TUFLOW GPU Workflow.

This is the workflow for the “RosgenC4” scenario with Tuflow Classic or Tuflow GPU.

Dated September 11, 2018

Written by Prof. Gregory Pasternack

Thanks for assistance from Chelsea Hopkins, Paulo Silva, and the Tuflow support staff.

The purpose of this workflow is to document the exact steps used to make a 2D model for Tuflow GPU, using a simple case example of a U-shaped channel along with two topographic variants for different kinds of riffle-pool couplets. The workflow is written generically and may be used by anyone. However, this is not intended as a comprehensive lesson in Tuflow. It is a good starting point and a good reference to customize for your own workflow. Following these steps exactly will work, but after that it would be wise to take advantage of the tutorial videos and webpages at <http://www.tuflow.com>. In addition, you should download the PDF of the Tuflow manual and have that as a reference on your computer. That manual has appendices (see hyperlinks at the bottom of the title page) that show all the commands and how to set their options.

Because many people new to Tuflow have not committed to purchasing it, this exercise is designed to work using the DEMO model for Tuflow Classic, so you can “test drive” the software.

# Setting up your computer for Tuflow

## Follow the guidance at this link to obtain and set up Notepad++ on your computer and add the extras and plugins that make Tuflow work best. Be sure to do the steps under the Basic section to add syntax highlighting and file navigation. <http://wiki.tuflow.com/index.php?title=NotepadPlusPlus_Tips>

## Click on this link to download an ArcGIS toolbox that has Tuflow commands. Follow the instructions to install this into your local ArcGIS. <http://www.tuflow.com/Tuflow%20Utilities.aspx>

## Follow the link and instructions at this webpage to get the Tuflow\_to\_GIS.exe utility program. This is most useful for producing velocity vector output you can use in ArcGIS. <http://wiki.tuflow.com/index.php?title=TUFLOW_to_GIS>

## Go to the NVIDIA driver website at <http://www.geforce.com>, download the NVIDIA GeForce Experience installer, and install that. After installation, use that program to update the NVIDiA driver to the latest version. In the future it will alert you when updates are available.

## If you have purchased a license and have a hardware dongle, then download and install the Codemeter software from <http://www.tuflow.com/Tuflow%20Latest%20Release.aspx> so that the computer can find the Tuflow dongle.

# Preparation in Windows Explorer

## Choose a model project prefix that captures the essence of what the model is for. This workflow will call it Name\_. If the river area to be modelled is fixed for all scenarios that will be evaluated, as it should be, then I suggest naming with an abbreviation for the area of the river. I do not recommend using the existing name of a geomorphic reach, as this could create confusion when the model domain is not an exact match to the reach area. Best to use a different naming convention for model domains versus river reaches and segments.

## For this exercise we will begin with the scenario called “VanillaC4”, so for all steps to follow until told otherwise, substitute “VanillaC4\_” in place of “Name\_”. I’ll explain more about this scenario later.

## Make sure you have the latest release of Tuflow and you put that downloadable folder in C:\Program Files\. Name the folder that has the Tuflow program files as Tuflow\_w64.

## Copy the entire folder structure called Name\_Tuflow\_template\_folders from the Tuflow folder on the file server and paste it into your desired project location. Rename this folder as RosgenC4\_tuflow.

## I have provided you with a folder named, “0\_datainputs”. This folder has three topographic scenarios. In this exercise you will run models of all 3 of these with different settings to gain some appreciation for issues related to 2D modeling of rivers. These example DEMs are purely artificial constructs. They are small enough that they can be used with Tuflow’s DEMO mode with no paid license.

## We will begin with the VanillaC4 channel scenario using the files in the “VanillaC4\_channel\_inputs” folder. I define a *vanilla channel* as one that has all the correct reach-scale values for any particle type of river, but it has no sinuosity and no topographic variability functions in it. It is essentially a canal. No offense to vanilla, but I think it is important to understand that most river classifications only offer vanilla channels when taken literally. Many of the unique features of channel types do not come from their reach-scale attributes, but from their layers of complexity added on top of them. This will be demonstrated on a preliminary basis using the two other scenarios after you complete the modeling of the vanilla channel.

## The reach-scale attributes of this topographic scenario come from real data representing a “C4” channel in the Rosgen (1986) classification system. After we model vanilla version of it in this exercise, then you can repeat this whole exercise but swapping out the “in phase” and “out of phase” channels that have linked bed elevation vs width geomorphic covariance structures. I have provided the native “synthetic river valley” Excel files that show the origin of all three of these scenarios, along with the RiverSynth1.1 user’s manual that explains how those work. The bottom line is that here are three wildly different topographies that all meet the same definitional criteria for a Rosgen C4 channel (except sinuosity, which was omitted for simplicity), yet as you’ll see form running them with a 2D model, the reach-scale attributes have litle to do with their hydrogeomorphic functionality.

## Navigate to the polygon SHP file of the boundary of the DEM you have for your project, in this case “VanillaC4\_boundary.shp”. Select and copy the .prj file for that polygon and paste it into the subfolder model/gis within RosgenC4\_tuflow. Rename the pasted file as Projection.prj.

## Right click and edit the file runs/NAME\_run\_001\_TUFLOW.bat within Name\_tuflow and make sure that the path in the first line is the correct one to get to the 64-bit (assuming you have a 64-bit computer, otherwise aim for 32-bit) Tuflow\_iSP (single precision) .exe program file. If not, write the correct path. Save and close this file. Do not replace Name\_ in the name of this .bat file; just leave it as is. Do not change the name of the file Name\_001.tcf.

## Double click on NAME\_run\_001\_TUFLOW.bat.

## If this step worked, you have now created a folder within model/gis called empty that contains a whole bunch of .SHP files. These are the template files you will need later in this workflow, and they are all set up with the correct projection for your project. If this step did not work, then re-affirm that the Tuflow .exe file is in the correct location pointed to in the .bat file.

## In Windows, navigate into the model/gis/empty folder. Copy all the files for the following SHP files and paste them one level up in model/gis: 2d\_bc\_empty\_L, 2d\_code\_empty\_R, 2d\_loc\_empty\_L, 2d\_mat\_empty\_R, 2d\_po\_empty\_L, 2d\_po\_empty\_P, 2d\_sa\_empty\_R. In these names, L=line, R=region=polygon, P=point.

Take note that this exercise is written generically to work with all your future models that do basic 2D modeling, so some of the steps will seem unnecessary or more complex than warranted for these simple examples. This is intentional so you can retain and use this workflow for substantially more challenging situations.

# Preparation in ArcGIS

## Open ArcGIS and load in the following files from the VanillaC4 folder:

1. Polygon SHP file of the boundary of the DEM.

Load this first to set the projection for ArcGIS.

Polygon SHP file with the Manning’s n polygons (only needed if you plan to spatially distribute Manning’s n with polygons). In this exercise we will use a constant Manning’s n, so do not worry about this.

1. Aerial imagery (if you have it… strongly recommended). Not needed for this exercise.
2. Green LiDAR intensity imagery (if you have it). Not needed for this exercise.
3. DEM raster file in the max resolution you want to make a 2D model for. In this exercise, load the .asc version of the DEM.

## The DEM must be in the same projection as the polygon shapefile, so if it is not, then perform a raster transformation to get it into the correct projection.

## Save your ArcGIS project as RosgenC4.mxd, and I recommend saving it at the top level within 0\_datainputs. Best to not get that mixed up with the tuflow file structure.

## The next step would normally be to create a polygon SHP file that reflects your initial expectation of the wetted area of the final answer to the simulation. This is later going to be called the "code" area in Tuflow and also is referred to as the active area of the computational mesh. You want to overestimate the expected wetted area and avoid any risk of underestimating it. In this artificial example with RosgenC4, the model domain is relatively small and for starters you have no idea what the wetted area might be. Therefore, you will use the provided “VanillaC4\_boundary” polygon for this purpose. Later you’ll refine the model anyway, so for not it is not too important to be very precise with the model domain, as long as you do not exceed the number of elements permitted for the DEMO mode, if you do not have your on license yet. In creating the wetted area polygon it is important that the polygon consists of straight line segments at each inflow and outflow boundary, because later we will create flow boundary lines and polygons that will likely be snapped to these features. Therefore, interpret how you draw these boundary line segments to be perpendicular to the expected flow. If you use any arbitrary DEM boundary, and if the inflow and outflow boundaries are not straight lines, then you’ll have to edit the flow boundaries as indicated while also keeping the polygon within the extent of the DEM. However, for this exercise the inflow and outflow lines boundaries are already simple straight lines.

## For the file 2d\_code\_empty\_R.shp in model/gis, now rename \_empty\_ to \_Name\_, where that is \_VanillaC4\_ in this case as one last reminder. Add to your ArcGIS map.

## Start Arc’s Editor tool and set the target for editing to 2d\_code\_Name\_R.shp.

## Use Editor’s Edit tool to select the polygon that reflects your initial expectation of the wetted area of the final answer to the simulation, right click on it, select Copy from the contextual menu, right click again, select paste, and then select the option to paste this polygon into 2d\_code\_Name\_R.shp.

## Open the attribute table for 2d\_code\_Name\_R.shp. Select and delete the first row out of it, which is an empty placeholder that has no purpose for Tuflow at this point. Arc does not allow truly empty SHP files, so all the empty files have one empty row that you will delete out each time you set up a file for use with Tuflow. This is not strictly required in recent model releases, but it is best to do it to have all files clean, simple, and understandable.

## Now in the remaining row that has the actual “code” polygon, change the value of the Code field form 0 to 1. A value of 1 means that all model cells within the code polygon will be active in the model. Save edits and stop editor.

## This next step is not required for this exercise, because the boundary is already exactly a rectangle, so just read and take note of it. This is very important when you have a non-rectangular expected wetted area. Note that the entirety of the rectangle for the final model must be within the DEM. Use ArcGIS to create the most compact rectangular polygon SHP file fitting your overestimated active model area (aka code area) and still within the DEM. To do this, run the ArcGIS "Minimum Bounding Geometry" tool. Choose your active flow area polygon as the input feature. Give your output file a name such as NAME\_coderectangle\_R.shp, where NAME is your model's file name prefix. Choose the geometry type of "Rectangle\_by\_width". Save this file in a the vanilla\_channel\_inputs folder.

## This next step is not required for this exercise, because the boundary is already exactly a rectangle, so just read and take note of it. Run the ArcGIS "Buffer" tool. The input file is NAME\_coderectangle\_R.shp, the output file is NAME\_codepoly\_buffer\_R.shp. Set the linear distance to something like 100 ft or 50 m for a typical case- just enough to insure that there is extra room around your anticipated active area so the active area is well within the DEM. Choose side type FULL and Method PLANAR. Note that the output SHP file has rounded ends, which is not desirable.

## This next step is not required for this exercise, because the boundary is already exactly a rectangle, so just read and take note of it. Re-run the "Minimum Bounding Geometry" tool, but this time set the input file as NAME\_codepoly\_buffer\_R.shp and name the outpfile as Name\_coderectangle\_buffer\_R.shp. This will create an exterior rectangle around codepoly, so this will be the final total model domain as we move forward.

## Normally the next step would be to use Arc’s measure tool to measure the X and Y dimensions of Name\_coderectangle\_buffer\_R.shp and write those values down in your project notes file for later reference. For this RosgenC4 exercise, do this step using 2d\_code\_VanillaC4\_r. The dimensions should be 762 “long” (from right to left) and 129 “wide” (from top to bottom). Note that I am rounding up to the nearest whole number as the grid has to be an integer size. Also, note that the flow is going to be from right to left, which you can tell from the slope implied in the DEM.

## For the file 2d\_loc\_empty\_L.shp in model/gis, now rename \_empty\_ to \_Name\_. Add to your ArcGIS map.

## In ArcGIS, start editor and choose to edit 2d\_loc\_Name\_L.shp. Set the snapping options for editor to points and vertices only; make sure edge snapping is off. This step is going to set the geographical origin and orientation of the grid based on a line we create now. Normally you would have this line oriented along the file, Name\_coderectange\_buffer\_R.shp. However, because this RosgenC4 exercise is perfectly oriented to flow from right to left, you’re going to orient the model by drawing a line from the bottom left to the bottom right along th VanillaC4\_boundary polygon. Use snapping.

## Open the attributes table for 2d\_loc\_Name\_L.shp and delete out the empty polygon in the first row. Recall that Arc cannot have truly empty SHP file, we delete the empty row in all these empty SHP files once we have created the proper elements needed in it. Save edits and stop editor.

## This next step is not required for this exercise, because the boundary is already exactly a rectangle, so just read and take note of it. This next step is necessary when you have a ginormous DEM and it is not necessary to load all of that into the model and have such big files. This step also converts any generic DEM into an .asc formatted DEM. For this exercise, our DEM is very small, so this step can be skipped- except it is now time to you’re your .asc DEM (including its .prj file) and paste the copy into the folder /model/grid. Add the project DEM raster file to ArcMap. If you have not already, clip it to the coderect\_buffer boundary for this project. Use the conversion tool Raster To ASCII. Set the name of your choice and make sure the file extension is .asc. Set the output folder location to /model/grid.

## For the file 2d\_bc\_empty\_L.shp in model/gis, now rename \_empty\_ to \_Name\_HT\_. Add to your ArcGIS map. This file will contain your downstream flow exit line (L) boundaries that will be defined with a water surface elevation (H) through time (T).

## This next step is not required for this exercise, because the boundary is already exactly a rectangle, so just read and take note of it. If you have Green LiDAR intensity imagery, turn it on now as the base imagery. You only really need to see the images at the flow boundaries, so if you have to load it from files, just load those, identifying them from the LiDAR index SHP file.

## Start Editor and choose to edit 2d\_bc\_Name\_HT\_L.shp. Set the snapping options for editor to points and vertices only; make sure edge snapping is off. Create lines in this file snapped to the downstream outflow boundaries defined by 2d\_code\_Name\_R.shp. For the RosgenC4 exercise, there is only one output and we will begin by treating the entire length of the downstream code rectangle boundary as the downstream flow boundary. That way the exit water level boundary will be exactly snapped to the active model domain along the boundary. The model can solve to have the actual water level lower than the topo across the whole boundary, so some of the boundary cells are anticipated to be dry, and that is fine. The directionality of how you draw the line does not matter. For this exercise, it needs to go across the bottom boundary of the wetted area rectangle.

## Open the attributes table for 2d\_bc\_Name\_HT\_L.shp and delete out the empty polygon in the first row. Set the type to HT for each outflow boundary and write in a name for each. For the RosgenC4 exercise, use the name, “RPout”. Take note of the names for later use. Save Edits and Stop Editing.

## For the file 2d\_sa\_empty\_R.shp in model/gis, now rename \_empty\_ to \_Name\_QT\_. Add to your ArcGIS map. This file will contain your upstream inflow region (R) boundaries that will be defined with a discharge (Q) through time (T). In some applications, this file will also contain outflow region boundaries that have a known exit discharge. Tuflow GPU cannot use lines for the inflows at this time, so a polygon (aka region, R) file is used.

## Start Editor and choose to edit 2d\_sa\_Name\_QT\_R.shp. Set the snapping options for editor to points and vertices only; make sure edge snapping is off.

## For each inflow and outflow boundary defined by 2d\_code\_Name\_R.shp, create a polygon that has one boundary along the flow boundary, but then has a width away from the flow boundary as well. For the RosgenC4 exercise, there is only one inflow boundary and it as along the right side of the domain. We will only be modeling flow in the channel, and you can see that the channel makes up the central 1/4 of the DEM, with a floodplain surface on either side of the channel. If you load and view the file “VanillaC4\_XYZpts” , then you will see all the channel points used to make the DEM, but the outermost lines of points are at the floodplain elevation and beyond. You will want to make your inflow boundary polygon to contain the full width of the bankfull channel, except say the outer 1 point on each side of the cross-section. These polygons do not have to be snapped to the boundary line, but they can be. The main thing is that it is critical that each inflow boundary polygon contains the cell center of at least one computational grid cell. Water flows from the cells whose midpoints are within a polygon. Since you have not yet created a computational mesh, you cannot be sure whether this is true or not, so just make sure that the boundary is relatively thick compared to the cell size you plan to use. In the RosgenC4 exercise, since the points are roughly 1 m apart, if you set the thickness of the inflow region to snap to the eleventh point downstream from the inflow boundary, then it will be roughly 11 m thick and that is good for our first trial run, which will use a 10-m grid size. We will adjust flow boundaries later when we check and refine the model. Note that there is no harm if some of these boundary cells ought to be dry, as the boundary condition algorithm distributes water to the deepest cells first and can turn off cells that are too high to be within the flow area. These regions should be created roughly perpendicular to flow as the model assumes that the water level is about constant within the region.

## Open the attributes table for 2d\_sa\_Name\_QT\_L.shp and delete out the empty polygon in the first row. Write in a name for each polygon, in this case use “RPin”. Take note of the names for later use. Save edits and stop Editor.

## In Windows, navigate to the file /bc\_dbase/2d\_bc\_NAME.csv. Type in the name of your project in place of NAME in the file name, again this is vanilla. Open this file in MS Excel. As a starting point, the template file has 3 inflow and 3 outflow rows, so you must delete or add rows as needed for your model. For the RosgenC4 exercise, we only have one inflow and one outflow boundary. In the first column, write in the names of all of your boundary lines and regions being sure to exactly match the name field in the attributes tables of your BC and SA SHP files. Copy and paste these names into column D, which has the header “Column 2”. In the column with the header “Source”, type in the name of your project in place of NAME in each row. Save, keeping the format as .csv.

## In Windows, navigate to the file /bc\_dbase/NAME\_bc\_data.csv. Type in the name of your project in place of NAME in the file name. Open this file in MS Excel. This file contains the actual flow and water surface elevation data to go with the simulation. For a steady state simulation, use the same values for all time entries. For an unsteady simulation, write in the time series for how flow changes through time, with time given in hours. As a starting point, the template file has 3 inflow and 3 outflow columns plus a time column, so you must delete or add columns as needed for your model. For the RosgenC4 exercise, we only have one inflow and one outflow boundary. Replace the header row cell entries with the correct names of each inflow and outflow boundary line or region as needed. Replace the existing numbers in there with your correct numbers for your model runs. Refer to the README\_first file to find out this information, but I’ll save you the time and it is 121.46 m3/s for inflow and 1003.432 m for water surface elevation. These values are close to the analytically computed bankfull values. Boundaries defined in the 2d\_bc\_Name\_L.shp file require a water surface elevation. Boundaries defined in the 2d\_sa\_Name\_R.shp file require discharge inputs. For outflow regions, discharge should be represented by a negative number. Units should be the same as defined in the projection for the project. Save, keeping the format as .csv.

## The next several steps up to and including step 3.33 are only used if you are spatially distributing Manning’s n bed roughness on the basis of map polygons, with each polygon having its own n-value. We are not doing these steps in the RosgenC4 exercise. You must already have a polygon SHP file containing all your roughness facies, a field that has a numerical ID for each material type, and a field with the Manning’s n value for each polygon. Begin with a version of this file that is “dissolved”, so there is only 1 polygon per Manning’s n value. Note that this file does not have to cover the whole model domain or active area as long as the uncovered area all has the same Manning’s n-value, which will be assigned as a default. Thus, you may either have a complete map or just a map with the areas that deviate from default.

## For the file 2d\_mat\_empty\_R in model/gis, now rename \_empty\_ to \_Name\_. Add to your ArcGIS map.

## Start Editor and use the Edit tool to select all the Manning’s n polygons. Right click on the map, select Copy from the contextual menu, right click again, select paste, and then select the option to paste this polygon into 2d\_mat\_Name\_R.shp.

## Open the attribute table for 2d\_mat\_Name\_R.shp and delete out the empty polygon in the first row. Type in unique ID numbers for each polygon in the Materials Field. It does not matter what order is used, but take note of these ID’s along with the Manning’s n value for each one and a comment about what kind of bed facies that ID represents. Do not use a material ID of 0. These will be used later. Save edits and stop Editor.

## Tuflow does not allow any “dissolved”, multipart polygons, so each individual polygon must have its own representation. If you do have a multipart polygon file at this point, then you must use Arc’s tool called Multipart To Singlepart to break it into individual polygons. The reason to keep the file dissolved prior to now is to reduce the number of material ID #s you have to manually type in, but now the file has to have a unique singlepart polygon for each polygon in the SHP file.

## *Alternative for Materials: If having trouble using a materials shapefile, materials can also be defined using a raster of material IDs or a raster of Manning’s n values. Substitute the appropriate command in the Tuflow Geometry Control file (to be defined in section 5): a raster of material IDs: Read GRID MAT == <grid\_file> OR a raster of Manning’s N values: Read GRID CnM == <grid\_file>*

## This step is required regardless of whether you use Materials polygons or not. In Windows, navigate to the file /model/materials.csv. Add the name of your project to the beginning of the file name (NAME\_materials.csv). Open this file in MS Excel. Based on your notes, manually type in the correct Manning’s n value and description for each material ID. The template file has 4 materials, so add or delete rows as needed. There must be at least 1 material with a realistic Manning’s n value for the model to work. For the RosgenC4 exercise, delete materials 2, 3, and 4. Set the manning’s n for material 1 to 0.035. . Save, keeping the format as .csv.

## For the file 2d\_po\_empty\_P.shp in model/gis, now rename \_empty\_ to \_Name\_. Add to your ArcGIS map. This file will contain the monitoring points where you will track hydraulics through time.

## Start Editor and create points within the “code” area where you are pretty sure it will be wet. I recommend creating points from upstream to downstream as well as including a point relatively close to each entrance and exit as well as points in a variety of easy and difficult hydraulics based on your expectations and what you see in your aerial imagery. For example, put a couple points on riffles, chutes, eddies, and simple straightaways. These points will help you insure the model has converged to the level of precision you desire in different settings. . For the points near the flow boundaries, put them a little more than 1 bankfull width upstream so that they are away from any undesirable boundary effects.

## When done adding points, open the attribute table for 2d\_po\_Name\_P.shp and delete out the empty polygon in the first row. Then in the “Type” field, use the field calculator to make all the entries “V\_”. The underscore is needed, as it is a 2-character code. Type in a label for each monitoring point, which could be a simple ordered numbering or could have names related to the setting. I recommend adding “\_start” and “\_end” to the points at the upstream and downstream ends of the model domain, respectively. The expectation is that the downstream point will be last to converge, but that is not always the case. Save edits and stop editing.

## For the file 2d\_po\_empty\_L.shp in model/gis, now rename \_empty\_ to \_Name\_. Add to your ArcGIS map. This file will contain the monitoring XS lines where you will track discharge through time through the model domain.

## Start Editor and create XS lines across the whole “code” area where you are pretty sure it will be wet. Each line should be created from left to right **from the perspective of looking downstream**. This ensures that the outputs will be positive. If you forget, it is not important, but the discharges will have negative values, which doesn’t matter. For the RosgenC4 exercise, when oriented from right to left, the river’s left is on the screen’s bottom. Try to make the lines as perpendicular to what you think the flow direction will be as possible, and you’ll want to check these against the model results when the simulation is done to insure that the computed Q is legitimately representative of a cross-section. I recommend creating lines from upstream to downstream as well as including a line close to each entrance and exit, but again, not within 1 bankfull width of it to avoid boundary effects. Additional XS lines should be spaced in roughly uniform distances down the domain (for long reaches) to make sure that mass is conserved at the XS level down the whole river and/or to see how flow accretions are aggregating. When placing XS lines, try to choose XS locations where the channel is more uniform to improve the quality of the discharge estimate.

## When making these XS lines for the PO\_L file, they must be entirely within the code polygon. If you later make the code polygon much smaller, then you’ll have to clip these down as well.

## When done adding lines, open the attribute table for 2d\_po\_Name\_L.shp and delete out the empty polygon in the first row. Then in the “Type” field, use the field calculator to make all the entries “Q\_”. The underscore is needed, as it is a 2-character code. Type in a label for each monitoring point, which could be a simple ordered list or could have names related to the setting. I recommend adding “\_in” and “\_out” to the lines at the upstream and downstream ends of the model domain, respectively. The expectation is that the downstream line will be last to converge, but that is not always the case. Save edits and stop editing.

# Set up the Name\_002.tcf file

## In Windows, navigate to the runs folder. Copy the file Name\_001.tcf, paste a copy in the same folder, and rename it as Name\_002.tcf. Again, Name\_ is VanillaC4\_ for now.

## Add a new first line to the file with the following text:

Demo Model == ON

## Open Name\_002.tcf in Notepad++. Add an “!” at the start of line 4 (about writing empty GIS files) followed by a space. This action will turn off this command and just leave it there as a comment.

## **We are doing this tutorial in SI units, but for future reference IF** you are working in US Customary units (i.e., US foot), then on the next line add the following: (again, do not use this for this tutorial!)

Units == US Customary

## Add a new line to the file with the following text:

Solution Scheme == HPC

## IF YOU HAVE AN NVIDIA GPU CARD ON YOUR COMPUTER and a license for Tuflow HPC, then on the next line, paste in the text below to turn on the HPC solver using GPU hardware

Hardware == GPU

## IF YOU DO NOT HAVE AN NVIDIA GPU CARD ON YOUT COMPUTER but you do have a paid license for Tuflow HPC, then on the next line, paste in the text below to run the HPC solver using CPU hardware. Otherwise, if you are running Tuflow Classic in DEMO mode, then ignore this line as unnecessary:

Hardware == CPU

## Press return twice to leave a blank line before adding more.

## Write in the following lines:

Geometry Control File == ..\model\NAME\_002.tgc

BC Control File == ..\model\NAME\_002.tbc

BC Database == ..\bc\_dbase\2d\_bc\_NAME.csv

Read Materials File == ..\model\NAME\_materials.csv ! This provides the link between the material ID defined in the .tgc and the Manning's roughess

Read GIS PO == ..\model\gis\2d\_po\_NAME\_P.shp ! Monitoring point locations

Read GIS PO ==..\model\gis\2d\_po\_NAME\_L.shp ! Monitoring XS lines

## For each line in the previous step, replace NAME\_ with the project prefix. These lines tell Tuflow where to look for the various files that control the simulation.

## Create a blank line and then paste in the four lines below, and these control aspects of the hydraulics of the simulation. Here are commentson the next four lines: Note that you should put the water surface elevation from your downstream outlet XS on the IWL line in place of 9999. Note that the Cell Wet/Dry Depth command is just for model computation to determine when water can flow into a cell during the next timestep- it does not literally force a cell to be dry. Conceptually, if each cell is a cube with differing heights, the cell’s “model height” is the wet/dry depth plus its elevation. Thus, water has get high enough to flow over/into that cell from an adjacent cell. But once it does, it doesn’t have to necessarily “fill” to the value of this parameter to be considered wet. A cell can have any depth in it in the end. The default value for Tuflow is 0.002 (and the default value is in SI units of meters, so that’s 2 mm). For real rivers where such thin skimming flow is not sensible relative to the grain size present, then a larger number would be more appropriate, but how large is uncertain and not well tested. Thus, we set the value here to 0.01 (i.e., 0.01 m = 1 cm). if you truly want to force cells below a threshold depth to be dry, then there is the command, ‘Map Cutoff Depth,’ that would change the cutoff depth for map outputs only. We will not use that here.

Viscosity Formulation == SMAGORINSKY

Viscosity Coefficients == 0.5, 0.005

SET IWL == 7.3 ! matches the downstream WSE

Cell Wet/Dry Depth == 0.01

## Create a blank line and then paste in the three lines below, and these control the simulation itself. The end time of 2 hours is adjustable based on however long you think it will take to run the simulation. The first time step is determined using the formula in the comment, so it should be ~0.2-0.5 times the metric cell size. Note that Tuflow HPC uses an adaptive time step algorithm anyway, so this value will change in the simulation anyway- unless you do not have an NVIDIA GPU card and are not using Tuflow HPC, in which case you will be running this on your CPU using Tuflow Classic and this choice does matter. If it is too small or too high, it could make the model unstable.

Start Time == 0 ! Start Simulation at 0 hours

End Time == 2 ! End Simulation at 2 hours

Timestep == 2.5 ! Use a 2D time step that is ~1/4 of the grid size in m (10 m \* 0.25 -> 2.5 s)

## Note that for the end time, we don’t actually know right now how long the model is going to take to fully solve. This is something we will have to evaluate later and refine as needed. For now, use 2 hours as a default for this example.

## Note that for the timestep, Tuflow HPC computes and adjusts its own timestep, so this is not critical, whereas for Tuflow Classic this is the fixed value you set, so it does matter.

## Create a blank line and then paste in the lines below, and these control the output. Adjust the CPU time at which you want map output to begin (i.e., no need to map output in the first few hours if the model is not even close to finished). Map output interval is up to you, and just depends on how much disk space you have available. For initial runs, it helps to have a short interval so you can figure out when the model is converging, while for final runs when you only care about the final output, then you can have it output less often. For the values in red below, those are user adjustable. Start with 0 when in doubt, but to suppress excessive model output that is too early in the simulation, you can use a higher number based on your judgment.

Log Folder == Log ! Redirects log output (eg. .tlf and \_messages GIS layers to the folder "log"

Output Folder == ..\results\002\ ! Redirects results files to TUFLOW\Results\002\

Write Check Files == ..\check\002\ ! Specifies check files to be written to TUFLOW\check\002\

Map Output Format == GRID XMDF ! Output directly to GIS (grid) as well as SMS (xmdf compact) format

Map Output Data Types == h d n V BSS dt ! Output: WSE, Depth, Manning’s n, Velocity, Bed Shear Stress, timestep

Start Map Output == 0 ! Start map output at 0 hours

Map Output Interval == 600 ! Output every 600 seconds (10 minutes)

GRID Map Output Data Types == h d n V BSS dt

Time Series Output Interval == 60 ! time interval of output at points in seconds

## Save the name\_002.tcf file and close it.

# Set up the Name\_002.tbc and Name\_002.tgc files

## In Windows, navigate to the model folder and rename the files Name\_002.tgc and Name\_002.tbc, replacing Name\_ with the project name.

## Open Name\_002.tbc with Notepad++ and in each line, replace Name\_ with the project name. One at a time, highlight just each file path and use the “Alt-space” macro shortcut in Notepad++ to make sure it opens the file at that path. For any .SHP file, what opens will look weird, because that is a binary file, but that is fine. The key is to make sure it opens a file. If no file opens or a warning comes up, it means that the path is wrong and it needs to be fixed. Save the file and close it.

## Open Name\_tgc with Notepad++ and in each line, replace Name\_ with the project name.

## Change the cell size to 10, which is 10 m. For the first model run, it is recommended to use a moderately large cell size so that the model will run quickly and then you can see how much of the “code” area is actually necessary. However, if the cells are very large, then it is possible that the inflow “sa” polygons will miss all cell midpoints and create an error. To avoid that, simply use large “sa” polygons for the first run or else try it and then you can fix the problem as part of the model check steps below, and then re-run it after fixing it. For an initial run with a large grid though, there is no harm in oversizing the SA polygons.

## Using the notes you took back in the step where you created the file Name\_coderectangle\_buffer\_R.shp, recall that the wetted area is 762 m long and 129 m wide. Well, these numbers are not divisible by 10 and for this simple exercise we want to have the rectangular grid be a nice multiple of 10. Therefore, we’ll round up and make the dimensions as 770 m long and 140 m wide. Enter these numbers to define the Grid Size as 770,140.

## Set the Zpts value to a high level about your terrain, so that if there is a place in the grid that is not within your DEM, it will stand out obviously so you can find and fix it. For the RosgenC4 exercise, the max elevation is 1011.5 m, so use 1050 m.

## Set the default material ID # you want the grid to take for cells that are not within your Manning’s n polygons. Usually this is 1.

## As this exercise is not using roughness polygons, add an exclamation mark and space at the start of the line that says “READ GIS MAT’. This will turn off this line, as we have no .shp input file for that.

## Save the file and close it.

# Review all files

## At this point the setup of the Tuflow model is complete. Now it is time to go back and review every final file needed for the run. You do not have to review every step in the workflow, though that might be wise, but you do have to at least put your eyes on every final file used by Tuflow and make sure everything is correct.

## The best way to review your files is to open the Name\_002.tcf file and work your way down line-by-line checking everything. For each line with a file path, use the Alt-space shortcut to make sure that the path is correctly specified and opens the indicated file. For all other lines, review the text.

## When you get to the Name\_002.tcf file lines that call the Name\_002.tgc and Name\_002.tbc files, open those and review them line-by-line as well, with the same checks as in the previous step.

# Run Tuflow Classic or Tuflow HPC

## In Windows, navigate to the runs folder. Copy the file Name\_run\_001\_TUFLOW.bat, paste a copy in the same folder, and rename it as Name\_run\_002\_TUFLOW.bat, replacing Name with the project name, VanillaC4.

## To open and edit the file Name\_run\_002\_TUFLOW.bat, do not double click on it, but instead right click on it and choose “Edit” from the contextual menu. You can edit this in Notepad or Notepad++, whichever you prefer. In the file, change the file name that is called from Name\_001.tcf to VanillaC4\_002.tcf. Save and close.

## If you have a Tuflow license, make sure the Tuflow hardware USB dongle is plugged into a USB port on the computer (can be in a port on a USB multi-port adapter as well) and that the Codemeter software is installed.

## Double click Name\_run\_002\_TUFLOW.bat and the model will run.

## Watch the DOS console window that opens up. It will scroll fast, so don’t worry about that. If an error comes up that shuts down the model altogether, it is mostly like that a file cannot be found, which means one of the file paths was wrong or the indicated file is not where it is supposed to be. Otherwise, the model will run and information will be saved in the \runs\log folder.

## As the model runs, first it will do a “pre-processing” sequence of steps that will build the model and second it will solve the model. While the model is building, leave everything alone in the Tuflow project folder. After it starts solving, you can start mucking around if you want, especially in the check folder.

## In ArcGIS, navigate to the check/002 folder and add the file Name\_002\_grd\_check\_R.shp to ArcMap where the rest of your project GIS files are open. Change the file to have hollow polygons.

## Check that the grid is entirely within the region for which there is a DEM. It is ok if some edges of the grid are outside the “code” polygon, but everything must be within the DEM area, because Tuflow needs elevations at grid corners and midpoints, so those have to be in the DEM area.

## For Tuflow Classic, the speed at which your model will solve depends on your CPU speed primarily, but if there has to be a lot of read/write time, then your throughput and RAM speeds can matter as well.

# Model Performance Checks

## The first step in reviewing a completed model is to check the Name\_002.tlf file within runs/log. “tlf” stands for Tuflow log file. Navigate to this folder and open this file in Notepad++.

## Use the search function in Notepad++ to look for all occurrences of the words error, warning, check, and messages. Scroll through those occurrences and see if any of those are indicating problems. If so, act accordingly per the Tuflow wiki that lists all error codes.

## In ArcMap, load the file runs/log/Name\_002\_messages\_P.shp. If there were no problems, then this will be an empty SHP file. If there are problems, then zoom to each point and review.

## Open the runs/log file Name\_002.gpu.tlf. This file contains the log of the DOS console window from when the model was running. Scroll to the end of it and work up. It also indicates whether there were any warnings or messages. For Tuflow Classic, pay attention to the “MCE”, which is the mass conservation error (this is not relevant to Tuflow GPU).

# Using model results

## Follow the model review checks in the 2D modeling textbook by Pasternack (2011). Note that Tuflow GPU exactly conserves mass, so there cannot be any mass error, so those checks are unnecessary.

## In the results folder, find a timestep .flt file. Review this model timestep information as a method to identify possible locations containing data input errors, see: https://wiki.tuflow.com/index.php?title=Tutorial\_Module03

## Model results are stored in the results folder. If you added PO files, you will have some .csv results files that show the time series of velocity (points) and discharge (lines), so those are a good place to start. Check to make sure that all points and lines have converged through time to a constant value. If any point is fluctuating a lot at the end or still trending, then the model is either unstable or not done. Note that values can fluctuate depending on the setting of the point. Points in a uniform flow area would likely not fluctuate much, whereas points near the edge of an eddy or a flow obstruction could vary more.

## Using Tuflow GPU, you can assess the time step that the model stopped. Open the appropriate .gpu.tlf file in the /runs/log for the model run. Scroll to the bottom and see where the volume and the nwet (number of wet cells) columns each reach a steady level or fluctuation, not trending up or down. If either column is still trending, the model is not finished. Second, use the PO file .csv results to see at what time step the results have converged. In general, higher flows likely take less time to run than lower flows, but it could be the opposite depending on the situation. Therefore, it is best to run your highest and lowest flows first, so you can see the range of time steps that it takes and then adjust the model run time accordingly. The .gpu.tlf file and the PO .csv file are where you look to see how much time it took to run a model.

## XMDF files are for use with SMS and .flt files are for use with ArcGIS. Arc can read .flt files, but some functions require you to convert them to a raster before using them (though the raster calculator does not).

## Use SMS to get a quick look at all the results. Open SMS and then drag and drop the.2dm file in the results/002 folder onto the SMS window. After that loads, drag and drop the XMDF file. SMS will show a list of times in the window, so click on the first time and then use the down arrow to quickly scroll through the time series to see how the model went. You can change the contours and vectors as desired.

## Alternately, you can use ArcGIS to set the spatial referencing for the .flt files and then open them in ArcMap. Then you can compute Velocity “DEM” difference maps between time steps, and review the bulk statistics of velocity changes through time to decide when the model converged to an acceptable level of precision, such as 0.001 m/s. The same thing could be done by scripting in R or Python.

# Revising the model

## Now that an initial run has been done with a coarse grid, it is possible to review all the GIS inputs and decide whether you want to modify them for subsequent runs at higher resolution. Check to be sure that the monitoring lines are perpendicular to the flow and that they are not too near any eddies or islands as this can cause excessive fluctuations.

## The final depth raster may be used to significantly reduce the “code” area of the model run, which will reduce run times for the model. To do this, use the raster calculator Con() function to both convert the raster to a constant value of 1 and save the output as a raster in .tif or .img format. Then use the conversion tool to turn that into a polygon, keeping the “simplify” option unchecked so the polygon exactly follows the raster boundary. Then buffer the polygon out by 2 times cell size to be safe. Edit the buffer polygon output file to delete any extra small polygons it creates and just keep the main polygon. Also, edit the vertices of the buffer polygon to have straight lines at the flow boundaries (which then means you need to copy/paste/rename and revise the flow boundary BC and SA SHP files as well to match the new code polygon). If running the model with a smaller grid size, the SA polygons should be edited to only cover a few grid cells.

## For each file you want to revise, grab a new \_empty\_ file from the /model/gis/empty folder and use the steps from the GIS prep section above to copy/paste/rename the revised features from the source files you just made to these empty files. Delete out the blank row in each empty file when you edit it. For the filenames, add a numerical iteration to the file name each time you change it, so if you change the code file, then change \_Name\_ to \_Name\_003\_ in the filename. This way the filename will match the model run that it goes with, as this will be used for the next model iteration, which would be Name\_003.tcf.

## Note that all this GIS revision should only be done when the run time is much greater than the time it takes to refine the code polygon. For example, if it takes you 30 minutes to make the new code polygon and revise the BC and SA files as well, but the run time is only 30 minutes, then it is not worthwhile. Also, if your final goal is a 1 m grid and your first run is a 30 m grid, then it might make more sense to get down to a 10 m grid before doing this step. Running a 30 m coarse grid first is just useful to very quickly confirm that everything in the model is working as you expected. Then you can refine down to a moderate resolution, run it, get a good code area, and then just do one iteration of the code, BC, and SA files before running your target final resolution.

## It is not necessary to refine the code area very tightly for each discharge, and if you plan to run multiple discharges, then it is best to keep a moderate extra active area to accommodate higher flows. You have to decide on the tradeoff between lowest computation time with the smallest code domain versus more flexibility of having a larger code domain.

## When you go to revise the .tcf, .tbc, and .tgc files, do not write over the existing Name\_002.\* files. Instead, copy those files and past the copies into the same folder. Then set the name to Name\_003.\* and edit those. Each time you make a change to the model design, simply iterate the file name of the required files that changed. The Name\_###.tcf file will need to be iterated for every new model attempt you do, but the rest of the files do not have to be copies and iterated. Only add a number when you change a file. For example, if you change the code domain, then name the new of that file should be updated with \_Name\_###\_, but if you don’t change anything else, then you don’t need to add the \_###\_ text to every single file. Also, for every file whose name you change, you must go into the new tuflow control files and update the file paths or else the model will use the old versions form the previous run.

## Every time you iterate a model, be sure to change the file path for the check and results files, say from a folder called 002 to one called 003. Otherwise, the model will overwrite on the previous run’s results.

## Use either a text file or the template Tuflow modeling log .xls file to keep track of exactly what changes were made form run to run. It does no good if you make 20 simulations and no one understands what was changed. Even though one could study all the files to figure it out, it is poor practice as a modeler and a scientist to do that. Keep good notes for yourself and future generations.

# Storing files

## While you are developing a model, it helps to have all the files in the results folder for review as you make changes.

## Once you have the final model results you want, there is no reason to keep the model outputs from earlier developmental simulations. Delete those. Remember, Tuflow is a procedural model, not a hardwired model, so you can always recover the results by re-running the developmental models, which makes for great transparency without having to bloat the file server.

## For the final model run, you do not have to save all the output files. You can try compressing them into a .zip folder to see if that saves space, but if not, then delete all the files, except those for the final 2 time steps. We use the final time step, but we keep the second to last as a backup in case the last one gets corrupted. Also, it is most likely that the model completed long before the last 2 so they should be almost identical.

# Additional Options for Running Models in Tuflow

## **Start a model run from previous result**: A model run can be kick-started by using the height result .flt file from a previous model run. To start a model from a previous result, use the code Read Grid IWL == <height .flt file path> in the Tuflow geometry control file (.tgc) and remove or comment out the ‘Set IWL ==’ line in the .tcf file.

## **Double precision version of Tuflow GPU**: Always run the models in single precision first. Double precision models can be up to four times slower. If the steady state results exhibit oscillations in the water level, flow or velocity, run the model in double precision.

## To run a model in double precision, call the double precision Tuflow executable file (TUFLOW\_iDP\_w64.exe) in the batch file within the Set TF\_EXE command. If a model is started in double precision, it will trigger an error 2420 message by default. Therefore, also set these two commands in the TCF file: Model Precision == DOUBLE and GPU DP Check == OFF. The second line will bypass the error message.

## **Run multiple models in a batch**: The batch file and other control files can be configured to sequentially run multiple model simulations without pause between the runs. This set up can also allow the following model run to be started from the results of the previous model. In this case, the models must be ordered from lowest to highest discharge. Follow the below example steps A through F to reorganize the Tuflow files to run multiple models from one batch file.

Example Model Name: H20DPD

Example Scenario Names: 03ft or 10ft

Example Event Names: backwater\_fill, LiDAR\_543pt5, RTK\_653, and Kayak\_671

1. SET UP BOUNDARY CONDITIONS:

BC Data files: In the bc\_dbase folder, create a file for each event containing the boundary condition information and using a consistent naming system for each event name, such as **NAME\_bc\_data\_EVENT.csv**. If backwater filling flows are needed, create a file with the event name of backwater\_fill. At time zero, this file should include the higher flows that fill up the backwater areas and for the last few time steps, it should end with the conditions of the lowest flow event.

BC Database Call file: Create a **2d\_bc\_NAME\_###.csv file.** For each cell of the source column**,** type NAME\_bc\_data\_ \_ \_event\_ \_.csv. *Note the spaces are just to show the number of underscores; don’t use spaces in the file names.*

1. SET UP TUFLOW CONTROL FILE:

Create a copy of a previous Tuflow control file and rename it **NAME\_~s1~\_~e1~\_###.tcf**.

* Update appropriate file names for NAME\_###.tgc, NAME \_###.tbc, and 2d\_bc\_ NAME \_###.csv. We will create the TGC and TBC in the next steps.
* Use ! to comment out Set IWL and End Time lines. These will be defined later per model.
* Update Output Folder and Write Check Files lines using <<~s1~>> and <<~e1~>> as placeholders for scenario and event names in file names.
  + Output Folder == ..\results\<<~s1~>>\_<<~e1~>>\
  + Write Check Files == ..\check\<<~s1~>>\_<<~e1~>>\
* Add this line at the very bottom. We will create this file in the next steps: Event File == Events\_NAME\_023.tef

1. SET UP GEOMETRY CONTROL FILE:

Copy and renumber a previous **NAME\_###.tgc** file. If any of the models use different grid sizes, replace the Cell Size == line with the following text and appropriate cell sizes.

! GRID SIZES

If Scenario == 03ft

Cell Size == 3

Else If Scenario == 10ft

Cell Size == 10

End If

To start each model using the results from the previous model, at the end of the TGC file copy and paste the following text. Update event names to match names used in bc\_data files and file output paths as specified in the TCF file. The time in the results file name should be the same as the end time of each model. Add or remove lines as needed.

! INITIAL WATER LEVEL COMMANDS

If Event == LiDAR\_543pt5

Read Grid IWL == ..\results\10ft\_backwater\_fill\grids\H20DPD\_10ft\_backwater\_fill\_023\_h\_020\_00.flt

Else If Event == RTK\_653

Read Grid IWL == ..\results\03ft\_LiDAR\_543pt5\grids\H20DPD\_03ft\_LiDAR\_543pt5\_023\_h\_020\_00.flt

Else If Event == Kayak\_671

Read Grid IWL == ..\results\03ft\_RTK\_653\grids\H20DPD\_03ft\_RTK\_653\_023\_h\_020\_00.flt

End If

1. SET UP BOUNDARY CONTROL FILE:

If any of the models use different source area polygon files for the different cell size models, copy and renumber a **NAME\_###.tbc** file. Replace the Read GIS SA == line with the following text and update appropriate cell sizes and file paths. If all runs use the same source area polygons, this step is not necessary.

! SOURCE AREA POLYGONS

If Scenario == 03ft

Read GIS SA == ..\model\gis\2d\_sa\_H20DPD\_QT\_R\_3ft\_LowQ.shp

Else If Scenario == 10ft

Read GIS SA == ..\model\gis\2d\_sa\_H20DPD\_QT\_R\_10ft\_LowQ.shp

End If

1. SET UP EVENT FILE:

In the Runs folder, create a new text file with the name **Events\_NAME\_###.tef**. Copy and paste the following text into the document. This file defines the order of the model runs by event name. These should be lowest to highest flow. BC Event Source defines the ‘\_ \_event\_ \_’ text as the event name in the bc database file. Other commands can be added to allow you to define parameters that are different for each model. Except for the model end times, if the commands are the same, it makes sense to keep it in the TCF file.

Since this is the last file called in the TCF, commands here will override previous commands in the TCF. Commands that could be helpful in this file: End Time with Start Map Output, Model Precision with GPU DP Check, Maximum Courant Number, etc.

Define Event == backwater\_fill

BC Event Source == \_\_event\_\_ | backwater\_fill

Start Map Output == 18 ! Start map output at X hours

End Time == 20

End Define

!-----------------------------------------------------------------------------

Define Event == LiDAR\_543pt5

BC Event Source == \_\_event\_\_ | LiDAR\_543pt5

! Model Precision == DOUBLE

! GPU DP Check == OFF

! Maximum Courant Number == 0.5

Start Map Output == 28 ! Start map output at X hours

End Time == 30

End Define

!-----------------------------------------------------------------------------

Define Event == RTK\_653

BC Event Source == \_\_event\_\_ | RTK\_653

Start Map Output == 18 ! Start map output at X hours

End Time == 20

End Define

!-----------------------------------------------------------------------------

Define Event == Kayak\_671

BC Event Source == \_\_event\_\_ | Kayak\_671

Start Map Output == 18 ! Start map output at X hours

End Time == 20

End Define

1. SET UP BATCH FILE:

Create a new **NAME\_run\_###.bat** batch file. Right click and edit. Paste the following text into the document:

Set TF\_SP="C:\Program Files\Tuflow\_w64\TUFLOW\_iSP\_w64.exe"

Set TF\_DP="C:\Program Files\Tuflow\_w64\TUFLOW\_iDP\_w64.exe"

Start "TUFLOW" /min /wait %TF\_SP% -b -x -s1 10ft -e backwater\_fill H20DPD\_~s1~\_~e1~\_023.tcf

Start "TUFLOW" /min /wait %TF\_SP% -b -x -s1 03ft -e LiDAR\_543pt5 H20DPD\_~s1~\_~e1~\_023.tcf

Start "TUFLOW" /min /wait %TF\_SP% -b -x -s1 03ft -e RTK\_653 H20DPD\_~s1~\_~e1~\_023.tcf

Start "TUFLOW" /min /wait %TF\_SP% -b -x -s1 03ft -e Kayak\_671 H20DPD\_~s1~\_~e1~\_023.tcf

Update the file paths in the first two lines to the Tuflow single and double precision executable files on your computer, if both are needed. Each of the next lines represents a model event. The scenario is defined after *–s1* as 10ft or 03ft (mesh grid size) and the event is defined after *–e* as the event name. If needed, you can change %TF\_SP% to %TF\_DP% to call the double precision software.

## Save and double click on the batch file to start. Check often as Tuflow will create pop-ups if any errors occur.