCII raster file (without header) as controlled by `print_interval`. This information can be written either in ASCII or in binary –the latter is recommended for large scale domains.

- `print_interval`= number of seconds after which the outputs will be saved.

Example Usage

```
print_interval = 1800 % outputs will be saved every 1800 seconds  
in the simulation, throughout the duration of simulation.
```

- `time_series_flag`= switch to activate or deactivate the output of water stage hydrographs at user defined locations.

Example Usage

```
time_series_flag=0 % to deactivate this option
time_series_flag=1 % to activate
```

If `time_series_flag=1` provide information about `observation_loc_file`

- `observation_loc_file`: file containing coordinates of the user-defined locations where output is desired.

6.7 Input Output File Settings

The model deals with two different type of file types – ASCII or Binary. The type of input or output files can be controlled in this section. A conversion tools between ascii and binary file types is detailed in Section 8. The user can decide suitable file type using the following controls.

Example Usage

```
input_format=ASC # Possible options BIN or ASC
output_format=ASC
```

The user can decide suitable file type using the following controls.

```
outfile_pattern="%s/%s/%s_%02d_%02d" # controls file naming
convention
output_option=PAR # Possible options PAR or SEQ
```

A switch in the configuration file allows the user to choose the mode in which the data is written.

`SEQ` sequentially, i.e., gathering all the subdomains in a single file during the computation

`PAR` in parallel

6.8 Other Miscellaneous parameters

- `courant` = Courant-Friedrichs-Lewy (CFL) number

Example Usage

```
courant=0.5
```

According to the stability condition, this value should not exceed 0.5

- `hextra` = water depth tolerance value below which velocities are set to zero

Example Usage

```
hextra=0.001
```


- `gpu_direct_flag`= this parameter should be set to `0` unless your system has multiple GPUs with CUDA-Aware MPI implemented on it. In that case, the value can be set to `1` to get maximum performance

Example Usage

```
gpu_direct_flag=0
```

DRAFT

7. Test Cases Overview

A set of test cases are provided with the TRITON release, detailed in Morales-Hernández et al., in preparation. These test cases provide examples of real-world like examples and are aimed to highlight different capabilities of the model and its performance. The configuration file for each case is available at :

```
$TRITON/input/case01.cfg : Case 1 - TaumSauk Dam Failure
$TRITON/input/case02a.cfg : Case 2a - Paraboloid (0.04m)
$TRITON/input/case02b.cfg : Case 2b - Paraboloid (0.02m)
$TRITON/input/case02c.cfg : Case 2c - Paraboloid (0.01m)
$TRITON/input/case02d.cfg : Case 2d - Paraboloid (0.005m)
$TRITON/input/case03.cfg : Case 3 - Runoff + Streamflow
$TRITON/input/case04.cfg : Case 4 - Hurricane Harvey (30m)
$TRITON/input/case05.cfg : Case 5 - Hurricane Harvey (10m)
```

Note : Please note that Cases 1 and 2a are ideal for quick testing on single CPU based machines. Case 3, 4 and 5 are medium to large scale cases only suitable for HPC machines or GPU based machines.

Case 1 : This test case is an example of dam failure simulation. For more details refer to Kalyanapu et al., 2011.

Case 2 : Test case 2a through 2d is a test case with a frictionless paraboloid topography using four different grid resolutions of 0.04, 0.02, 0.01 and 0.005m. This test case is detailed in Section 5.1 of Morales-Hernández et al. (in preparation).

Case 3 : Test case designed to simulate a 5-day flood event, using both streamflow and runoff as the hydrologic drivers. This test case is detailed in Section 5.2 of Morales-Hernández et al. (in preparation).

Case 4 : Test case designed to simulate a 10-day flood event during Hurricane Harvey over Harris County, TX, with a 30 m grid resolution. This case is a variant of Case 5 which uses 10 m grid resolution.

Case 5 : Test case designed to simulate a 10-day flood event during Hurricane Harvey over Harris County, TX, with a 10 m grid resolution. This test case is detailed in Section 5.2 of Morales-Hernández et al. (in preparation).

On successful building of the executable, any of the given cases can be used. On successful simulation execution, users can expect to see similar messages on their screen.

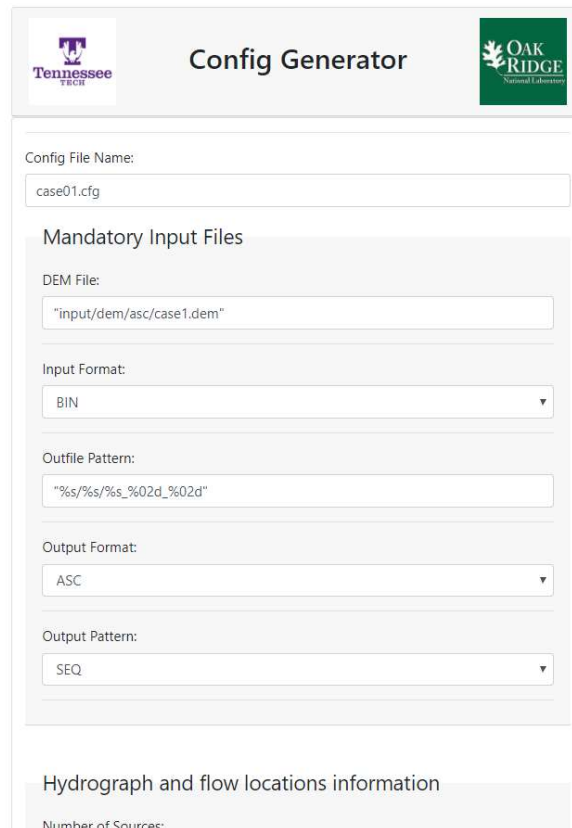
```
bash-4.2$ jsrun -n 6 -r 6 -a 1 -g 1 -c 1 ./build/triton "input/cfg/case02a.cfg"
[...] Reading configuration file
[OK] Configuration file read
[--] No sources defined
[--] No runoff defined
[--] No external boundary conditions defined
[--] 800 observation points defined
[...] Reading file input/dem/asc/case02a.dem
[OK] File input/dem/asc/case02a.dem read
[...] Reading file input/initc/inith/asc/case02a.inith
[OK] File input/initc/inith/asc/case02a.inith read
[...] Reading file input/initc/initqx/asc/case02a.initqx
[OK] File input/initc/initqx/asc/case02a.initqx read
[...] Reading file input/initc/initqy/asc/case02a.initqy
[OK] File input/initc/initqy/asc/case02a.initqy read
[...] Creating partition data
[OK] Data has been partitioned
[OK] Simulation starts
[1] Time: 1.12142      dt: 2.0875e-06  it: 80941
[2] Time: 2.24285      dt: 4.99999e-06 it: 193084
[3] Time: 3.36427      dt: 4.99999e-06 it: 305227
[4] Time: 4.4857       dt: 5.00001e-06 it: 417370
[5] Time: 5.60712      dt: 5.00004e-06 it: 529513
[6] Time: 6.72855      dt: 5.00004e-06 it: 641656
[7] Time: 7.84997      dt: 5.00004e-06 it: 753799
[8] Time: 8.9714       dt: 5.00004e-06 it: 865942
[9] Time: 10.0928     dt: 5.00004e-06 it: 978085
[10] Time: 11.2142     dt: 5.00004e-06 it: 1090228
[11] Time: 12.3357     dt: 5.00004e-06 it: 1202371
[12] Time: 13.4571     dt: 5.00004e-06 it: 1314514
[OK] Simulation ends
bash-4.2$
```

8. Other Tools

A set of utility tools are provided to ease the pre and postprocessing of the input/output files. These tools are located in `$TRITON/tools/`

8.1 Configuration File Generator

This tool provides a graphical user interface to create the configuration file for TRITON. It is currently supported only for the Windows machines. To use this tool, run the executable “main” from the following location `$TRITON/tools/ConfigGenerator/Windows EXE/`. When launched successfully, this executable will open a localhost web browser and show a user interface. Provide necessary information (as shown below, as an example), and click Generate Config button to download the newly created configuration file. Alternatively, users can start with one of the pre-existing configuration files from the test cases and modify per their needs.



8.2 Ascii to Binary Conversion

This tool converts ascii files (without header) to binary format.

```
$> cd $TRITON/tools/ascii2bin
$> module load gcc
$> make clean && make
$> ./ascii2bin $INPUT_DIR $OUTPUT_DIR $TYPE
```

Where,

`$INPUT_DIR` = Input file/folder directory of Ascii files

`$OUTPUT_DIR` = Output file/folder directory of Binary files

`$TYPE` = 1 for single file conversion , 2 to convert all the files in the input folder

If the ascii files contains header, use the `$TRITON/tools/ascii2bin_dem` folder.

8.3 Binary to Ascii Conversion

This tool converts binary files to ascii format.

```
$> cd $TRITON/tools/bin2ascii
$> module load gcc
$> make clean && make
$> ./ascii2bin $INPUT_DIR $OUTPUT_DIR $TYPE
```

Where,

`$INPUT_DIR` = Input file/folder directory of Ascii files

`$OUTPUT_DIR` = Output file/folder directory of Binary files

`$TYPE` = 1 for single file conversion , 2 to convert all the files in the input folder

8.4 Discharge to Velocity Conversion

This tool enables conversion of discharge files to velocity.

```
$> cd $TRITON/tools/discharge2velocity
$> module load gcc
$> make clean && make
$> ./discharge2velocity $INPUT_DIR $FILE_TYPE $TYPE
```

Where,

`$INPUT_DIR` = Input file/folder directory of discharge

`$OUTPUT_DIR` = Output file/folder directory of velocity

`$FILE_TYPE` = 1 for Ascii, 2 for Binary

`$TYPE` = 1 for file, 2 for folder

8.5 Join Utility

If users enable `output_option=PAR` , the join utility can be used as a postprocessing tool, to create a merged ascii outputs i.e. a single output file for every print interval. The input files should be located in the `$TRITON/output/asc` folder.

8.6 ASCII/Binary to NetCDF Conversion

This tool located in `$TRITON/tools/NetCDFconverter`, allows users to convert the files from ASCII/Binary format to NetCDF format.

Windows:

Execute the “main.exe” file.

Linux:

For Linux based implementation, following dependencies are required.

Python3, Python-netCDF4, Python-eel, Python-Tkinter

```
$> cd $TRITON/tools/NetCDFconverter/Linux EXE/  
$> ./main
```

To convert a file, first select one of the “ASCII to NETCDF” or “Binary to NETCDF” button on the top (as shown in the example below). Next, selected the input file type using the dropdown button. The option ‘DEM’ refers to files with header information. If the header is missing from the input file, use the ‘other’ option. Select the input file using the “Select File” Button. In case of DEM file, the dimension and attributes will be obtained from the header automatically. User can provide the “attribute name”, which will be the variable in the NetCDF file and projection file information. Finally clicking “Generate” button will allow user to navigate to deseired directory and save the NetCDF file.

The screenshot shows a web application titled "Triton ASCII/Binary to NetCDF Conversion". At the top, there are logos for Tennessee Tech and Oak Ridge National Laboratory. Below the title, there are two buttons: "ASCII To NetCDF" (highlighted in green) and "Binary To NetCDF". The main form area contains several sections: 1. "File Type:" with a dropdown menu currently set to "DEM". 2. "Input File ? :" with a "Select File" button. 3. "Attribute Name ? :" with an empty text input field. 4. "Dimension Info ?" section with two empty text input fields, a red "x" icon, and an "Add Dimension" button. 5. "NetCDF Attribute ?" section with two empty text input fields, a red "x" icon, and an "Add Attribute" button. 6. "prj file:" with a "Select File" button. At the bottom of the form is a large green "Generate" button.

9. References

- Kalyanapu, A. J., Shankar, S., Pardyjak, E. R., Judi, D. R., & Burian, S. J. (2011). Assessment of GPU computational enhancement to a 2D flood model. *Environmental Modelling & Software*, 26(8), 1009-1016.
- Morales-Hernández, M., M. B. Sharif, A. Kalyanapu, S. K. Ghafoor, T. T. Dullo, S. Gangrade, S.-C. Kao, and K. J. Evans (2020), TRITON: A multi-GPU open source 2D high resolution flood model, in preparation.

DRAFT